EMNLP 2014

The 2014 Conference on Empirical Methods In Natural Language Processing

Proceedings of the Conference

October 25-29, 2014
Doha, Qatar

Sponsors

Diamond

Platinum

Gold

Silver

Bronze

Supporter

IBM **Research**

Sponsor of Student Volunteers

©2014 The Association for Computational Linguistics

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-937284-96-1

Preface by the General Chair

Welcome to the 2014 Conference on Empirical Methods in Natural Language Processing.

The EMNLP conference series is annually organized by SIGDAT, the Association for Computational Linguistics' special interest group on linguistic data and corpus-based approaches to NLP. This year the conference is being held from October 25, 2014 (Sat.) to October 29, 2014 (Wed.) in Doha, Qatar.

In the past five years, the EMNLP conference attendance has been continuously growing, reaching just over 500 paying attendees in 2013, and it is nowadays considered as one of the leading conferences in Computational Linguistics and Natural Language Processing.

Given the growing trend, we believed it was the right time to lead EMNLP into an organization structure typical of large and important conferences. Therefore, we proposed several novelties: first of all, a large organization committee consisting of twenty (plus twenty-six area chairs) well-known members of the ACL community, who carried out several tasks required by the new achieved scale.

Secondly, as this is the first conference edition spanning five days, in addition to six workshops, we also selected and included for the first time an excellent selection of eight tutorials. We defined a registration policy that allows the participants to attend any of the tutorials and workshops (held on October 25th and 29th) by just paying a low flat rate on top of the registration fee for the main conference. We believe this can greatly increase the spread of advanced technology and promote a unified view of the techniques and foundations of our research field.

Thirdly, as a standalone conference, EMNLP required the definition of new administrative procedures and policies, regarding sponsorship booklets, double submission, scholarship assignment, and the joint EACL-ACL-EMNLP call for workshop proposals.

Next, EMNLP is finding new ways to foster the dissemination of research work by facing the increasing number of papers to be presented at the conference. Our new approach consisted in presenting posters in nine sessions each proposing a small numbers of papers: this way poster presentations can receive the space and consideration that they deserve. Then, we are adding a surprise in terms of paper presentation and dissemination, which will be unveiled only few days before the start of the conference.

Finally, this is the first time that an ACL conference is largely supported by a government research foundation. The Qatar National Research Foundation (QNRF) has included EMNLP 2014 as one of its local funding events. This enabled EMNLP and SIGDAT to perform unprecedented student scholarship support: more than 30 students were sponsored (partially or entirely) for participating in the conference. The obtained funds also allowed for offering a social dinner free of charge to all the attendees and still closing the conference budget in active, thus creating additional resources that SIGDAT can use to support the upcoming conferences.

The novelties above as well as the traditional activities that the EMNLP conference series proposes to its members could not have been organized without the work of our large committee. In this respect, I would like to thank our PC co-chairs Walter Daelemans and Bo Pang, who greatly used their large experience with program committees of our community for selecting an excellent program.

Special thanks go to our publication chair Yuval Marton, who did a terrific job in organizing and preparing the proceedings. As a side effect of his proactive action, workshop organizers and future publication chairs using the SoftConf START/ACLPUB systems can now streamline the inclusion of workshops and conference schedules in the proceedings, without heavy manual customization.

We are very grateful to Enrique Alfonseca and Eric Gaussier for selecting interesting and successful

workshops and to Lucia Specia and Xavier Carreras, who, for the first time, carried out the new task of selecting tutorials for an EMNLP conference. The workshops and tutorials nicely filled the additional two days of EMNLP, making our conference even more valuable.

Many thanks are due to Katrin Erk and Sebastian Padó, who were challenged by the new activity (for EMNLP) of defining policy for the selection and assignment of participation scholarships to the most deserving students. The uncertainty over the final amount of funds and their diverse nature made this task particularly difficult. Nevertheless, they were able to find appropriate and successful solutions.

As any large conference, we could count on the help of publicity co-chairs to advertise the old and new EMNLP features. We give our gratitude to Mona Diab and Irina Matveeva for their professional work.

Fund hunting is a very important activity for conferences, in this respect, I would like to thank our sponsorship co-chairs, Jochen Leidner, Veselin Stoyanov and Min Zhang, for helping us to look for sponsors in three different continents.

Regarding the SIGDAT side, a special thank is devoted to Noah Smith, who promptly answered any question I came out with. I am also grateful to the other SIGDAT officers (past and new): Eugene Charniak, Mark Johnson, Philipp Koehn, Mark Steedman, who were always there to give suggestions and solutions to critical issues that inevitably arise in any large event.

Many thanks also to Tim Baldwin, Anna Korhonen, Graeme Hirst and David Yarowsky who provided much useful information from past conferences. Last but not least, I would like to thank Priscilla Rasmussen for her help and advice, and her undoubtful qualities of soothsayer regarding the estimation of conference numbers.

Coming back to the sponsor topic, we are enormously thankful to QNRF, for accepting our proposal to fund EMNLP: this has made it possible to sponsor an unprecedented number of students and offer a banquet free of charge to all participants (we needed to create a new level of sponsorship for them, namely, Diamond). We are very grateful to The Qatar Computing Research Institute, which in addition to providing the very valuable Platinum sponsorship, also provided the required man power for organizing the event.

In particular, EMNLP could not be organized in Qatar without the work of Kareem Darwish, the local organization chair. We are also very grateful to Kemal Oflazer, local co-chair and Francisco Guzman Herrera, local sponsorship chair, whose work was determinant to obtain the QNRF sponsorship. We are deeply in debt with the other local organizers, Lluís Màrquez, who also edited the conference booklet, Preslav Nakov, Fabrizio Sebastiani and Stephan Vogel for their help with the daily big and little issues.

Special thanks go to The Carnegie Mellon University in Qatar for helping us with the proposal preparation and management of the QNRF funds and also for supporting us with a Gold sponsorship. Additionally, many thanks go to our silver sponsors, Facebook and Yandex and our bronze sponsor iHorizons, who show the increasing interest of industry in the technology of our community for the design of real-world and high-societal impact applications. In this respect, we sincerely thank Google Inc. and IBM Watson, New York, for supporting the student participation with their scholarships.

Finally, and foremost, thanks to all the authors and conference attendees who are the main actors of this event, bringing the real value to it and determining its success. My personal thanks also go to the entire SIGDAT committee, for choosing me as the chair of this fantastic conference, held in a fascinating venue.

Alessandro Moschitti

General Chair of EMNLP 2014

Preface by the Program Committee Co-Chairs

We welcome you to the 2014 Conference on Empirical Methods in Natural Language Processing.

As in the previous EMNLP, we invited both long and short papers with a single submission deadline. Short papers encourage the submission of smaller and more preliminary contributions.

We received 790 submissions (after initial withdrawals of unfinished submissions and removal of duplicates), of which 28 were rejected before review for not adhering to the instructions in the call for papers regarding paper length or anonymity. The remaining 510 long and 252 short papers were allocated to one of the fourteen areas. The most popular areas this year were Machine Translation, Semantics, and Syntax (Tagging, Chunking, and Parsing).

Reviewing for a conference this size involves an army of dedicated professionals volunteering to donate their valuable and scarce time to make sure that the highest possible reviewing standards are reached. We are very grateful to our 26 area chairs and a programme committee of more than 500 for their efforts. We accepted 155 long and 70 short papers, representing a global acceptance rate of just under 30%. Nine papers accepted by the ACL journal TACL were added to the program.

Based on the reviews and on nominations by the area chairs, 5 long papers were shortlisted for the best paper award. The best paper will be presented in a plenary best paper award ceremony. We would like to thank Mark Johnson and Claire Cardie for their willingness to serve in the best paper award committee that was set up and for providing excellent advice and motivation for their choice.

We are grateful to the authors for selecting EMNLP as the venue for their work. Congratulations to the authors of accepted submissions. To the authors of rejected submissions we would like to offer as consolation the fact that because of the competitive nature of the conference and the inevitable time and space limitations, many worthwhile papers could not be included in the program. We hope the feedback of the reviewers will be considered worthwhile by them and lead to successful future submissions.

We are very grateful to our invited speakers Thorsten Joachims and Salim Roukos. Thorsten Joachims is professor at the Computer Science and Information Science departments at Cornell University and shows how integrating microeconomic models of human behavior into the learning process leads to new interaction models and learning algorithms, in turn leading to better performing systems. Salim Roukos is senior manager of multilingual NLP and CTO of Translation Technologies at IBM T.J. Watson research Center and addresses IBM's approach to cognitive computing for building systems and solutions that enable and support richer human-machine interactions, and remaining opportunities in this area for novel statistical models for natural language processing. We thank them for their inspiring talks and presence at the conference.

We would also like to thank our general chair Alessandro Moschitti for his leadership, advice, encouragement, and support, Kareem Darwish and his colleagues for impeccable cooperation from local organization, and Yuval Marton for doing an excellent job assembling these proceedings.

It was an honour to serve as Programme Chairs of EMNLP 2014, and we hope that you will enjoy the conference and be able to think back later and remember a scientifically stimulating conference and a pleasant time in Doha, Qatar.

Bo Pang and Walter Daelemans

EMNLP 2014 Program Chairs

Organizers:

General Conference Chair

Alessandro Moschitti, Qatar Computing Research Institute

Program Committee Co-Chairs

Walter Daelemans, University of Antwerp
Bo Pang, Google

Workshops Co-Chairs

Enrique Alfonseca, Google Research at Zurich
Eric Gaussier, Université Joseph Fourier (Grenoble I)

Tutorial Co-Chairs

Lucia Specia, University of Sheffield
Xavier Carreras, Universitat Politècnica de Catalunya

Publication Chair

Yuval Marton, Microsoft Corporation

Publicity Co-Chairs

Mona Diab, George Washington University
Irina Matveeva, NexLP

Sponsorship Co-Chairs

Jochen Leidner, Thomson Reuters
Veselin Stoyanov, Facebook
Min Zhang, Soochow University

Student Scholarship Co-Chairs

Katrin Erk, University of Texas at Austin
Sebastian Padó, University of Stuttgart

Reviewing Coordinators

Mark Dredze, Johns Hopkins University
Jiang Guo (Student Volunteer), Harbin Institute of Technology

Area Chairs

Phonology, Morphology, and Segmentation
Tomaž Erjavec, Jožef Stefan Institute

Tagging, Chunking, Syntax and Parsing
Gosse Bouma, University of Groningen
Yuji Matsumoto, Nara Institute of Science and Technology

Discourse, Dialogue, and Pragmatics
Jennifer Chu-Carroll, IBM Watson Research Center
Olga Uryupina, University of Trento

Semantics
Rada Mihalcea, University of Michigan
Sameer Pradhan, Harvard Medical School

Summarization and Generation
Anja Belz, University of Brighton
Dilek Hakkani-Tür, Microsoft Research

NLP-related Machine Learning: theory, methods and algorithms
Ivan Titov, University of Amsterdam
Jerry Zhu, University of Wisconsin-Madison

Machine Translation
Chris Callison-Burch, University of Pennsylvania
Dan Gildea, University of Rochester

Information Retrieval, Text Categorization, and Question Answering
Sien Moens, Katholieke Universiteit Leuven
Hinrich Schütze, Ludwig Maximilian University of Munich

Information Extraction
Doug Downey, Northwestern University
Marius Pasca, Google

Text Mining and Natural Language Processing Applications
Massimiliano Ciaramita, Google
Hwee Tou Ng, National University of Singapore

Sentiment Analysis and Opinion Mining
Yejin Choi, Stony Brook University
Minlie Huang, Tsinghua University

NLP for the Web and Social Media
Irwin King, The Chinese University of Hong Kong
Qiaozhu Mei, University of Michigan

Spoken Language Processing
Pascale Fung, Hong Kong University of Science and Technology
Hugo Van hamme, Katholieke Universiteit Leuven

Computational Psycholinguistics
Sharon Goldwater, University of Edinburgh

Local Organization

Local Arrangements Co-Chairs
Kareem Darwish, Qatar Computing Research Institute
Kemal Oflazer, Carnegie Mellon University – Qatar

Local Sponsorship Chair
Francisco Guzmán, Qatar Computing Research Institute

Conference Handbook Editor
Lluís Màrquez, Qatar Computing Research Institute

Local Organizing Committee
Preslav Nakov, Qatar Computing Research Institute
Fabrizio Sebastiani, Qatar Computing Research Institute

Local QCRI Administration
Kimberly Mathern, Qatar Computing Research Institute
Lawrence Tingson, Qatar Computing Research Institute
Jacqueline Caparas, Qatar Computing Research Institute

Program Committee:

Omri Abend, The University of Edinburgh; Amjad Abu-Jbara, University of Michigan; Eneko Agirre, University of the Basque Country; Cem Akkaya, University of Pittsburgh; Iñaki Alegria, University of the Basque Country (UPV/EHU); Nikolaos Aletras, University of Sheffield; Enrique

Alfonseca, Google; James Allan, University of Massachusetts Amherst; Alexander Allauzen, Université Paris-Sud / LIMSI-CNRS; Waleed Ammar, CMU; David Andrzejewski, Sumo Logic; Gabor Angeli, Stanford University; Stephen Anthony, The University of New South Wales; Jordi Atserias Batalla, Yahoo! Research; Giuseppe Attardi, Università di Pisa; Michael Auli, Microsoft Research; Wilker Aziz, University of Sheffield;

Alexandra Balahur, Joint Research Centre, European Commission; Timothy Baldwin, The University of Melbourne; Borja Balle, UPC; David Bamman, Carnegie Mellon University; Carmen Banea, University of Michigan; Roberto Basili, University of Roma, Tor Vergata; Daniel Beck, University of Sheffield; Kedar Bellare, Facebook; Emily M. Bender, University of Washington; Michael Bendersky, Google; Fabrício Benevenuto, Federal University of Minas Gerais (UFMG); Jonathan Berant, Stanford University; Taylor Berg-Kirkpatrick, UC Berkeley; Nicola Bertoldi, FBK; Steven Bethard, University of Alabama at Birmingham; Chandra Bhagavatula, Northwestern University; Pushpak Bhattacharyya, CSE Department, IIT Bombay; Klinton Bicknell, Northwestern University; Chris Biemann, TU Darmstadt; Alexandra Birch, University of Edinburgh; Arianna Bisazza, University of Amsterdam; Yonatan Bisk, University of Illinois at Urbana-Champaign; Anders Björkelund, IMS, Stuttgart; Graeme Blackwood, IBM Research; Eduardo Blanco, Lymba Corporation; Roi Blanco, Yahoo! Labs; Phil Blunsom, University of Oxford; Bernd Bohnet, University Birmingham; Kalina Bontcheva, University of Sheffield; Houda Bouamor, Carnegie Mellon University; Jordan Boyd-Graber, University of Maryland; Pavel Braslavski, Kontur labs / Ural federal university; Chris Brew, Nuance Communications; Chris Brockett, Microsoft Research; Eric Brown, IBM Research; Paul Buitelaar, INSIGHT, National University of Ireland, Galway; Donna Byron, IBM Watson Solutions;

Aoife Cahill, Educational Testing Service; Erik Cambria, Nanyang Technological University; Marie Candito, Univ Paris Diderot - INRIA - Alpage; Hailong Cao, HIT; Sandra Carberry, University of Delaware; Marine Carpuat, National Research Council; Xavier Carreras, Universitat Politecnica de Catalunya; Francisco Casacuberta, Universitat Politècnica de València; Taylor Cassidy, IBM TJ Watson Research Center; Carlos Castillo, Qatar Computing Research Institute; Asli Celikyilmaz, Microsoft; Daniel Cer, Google; Özlem Çetinoğlu, IMS, University of Stuttgart; Jonathan Chang, Facebook; Berlin Chen, National Taiwan Normal University; Boxing Chen, NRC; Chen Chen, University of Texas at Dallas; Hsin-Hsi Chen, National Taiwan University; John Chen, AT&T Labs-Research; Lin Chen, University of Illinois at Chicago; Colin Cherry, NRC; Jackie Chi Kit Cheung, University of Toronto; David Chiang, USC/ISI; Jinho D. Choi, Emory University; Christos Christodoulopoulos, University of Illinois at Urbana Champaign; Grzegorz Chrupała, Tilburg University; Stephen Clark, University of Cambridge; Shay B. Cohen, University of Edinburgh; Trevor Cohn, University of Melbourne; Gao Cong, Nanyang Technological University; Paul Cook, The University of Melbourne; Bonaventura Coppola, IBM Research;

Bhavana Dalvi, Carnegie Mellon University; bharath dandala, University of North Texas; Dipanjan Das, Google Inc.; Martine De Cock, Ghent University; Adrià de Gispert, SDL Research; Steve DeNeefe, SDL Language Weaver; Pascal Denis, INRIA; Michael Denkowski, Carnegie Mellon University; Leon Derczynski, University of Sheffield; Ann Devitt, Trinity College Dublin; Nicholas Diakopoulos, Columbia University; Markus Dickinson, Indiana University; Michelangelo Diligenti, University of Siena; Georgiana Dinu, University of Trento; Doug Downey, Northwestern University; Eduard Dragut, Temple University; Mark Dredze, Johns Hopkins University; Markus Dreyer, SDL Language Weaver; Gregory Druck, Yummly; Lan Du, Macquarie University; Kevin Duh, Nara Institute of Science and Technology; Greg Durrett, UC Berkeley; Chris Dyer, Carnegie Mellon University; Marc Dymetman, Xerox Research Centre Europe;

Koji Eguchi, Kobe University; Patrick Ehlen, AT&T; Andreas Eisele, DGT, European Commission; Jacob Eisenstein, Georgia Institute of Technology; Jason Eisner, Johns Hopkins University; Micha Elsner, The Ohio State University; Tomaž Erjavec, Dept. of Knowledge Technologies, Jožef Stefan Institute;

Angela Fahrni, HITS gGmbH; Hui Fang, University of Delaware; Lei Fang, Tsinghua University; Benoit Favre, Aix-Marseille University LIF/CNRS; Anna Feldman, Montclair State University; Naomi Feldman, University of Maryland; Minwei Feng, RWTH Aachen University; Song Feng, Stony Brook University; Yang Feng, USC/ISI; Eraldo Fernandes, UFMS; Katja Filippova, Google; Andrew Finch, NICT; Margaret Fleck, Univ. Illinois, Urbana-Champaign; George Foster, NRC; Jennifer Foster, Dublin City University; Anette Frank, Heidelberg Universiy; Stefan L. Frank, Radboud University Nijmegen;

Michel Galley, Microsoft Research; Michael Gamon, Microsoft Research; Kuzman Ganchev, Google; Kavita Ganesan, Univ. Illinois, Urbana-Champaign; Wei Gao, Qatar Computing Research Institute; Claire Gardent, CNRS/LORIA, Nancy; Matt Gardner, Carnegie Mellon University; Dan Garrette, University of Texas at Austin; Guillermo Garrido, NLP & IR Group at UNED; Albert Gatt, University of Malta; Kallirroi Georgila, University of Southern California Institute for Creative Technologies; Shima Gerani, University of British Columbia; Ulrich Germann, University of Edinburgh; Andrea Gesmundo, Google Inc.; Daniel Gillick, Google; Kevin Gimpel, Toyota Technological Institute at Chicago; Filip Ginter, University of Turku; Roxana Girju, University of Illinois, Urbana-Champaign; Dan Goldwasser, University of Maryland; Sharon Goldwater, University of Edinburgh; Juan Carlos Gomez, KU Leuven; Carlos Gómez-Rodríguez, Universidade da Coruña; Matthew R. Gormley, Johns Hopkins University; Joao Graca, Inesc-Id; Spence Green, Stanford University; Edward Grefenstette, University of Oxford; Gregory Grefenstette, INRIA; Weiwei Guo, Columbia University;

Keith Hall, Google Research; Jirka Hana, Charles University; Greg Hanneman, Carnegie Mellon University; Sanda Harabagiu, University of Texas at Dallas; Christian Hardmeier, Uppsala universitet; Kazi Saidul Hasan, University of Texas at Dallas; Katsuhiko Hayashi, NTT CS Lab; Yifan He, New York University; Jeffrey Heinz, University of Delaware; James Henderson, Xerox Research Centre Europe; John Henderson, MITRE; Andreas Henrich, University of Bamberg; Aurélie Herbelot, Universität Potsdam; Tsutomu Hirao, NTT Communication Science Labs.; Graeme Hirst, University of Toronto; Hieu Hoang, University of Edinburgh; Julia Hockenmaier, University of Illinois Urbana-Champaign; Johannes Hoffart, Max-Planck-Institute for Informatics; Ales Horak, Masaryk University; Estevam Hruschka, Federal University of São Carlos; Fei Huang, Temple University; Liang Huang, City University of New York (CUNY); Xuanjing Huang, Fudan University; Mans Hulden, University of Helsinki;

Gonzalo Iglesias, SDL; Diana Inkpen, University of Ottawa; Kai Ishikawa, NEC Corporation; Abe Ittycheriah, IBM;

Heng Ji, Rensselaer Polytechnic Institute; Sittichai Jiampojamarn, Google Inc.; Jing Jiang, Singapore Management University; Yunliang Jiang, Twitter Inc.; Charles Jochim, IBM Research – Dublin; Richard Johansson, University of Gothenburg; Mark Johnson, Macquarie University;

Mijail Kabadjov, DaXtra Technologies Ltd.; Nobuhiro Kaji, University of Tokyo; Min-Yen Kan, National University of Singapore; Pallika Kanani, Oracle Labs; Rohit Kate, University of Wisconsin-Milwaukee; Andre Kempe, Nuance Communications; Jin-Dong Kim, Database Center for Life Science; Kevin Knight, USC/ISI; Philipp Koehn, University of Edinburgh; Oleksandr Kolomiyets, KU Leuven; Mamoru Komachi, Tokyo Metropolitan University; Fang Kong, Soochow University, National University of Singapore; Moshe Koppel, Bar-Ilan University; Anna Korhonen, University of Cambridge; Zornitsa Kozareva, Yahoo!; Mikhail Kozhevnikov, MMCI Cluster of Excellence, University of Saarland; Jayant Krishnamurthy, Carnegie Mellon University; Lun-Wei Ku, Academia Sinica; Sandra Kübler, Indiana University; Marco Kuhlmann, Linköping University; Roland Kuhn, National Research Council of Canada; Shankar Kumar, Google;

Shibamouli Lahiri, University of North Texas; Mathias Lambert, Amazon.com; Man Lan, East China Normal University; Hugo Larochelle, Université de Sherbrooke; Jey Han Lau, King's College London; Florian Laws, University of Stuttgart; Joseph Le Roux, Université Paris Nord; Gi-

anluca Lebani, University of Pisa; Jung-Tae Lee, Naver Corp.; Yoong Keok Lee, MIT; Jochen Leidner, Thomson Reuters; Alessandro Lenci, University of Pisa; Chee Wee Leong, Educational Testing Service; Omer Levy, Bar-Ilan University; Fangtao Li, Google Research; Hang Li, Huawei Technologies; Lishuang Li, Dalian University of Technology; Peng Li, State Key Laboratory of Intelligent Technology and Systems; Tsinghua National Laboratory for Information Science and Technology; Department of Computer Sci. and Tech., Tsinghua University, Beijing, China; Qi Li, Computer Science, Rensselaer Polytechnic Institute; Shoushan Li, Soochow University; Wenjie Li, The Hong Kong Polytechnic University; Xiao Ling, University of Washington; Christina Lioma, University of Copenhagen; Marina Litvak, Shamoon College of Engineering; Kang Liu, Chinese Academy of Sciences; Qun Liu, Dublin City University; Yang Liu, University of Texas at Dallas; Nikola Ljubešić, University of Zagreb; Oier Lopez de Lacalle, IKERBASQUE, Basque Foundation for Science - University of Edinburgh; Bin Lu, City University of Hong Kong; Wei Lu, Singapore University of Technology and Design; Xiaofei Lu, Pennsylvania State University; Marco Lui, University of Melbourne / NICTA VRL; Susann Luperfoy, MIT;

Wolfgang Macherey, Google; Suresh Manandhar, University of York; Gideon Mann, Google Inc; Daniel Marcu, SDL; André F. T. Martins, Priberam, Instituto de Telecomunicacoes; Cettolo Mauro, FBK; Arne Mauser, Google, Inc; Jonathan May, USC Information Sciences Institute; Elijah Mayfield, Carnegie Mellon University; Diana Maynard, University of Sheffield; Julian McAuley, UCSD; David McClosky, IBM Research; Ryan McDonald, Google; Louise McNally, Universitat Pompeu Fabra; Edgar Meij, Yahoo! Research; Adam Meyers, New York University; Haitao Mi, IBM Watson Research Center; Lukas Michelbacher, Center for Information and Language Processing, LMU, Munich; David Mimno, Cornell University; Bonan Min, Raytheon BBN Technologies; Shachar Mirkin, Xerox Research Centre Europe; Margaret Mitchell, Microsoft Research; Samaneh Moghaddam, Simon Fraser University; Saif Mohammad, National Research Council Canada; Behrang Mohit, Carnegie Mellon University; Mitra Mohtarami, National University of Singapore; Karo Moilanen, Dpt of Computer Science, University of Oxford; Christian Monson, Nuance Communications Inc.; Roser Morante, University of Antwerp; Andrea Moro, Sapienza University of Rome; Dana Movshovitz-Attias, Carnegie Mellon University; Thomas Mueller, CIS, University of Munich; Arjun Mukherjee, University of Illinois at Chicago; Dragos Munteanu, SDL Language Technologies; Smaranda Muresan, Columbia University; Gabriel Murray, University of the Fraser Valley; sung-hyon myaeng, Korea Advanced Institute of Science and Technology; Markos Mylonakis, Lexis Research;

Tetsuji Nakagawa, Google Japan Inc.; Preslav Nakov, Qatar Computing Research Institute; Jason Naradowsky, UMass Amherst; Tahira Naseem, MIT-CSAIL; Vivi Nastase, FBK; Roberto Navigli, Sapienza University of Rome; Mark-Jan Nederhof, University of St Andrews; Ani Nenkova, University of Pennsylvania; Graham Neubig, Nara Institute of Science and Technology; Guenter Neumann, DFKI; Vincent Ng, University of Texas at Dallas; Jian-Yun NIE, University of Montreal; Rodney Nielsen, University of North Texas ; University of Colorado; Malvina Nissim, University of Bologna;

Diarmuid Ó Séaghdha, University of Cambridge; Stephan Oepen, Universitetet i Oslo; Kemal Oflazer, Carnegie Mellon University - Qatar; Alice Oh, KAIST; Jong-Hoon Oh, NICT; Kouzou Ohara, Aoyama Gakuin University; Naoaki Okazaki, Tohoku University; Constantin Orasan, University of Wolverhampton;

Martha Palmer, University of Colorado; Sinno J. Pan, Institute for Infocomm Research; Kristen Parton, Facebook; Rebecca J. Passonneau, Columbia University; John K Pate, Macquarie University; Siddharth Patwardhan, IBM T. J. Watson Research Center; Mykola Pechenizkiy, TU Eindhoven; Slav Petrov, Google; Steven Piantadosi, University of Rochester, BCS; Barbara Plank, University of Copenhagen; Simone Paolo Ponzetto, University of Mannheim; Hoifung Poon, Microsoft Research; Maja Popović, DFKI; Matt Post, Johns Hopkins University; Sameer Pradhan, Harvard University; John Prager, IBM Research; Rashmi Prasad, University of Wisconsin-

Milwaukee; Adam Przepiórkowski, Institute of Computer Science at the Polish Academy of Sciences; Stephen Pulman, Oxford University; Sampo Pyysalo, NaCTeM;

Vahed Qazvinian, Google;

Altaf Rahman, Yahoo Labs; Maya Ramanath, IIT-Delhi; Owen Rambow, Columbia University; Sujith Ravi, Google Inc.; Marta Recasens, Google Inc.; Ines Rehbein, Potsdam University; Roi Reichart, Technion - Israel Institute of Technology; Sebastian Riedel, UCL; Korbinian Riedhammer, Int'l Computer Science Institute; Verena Rieser, Heriot-Watt University; Stefan Riezler, Heidelberg University; German Rigau, UPV/EHU; Laura Rimell, University of Cambridge; Alan Ritter, Carnegie Mellon University; Andrew Rosenberg, CUNY Queens College; Michael Roth, The University of Edinburgh; Alla Rozovskaya, Columbia University; Josef Ruppenhofer, Hildesheim University; Vasile Rus, The University of Memphis; Alexander M. Rush, MIT; Delia Rusu, Jozef Stefan Institute;

Markus Saers, Hong Kong University of Science and Technology; Mehdi Samadi, Carnegie Mellon University; Federico Sangati, FBK, Trento; Sunita Sarawagi, IIT Bombay; Anoop Sarkar, Simon Fraser University; David Schlangen, Bielefeld University; Helmut Schmid, CIS, Ludwig-Maximilians-Universität; Tobias Schnabel, Cornell University; Sabine Schulte im Walde, University of Stuttgart; H. Andrew Schwartz, University of Pennsylvania; Holger Schwenk, University of Le Mans; Pavel Serdyukov, Yandex; Hendra Setiawan, IBM T.J. Watson Research Center; Burr Settles, Duolingo, Inc.; Kashif Shah, University of Sheffield; Azadeh Shakery, University of Tehran; Shuming Shi, Microsoft Research Asia; Avirup Sil, IBM Research; Carina Silberer, School of Informatics, University of Edinburgh; Mario J. Silva, IST/INESC-ID; Khalil Sima'an, ILLC, University of Amsterdam; Sameer Singh, University of Washington, Seattle; Kevin Small, Amazon; Otakar Smrž, Seznam.cz; Jan Šnajder, University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, 10000 Zagreb; Richard Socher, Stanford University; Stephen Soderland, University of Washington; Anders Søgaard, University of Copenhagen; Thamar Solorio, UAB; Swapna Somasundaran, Educational Testing Services; Linfeng Song, ICT/CAS; Yang Song, Microsoft Research Redmond; Lucia Specia, University of Sheffield; Valentin Spitovsky, Google Inc.; Vivek Srikumar, Stanford University; Mark Steedman, University of Edinburgh; Benno Stein, Bauhaus Universität Weimar; Michael Strube, Heidelberg Institute for Theoretical Studies; David Suendermann, DHBW Stuttgart; Hisami Suzuki, Microsoft Corporation; Jun Suzuki, NTT CS Lab.; Ben Swanson, Brown University;

Hiroya Takamura, Tokyo Institute of Technology; David Talbot, Google Russia; Partha P. Talukdar, CMU; Chenhao Tan, Cornell University; Joel Tetreault, Nuance Communications; Cindi Thompson, University of San Francisco; Jörg Tiedemann, Uppsala University; Christoph Tillmann, TJ Watson IBM Research; Takenobu Tokunaga, Tokyo Institute of Technology; Kristina Toutanova, Microsoft Research; Ming-Feng Tsai, National Chengchi University; Yoshimasa Tsuruoka, University of Tokyo;

Jakob Uszkoreit, Google, Inc.; Masao Utiyama, NICT;

Andreas van Cranenburgh, University of Amsterdam; Antal van den Bosch, Radboud University Nijmegen; Benjamin Van Durme, JHU; Gertjan van Noord, University of Groningen; Menno van Zaanen, Tilburg University; Yannick Versley, University of Heidelberg; David Vilar, Pixformance GmbH; Aline Villavicencio, Institute of Informatics, Federal University of Rio Grande do Sul; Sami Virpioja, Aalto University; Andreas Vlachos, University College London; Adam Vogel, Stanford University; Vinod Vydiswaran, University of Illinois;

Stephen Wan, CSIRO; Xiaojun Wan, Peking University; Haifeng Wang, Baidu; Lu Wang, Cornell University; Leo Wanner, ICREA and Pompeu Fabra University; Nigel Ward, University of Texas at El Paso; Taro Watanabe, NICT; Bonnie Webber, University of Edinburgh; Furu Wei, Microsoft Research Asia; Albert Weichselbraun, University of Applied Sciences Chur; Gerhard Weikum, Max-Planck Institute for Informatics (MPII); Michael Wiegand, Saarland University; Jason D

Williams, Microsoft Research; Shuly Wintner, University of Haifa; Kam-Fai Wong, Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong; Fei Wu, google; Joern Wuebker, RWTH Aachen University;

Yunqing Xia, Tsinghua University; Peng Xu, Google Inc.; Wei Xu, University of Pennsylvania; Nianwen Xue, Brandeis University;

Bishan Yang, Cornell University; Yi Yang, Northwestern University; Limin Yao, University of Massachusetts, Amherst; Wen-tau Yih, Microsoft Research; Hong Yu, Soochow University; Jianxing Yu, Tencent Inc. Shenzhen, China; François Yvon, LIMSI/CNRS;

Fabio Massimo Zanzotto, University of Rome "Tor Vergata"; Richard Zens, Google; Luke Zettlemoyer, University of Washington; Congle Zhang, University of Washington; Hao Zhang, Google; Joy Ying Zhang, Carnegie Mellon University; Lei Zhang, University of Illinois at Chicago; Min Zhang, SooChow University; Qi Zhang, Fudan University; Yi Zhang, German Research Center for Artificial Intelligence; Yue Zhang, Singapore University of Technology and Design; Jun Zhao, Chinese Academy of Sciences; Kai Zhao, Graduate Center, CUNY; Guodong Zhou, Soochow University; Tom Chao Zhou, Baidu Inc.; Chengqing Zong, Institute of Automation, Chinese Academy of Sciences; and Willem Zuidema, University of Amsterdam.

Additional Reviewers:

Aditya Joshi, Alexander Beloborodov, Ali Elkahky, Alvin Grissom II, Antoine Rozenknop, Burcu Can, Caitlin Richter, Chen Li, Christian Gaida, Dave Carter, Dongwoo Kim, Fandong Meng, Feifei Zhai, Frank Ferraro, Georgeta Bordea, Hadi Amiri, Hector Martinez Alonso, Heng Yu, Henning Wachsmuth, Hugo Escalante, Jacob Andreas, Jan Rygl, Janis Dalins, Jeff Mitchell, Jeffrey Flanigan, Jenna Kanerva, Ji Liu, Ji Ma, JinYeong Bak, Jiri Materna, Jiwei Li, John Wieting, Juhani Luotolahti, Jun Xie, Kai Hakala, Kai Liu, Kepa Sarasola, Kristian Woodsend, Li Dong, Li Wang, Likun Qiu, Loic Barrault, Mark Cusick, Martin Riedl, Michael Voelske, Miikka Silfverberg, Mikael Kågebäck, Milos Jakubicek, Mitchell Koch, Mohamed Yahya, Mohit Iyyer, Muhammad Ibrahim, Muhua Zhu, Nadi Tomeh, Ni Lao, Patrick Lange, Pushpendre Rastogi, Qing Dou, Roy Schwartz, Scott Martin, Shumin Wu, Colorado, Steffen Remus, Suwisa Kaewphan, Tim Rocktaeschel, Tongfei Chen, Travis Wolfe, Veronica Perez-Rosas, Vojtech Kovar, Wei He, Wenbin Jiang, Xutao Li, Zhanyi Liu, Zhaohui Wu, and Zhen Hai.

Invited Speakers:

Salim Roukos, IBM T. J. Watson Research Center

Thorsten Joachims, Cornell University

Table of Contents

Conference Program

Sunday, October 26, 2014

08:00–17:00 ***Registration***

08:00–09:00 ***Refreshments***

Plenary Session

08:40–09:00 *Opening Remarks, QNRF Presentation, Conference Logistics, Paper Selection Process and Statistics*
Alessandro Moschitti, Rekha Pilla, Kareem M. Darwish, and Walter Daelemans

09:00–10:00 Invited Talk: *IBM Cognitive Computing - An NLP Renaissance!*
Salim Roukos

10:00–10:30 ***Coffee Break***

Session 1a: Neural Net Mixer I

10:30–10:55 *Modeling Interestingness with Deep Neural Networks*
Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He and Li Deng

10:55–11:20 *Translation Modeling with Bidirectional Recurrent Neural Networks*
Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker and Hermann Ney

11:20–11:45 *A Neural Network Approach to Selectional Preference Acquisition*
Tim Van de Cruys

11:45–12:10 *Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics*
Douwe Kiela and Léon Bottou

Sunday, October 26, 2014 (continued)

Session 1b: Discourse, Dialogue and Pragmatics

10:30–10:55 *Identifying Argumentative Discourse Structures in Persuasive Essays*
Christian Stab and Iryna Gurevych

10:55–11:20 *Policy Learning for Domain Selection in an Extensible Multi-domain Spoken Dialogue System*
Zhuoran Wang, Hongliang Chen, Guanchun Wang, Hao Tian, Hua Wu and Haifeng Wang

11:20–11:45 *A Constituent-Based Approach to Argument Labeling with Joint Inference in Discourse Parsing*
Fang Kong, Hwee Tou Ng and Guodong Zhou

11:45–12:10 *Strongly Incremental Repair Detection*
Julian Hough and Matthew Purver

Session 1c: Segmentation / Spoken Language

10:30–10:55 *Semi-Supervised Chinese Word Segmentation Using Partial-Label Learning With Conditional Random Fields*
Fan Yang and Paul Vozila

10:55–11:20 *Accurate Word Segmentation and POS Tagging for Japanese Microblogs: Corpus Annotation and Joint Modeling with Lexical Normalization*
Nobuhiro Kaji and Masaru Kitsuregawa

11:20–11:45 *Revisiting Embedding Features for Simple Semi-supervised Learning*
Jiang Guo, Wanxiang Che, Haifeng Wang and Ting Liu

11:45–12:10 *Combining Punctuation and Disfluency Prediction: An Empirical Study*
Xuancong Wang, Khe Chai Sim and Hwee Tou Ng

Session 1p: Machine Translation

10:30–12:10 *Poster Session*
Multiple presenters

Submodularity for Data Selection in Machine Translation
Katrin Kirchhoff and Jeff Bilmes

Improve Statistical Machine Translation with Context-Sensitive Bilingual Semantic Embedding Model
Haiyang Wu, Daxiang Dong, Xiaoguang Hu, Dianhai Yu, Wei He, Hua Wu, Haifeng Wang and Ting Liu

Transformation from Discontinuous to Continuous Word Alignment Improves Translation Quality
Zhongjun He, Hua Wu, Haifeng Wang and Ting Liu

Unsupervised Word Alignment Using Frequency Constraint in Posterior Regularized EM
Hidetaka Kamigaito, Taro Watanabe, Hiroya Takamura and Manabu Okumura

Asymmetric Features Of Human Generated Translation
Sauleh Eetemadi and Kristina Toutanova

Syntax-Augmented Machine Translation using Syntax-Label Clustering
Hideya MINO, Taro WATANABE and Eiichiro SUMITA

Testing for Significance of Increased Correlation with Human Judgment
Yvette Graham and Timothy Baldwin

Syntactic SMT Using a Discriminative Text Generation Model
Yue Zhang, Kai Song, Linfeng Song, Jingbo Zhu and Qun Liu

Learning Hierarchical Translation Spans
jingyi zhang, Masao Utiyama, Eiichiro Sumita and Hai Zhao

Neural Network Based Bilingual Language Model Growing for Statistical Machine Translation
Rui Wang, Hai Zhao, Bao-Liang Lu, Masao Utiyama and Eiichiro Sumita

Better Statistical Machine Translation through Linguistic Treatment of Phrasal Verbs
Kostadin Cholakov and Valia Kordoni

Fitting Sentence Level Translation Evaluation with Many Dense Features
Miloš Stanojević and Khalil Sima'an

A Human Judgement Corpus and a Metric for Arabic MT Evaluation
Houda Bouamor, Hanan Alshikhabobakr, Behrang Mohit and Kemal Oflazer

Learning to Differentiate Better from Worse Translations
Francisco Guzmán, Shafiq Joty, Lluís Màrquez, Alessandro Moschitti, Preslav Nakov and Massimo Nicosia

Two Improvements to Left-to-Right Decoding for Hierarchical Phrase-based Machine Translation
Maryam Siahbani and Anoop Sarkar

Reordering Model for Forest-to-String Machine Translation
Martin Cmejrek

12:10–13:30 ***Lunch Break***

Session 2a: Computational Psycholinguistics

13:30–13:55 *Aligning context-based statistical models of language with brain activity during reading*
Leila Wehbe, Ashish Vaswani, Kevin Knight and Tom Mitchell

13:55–14:20 *A Cognitive Model of Semantic Network Learning*
Aida Nematzadeh, Afsaneh Fazly and Suzanne Stevenson

14:20–14:45 *The Benefits of a Model of Annotation*
Rebecca J. Passonneau and Bob Carpenter

14:45–15:10 *Learning Abstract Concept Embeddings from Multi-Modal Data: Since You Probably Can't See What I Mean*
Felix Hill and Anna Korhonen

Session 2b: Tagging, Chunking, Parsing and Syntax

13:30–13:55 *Go Climb a Dependency Tree and Correct the Grammatical Errors*
Longkai Zhang and Houfeng WANG

13:55–14:20 *An Unsupervised Model for Instance Level Subcategorization Acquisition*
Simon Baker, Roi Reichart and Anna Korhonen

14:20–14:45 *Parsing low-resource languages using Gibbs sampling for PCFGs with latent annotations*
Liang Sun, Jason Mielens and Jason Baldridge

14:45–15:10 *Incremental Semantic Role Labeling with Tree Adjoining Grammar*
Ioannis Konstas, Frank Keller, Vera Demberg and Mirella Lapata

Session 2c: NLP for the Web and Social Media

13:30–13:55 *A Graph-based Approach for Contextual Text Normalization*
Cagil Sonmez and Arzucan Ozgur

13:55–14:20 *Entity Linking on Microblogs with Spatial and Temporal Signals*
Yuan Fang and Ming-Wei Chang

14:20–14:45 *ReNoun: Fact Extraction for Nominal Attributes*
Mohamed Yahya, Steven Whang, Rahul Gupta and Alon Halevy

14:45–15:10 *Hierarchical Discriminative Classification for Text-Based Geolocation*
Benjamin Wing and Jason Baldridge

Session 2p: Semantics

13:30–15:10 *Poster Session*
Multiple presenters

Probabilistic Models of Cross-Lingual Semantic Similarity in Context Based on Latent Cross-Lingual Concepts Induced from Comparable Data
Ivan Vulić and Marie-Francine Moens

Multi-Predicate Semantic Role Labeling
Haitong Yang and Chengqing Zong

Werdy: Recognition and Disambiguation of Verbs and Verb Phrases with Syntactic and Semantic Pruning
Luciano Del Corro, Rainer Gemulla and Gerhard Weikum

Multi-Resolution Language Grounding with Weak Supervision
R. Koncel-Kedziorski, Hannaneh Hajishirzi and Ali Farhadi

Incorporating Vector Space Similarity in Random Walk Inference over Knowledge Bases
Matt Gardner, Partha Talukdar, Jayant Krishnamurthy and Tom Mitchell

Composition of Word Representations Improves Semantic Role Labelling
Michael Roth and Kristian Woodsend

Automatic Domain Assignment for Word Sense Alignment
Tommaso Caselli and Carlo Strapparava

Nothing like Good Old Frequency: Studying Context Filters for Distributional Thesauri
Muntsa Padró, Marco Idiart, Aline Villavicencio and Carlos Ramisch

Aligning English Strings with Abstract Meaning Representation Graphs
Nima Pourdamghani, Yang Gao, Ulf Hermjakob and Kevin Knight

A Shortest-path Method for Arc-factored Semantic Role Labeling
Xavier Lluís, Xavier Carreras and Lluís Màrquez

Semantic Kernels for Semantic Parsing
Iman Saleh, Alessandro Moschitti, Preslav Nakov, Lluís Màrquez and Shafiq Joty

Multi-Modal Models for Concrete and Abstract Concept Meaning
Felix Hill, Roi Reichart, Anna Korhonen

An I-vector Based Approach to Compact Multi-Granularity Topic Spaces Representation of Textual Documents
Mohamed Morchid, Mohamed Bouallegue, Richard Dufour, Georges Linares, Driss Matrouf and Renato de Mori

15:10–15:40 ***Coffee Break***

Session 3a: Sentiment Analysis and Opinion Mining

15:40–16:05 *Explaining the Stars: Weighted Multiple-Instance Learning for Aspect-Based Sentiment Analysis*
Nikolaos Pappas and Andrei Popescu-Belis

16:05–16:30 *Sentiment Analysis on the People's Daily*
Jiwei Li and Eduard Hovy

16:30–16:55 *A Joint Segmentation and Classification Framework for Sentiment Analysis*
Duyu Tang, Furu Wei, Bing Qin, Li Dong, Ting Liu and Ming Zhou

16:55–17:20 *Positive Unlabeled Learning for Deceptive Reviews Detection*
yafeng ren, donghong ji and hongbin zhang

Session 3b: Semantics

15:40–16:05 *Resolving Shell Nouns*
Varada Kolhatkar and Graeme Hirst

16:05–16:30 *A Comparison of Selectional Preference Models for Automatic Verb Classification*
Will Roberts and Markus Egg

16:30–16:55 *Learning to Solve Arithmetic Word Problems with Verb Categorization*
Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni and Nate Kushman

16:55–17:20 *NaturalLI: Natural Logic Inference for Common Sense Reasoning*
Gabor Angeli and Christopher D. Manning

Sunday, October 26, 2014 (continued)

Session 3c: Machine Translation

15:40–16:05 *Modeling Term Translation for Document-informed Machine Translation*
Fandong Meng, Deyi Xiong, Wenbin Jiang and Qun Liu

16:05–16:30 *Beyond Parallel Data: Joint Word Alignment and Decipherment Improves Machine Translation*
Qing Dou, Ashish Vaswani and Kevin Knight

16:30–16:55 *Latent Domain Phrase-based Models for Adaptation*
Cuong Hoang and Khalil Sima'an

16:55–17:20 *Translation Rules with Right-Hand Side Lattices*
Fabien Cromieres and Sadao Kurohashi

Session 3p: Information Retrieval, Summarization and Question Answering

15:40–17:20 *Poster Session*
Multiple presenters

Learning to Translate: A Query-Specific Combination Approach for Cross-Lingual Information Retrieval
Ferhan Ture and Elizabeth Boschee

Semantic-Based Multilingual Document Clustering via Tensor Modeling
Salvatore Romeo, Andrea Tagarelli and Dino Ienco

Lexical Substitution for the Medical Domain
Martin Riedl, Michael Glass and Alfio Gliozzo

Question Answering with Subgraph Embeddings
Antoine Bordes, Sumit Chopra and Jason Weston

Correcting Keyboard Layout Errors and Homoglyphs in Queries
Derek Barnes, Mahesh Joshi and Hassan Sawaf

Non-linear Mapping for Improved Identification of 1300+ Languages
Ralf Brown

A Neural Network for Factoid Question Answering over Paragraphs
Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher and Hal Daumé III

Joint Relational Embeddings for Knowledge-based Question Answering
Min-Chul Yang, Nan Duan, Ming Zhou and Hae-Chang Rim

Adding High-Precision Links to Wikipedia
Thanapon Noraset, Chandra Bhagavatula and Doug Downey

Crosslingual and Multilingual Construction of Syntax-Based Vector Space Models
Jason Utt and Sebastian Padó

Finding Good Enough: A Task-Based Evaluation of Query Biased Summarization for Cross-Language Information Retrieval
Jennifer Williams, Sharon Tam and Wade Shen

Chinese Poetry Generation with Recurrent Neural Networks
Xingxing Zhang and Mirella Lapata

Fear the REAPER: A System for Automatic Multi-Document Summarization with Reinforcement Learning
Cody Rioux, Sadid A. Hasan and Yllias Chali

Improving Multi-documents Summarization by Sentence Compression based on Expanded Constituent Parse Trees
Chen Li, Yang Liu, Fei Liu, Lin Zhao and Fuliang Weng

Analyzing Stemming Approaches for Turkish Multi-Document Summarization
Muhammed Yavuz Nuzumlalı and Arzucan Özgür

Monday, October 27, 2014

08:00–17:00 ***Registration***

08:00–09:00 ***Refreshments***

Plenary Session

09:00–10:00 Invited Talk: *Learning from Rational Behavior*
Thorsten Joachims

10:00–10:30 ***Coffee Break***

Session 4a: Neural Net Mixer II

10:30–10:55 *Evaluating Neural Word Representations in Tensor-Based Compositional Settings*
Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh and Matthew Purver

10:55–11:20 *Opinion Mining with Deep Recurrent Neural Networks*
Ozan Irsoy and Claire Cardie

11:20–11:45 *The Inside-Outside Recursive Neural Network model for Dependency Parsing*
Phong Le and Willem Zuidema

11:45–12:10 *A Fast and Accurate Dependency Parser using Neural Networks*
Danqi Chen and Christopher Manning

Session 4b: Discourse, Dialogue and Pragmatics / Summarization and Generation

10:30–10:55 *Why are You Taking this Stance? Identifying and Classifying Reasons in Ideological Debates*
Kazi Saidul Hasan and Vincent Ng

10:55–11:20 *Chinese Zero Pronoun Resolution: An Unsupervised Probabilistic Model Rivaling Supervised Resolvers*
Chen Chen and Vincent Ng

11:20–11:45 *Unsupervised Sentence Enhancement for Automatic Summarization*
Jackie Chi Kit Cheung and Gerald Penn

11:45–12:10 *ReferItGame: Referring to Objects in Photographs of Natural Scenes*
Sahar Kazemzadeh, Vicente Ordonez, Mark Matten and Tamara Berg

Session 4c: Information Extraction

10:30–10:55 *Unsupervised Template Mining for Semantic Category Understanding*
Lei Shi, Shuming Shi, Chin-Yew Lin, Yi-Dong Shen and Yong Rui

10:55–11:20 *Taxonomy Construction Using Syntactic Contextual Evidence*
Tuan Luu Anh, Jung-jae Kim and See Kiong Ng

11:20–11:45 *Analysing recall loss in named entity slot filling*
Glen Pink, Joel Nothman and James R. Curran

11:45–12:10 *Relieving the Computational Bottleneck: Joint Inference for Event Extraction with High-Dimensional Features*
Deepak Venugopal, Chen Chen, Vibhav Gogate and Vincent Ng

Session 4p: Segmentation, Tagging and Parsing

10:30–12:10 *Poster Session*
Multiple presenters

Syllable weight encodes mostly the same information for English word segmentation as dictionary stress
John K Pate and Mark Johnson

A Joint Model for Unsupervised Chinese Word Segmentation
Miaohong Chen, Baobao Chang and Wenzhe Pei

Domain Adaptation for CRF-based Chinese Word Segmentation using Free Annotations
Yijia Liu, Yue Zhang, Wanxiang Che, Ting Liu and Fan Wu

Balanced Korean Word Spacing with Structural SVM
Changki Lee, Edward Choi and Hyunki Kim

Morphological Segmentation for Keyword Spotting
Karthik Narasimhan, Damianos Karakos, Richard Schwartz, Stavros Tsakalidis and Regina Barzilay

What Can We Get From 1000 Tokens? A Case Study of Multilingual POS Tagging For Resource-Poor Languages
Long Duong, Trevor Cohn, Karin Verspoor, Steven Bird and Paul Cook

An Experimental Comparison of Active Learning Strategies for Partially Labeled Sequences
Diego Marcheggiani and Thierry Artières

Language Modeling with Functional Head Constraint for Code Switching Speech Recognition
Ying LI and Pascale Fung

A Polynomial-Time Dynamic Oracle for Non-Projective Dependency Parsing
Carlos Gómez-Rodríguez, Francesco Sartorio and Giorgio Satta

Ambiguity Resolution for Vt-N Structures in Chinese
Yu-Ming Hsieh, Jason S. Chang and Keh-Jiann Chen

Neural Networks Leverage Corpus-wide Information for Part-of-speech Tagging
Yuta Tsuboi

System Combination for Grammatical Error Correction
Raymond Hendy Susanto, Peter Phandi and Hwee Tou Ng

Dependency parsing with latent refinements of part-of-speech tags
Thomas Mueller, Richárd Farkas, Alex Judea, Helmut Schmid and hinrich schuetze

Importance weighting and unsupervised domain adaptation of POS taggers: a negative result
Barbara Plank, Anders Johannsen and Anders Søgaard

POS Tagging of English-Hindi Code-Mixed Social Media Content
Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali and Monojit Choudhury

Data Driven Grammatical Error Detection in Transcripts of Children's Speech
Eric Morley, Anna Eva Hallin and Brian Roark

12:10–13:30 ***Lunch Break***

12:50–13:30 ***SIGDAT Business Meeting***

Session 5a: Tagging, Chunking, Parsing and Syntax

13:30–13:55 *Improved CCG Parsing with Semi-supervised Supertagging*
Mike Lewis and Mark Steedman

13:55–14:20 *A* CCG Parsing with a Supertag-factored Model*
Mike Lewis and Mark Steedman

14:20–14:45 *A Dependency Parser for Tweets*
Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer and Noah A. Smith

14:45–15:10 *Greed is Good if Randomized: New Inference for Dependency Parsing*
Yuan Zhang, Tao Lei, Regina Barzilay and Tommi Jaakkola

Session 5b: Semantics

13:30–13:55 *A Unified Model for Word Sense Representation and Disambiguation*
Xinxiong Chen, Zhiyuan Liu and Maosong Sun

13:55–14:20 *Reducing Dimensions of Tensors in Type-Driven Distributional Semantics*
Tamara Polajnar, Luana Fagarasan and Stephen Clark

14:20–14:45 *An Etymological Approach to Cross-Language Orthographic Similarity. Application on Romanian*
Alina Maria Ciobanu and Liviu P. Dinu

14:45–15:10 *Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space*
Arvind Neelakantan, Jeevan Shankar, Alexandre Passos and Andrew McCallum

Session 5c: Information Retrieval and Question Answering

13:30–13:55 *Tailor knowledge graph for query understanding: linking intent topics by propagation*
Shi Zhao and Yan Zhang

13:55–14:20 *Queries as a Source of Lexicalized Commonsense Knowledge*
Marius Pasca

14:20–14:45 *Question Answering over Linked Data Using First-order Logic*
Shizhu He, Kang Liu, Yuanzhe Zhang, Liheng Xu and Jun Zhao

14:45–15:10 *Knowledge Graph and Corpus Driven Segmentation and Answer Inference for Telegraphic Entity-seeking Queries*
Mandar Joshi, Uma Sawant and Soumen Chakrabarti

Session 5p: NLP for the Web, Social Media and Sentiment Analysis

13:30–15:10 *Poster Session*
Multiple presenters

A Regularized Competition Model for Question Difficulty Estimation in Community Question Answering Services
Quan Wang, Jing Liu, Bin Wang and Li Guo

Vote Prediction on Comments in Social Polls
Isaac Persing and Vincent Ng

Exploiting Social Relations and Sentiment for Stock Prediction
Jianfeng Si, Arjun Mukherjee, Bing Liu, Sinno Jialin Pan, Qing Li and Huayi Li

Developing Age and Gender Predictive Lexica over Social Media
Maarten Sap, Gregory Park, Johannes Eichstaedt, Margaret Kern, David Stillwell, Michal Kosinski, Lyle Ungar and Hansen Andrew Schwartz

Dependency Parsing for Weibo: An Efficient Probabilistic Logic Programming Approach
William Yang Wang, Lingpeng Kong, Kathryn Mazaitis and William W Cohen

Exploiting Community Emotion for Microblog Event Detection
Gaoyan Ou, Wei Chen, Tengjiao Wang, Zhongyu Wei, Binyang LI, Dongqing Yang and Kam-Fai Wong

Detecting Disagreement in Conversations using Pseudo-Monologic Rhetorical Structure
Kelsey Allen, Giuseppe Carenini and Raymond Ng

+/-EffectWordNet: Sense-level Lexicon Acquisition for Opinion Inference
Yoonjung Choi and Janyce Wiebe

A Sentiment-aligned Topic Model for Product Aspect Rating Prediction
Hao Wang and Martin Ester

Learning Emotion Indicators from Tweets: Hashtags, Hashtag Patterns, and Phrases
Ashequl Qadir and Ellen Riloff

Fine-Grained Contextual Predictions for Hard Sentiment Words
Sebastian Ebert and Hinrich Schütze

An Iterative Link-based Method for Parallel Web Page Mining
Le Liu, Yu Hong, Jun Lu, Jun Lang, Heng Ji and Jianmin Yao

Exploiting Social Network Structure for Person-to-Person Sentiment Analysis
Robert West, Hristo Paskov, Jure Leskovec, Christopher Potts

15:10–15:40 ***Coffee Break***

Session 6a: Machine Translation

15:40–16:05 *Human Effort and Machine Learnability in Computer Aided Translation*
Spence Green, Sida I. Wang, Jason Chuang, Jeffrey Heer, Sebastian Schuster and Christopher D. Manning

16:05–16:30 *Exact Decoding for Phrase-Based Statistical Machine Translation*
Wilker Aziz, Marc Dymetman and Lucia Specia

16:30–16:55 *Large-scale Expected BLEU Training of Phrase-based Reordering Models*
Michael Auli, Michel Galley and Jianfeng Gao

16:55–17:20 *Confidence-based Rewriting of Machine Translation Output*
Benjamin Marie and Aurélien Max

Session 6b: Semantic Parsing

15:40–16:05 *Learning Compact Lexicons for CCG Semantic Parsing*
Yoav Artzi, Dipanjan Das and Slav Petrov

16:05–16:30 *Morpho-syntactic Lexical Generalization for CCG Semantic Parsing*
Adrienne Wang, Tom Kwiatkowski and Luke Zettlemoyer

16:30–16:55 *Semantic Parsing Using Content and Context: A Case Study from Requirements Elicitation*
Reut Tsarfaty, Ilia Pogrebezky, Guy Weiss, Yaarit Natan, Smadar Szekely and David Harel

16:55–17:20 *Semantic Parsing with Relaxed Hybrid Trees*
Wei Lu

Monday, October 27, 2014 (continued)

Session 6c: NLP-Related Machine Learning

15:40–16:05 *Low-dimensional Embeddings for Interpretable Anchor-based Topic Inference*
David Mimno and Moontae Lee

16:05–16:30 *Weakly-Supervised Learning with Cost-Augmented Contrastive Estimation*
Kevin Gimpel and Mohit Bansal

16:30–16:55 *Don't Until the Final Verb Wait: Reinforcement Learning for Simultaneous Machine Translation*
Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan and Hal Daumé III

16:55–17:20 *PCFG Induction for Unsupervised Parsing and Language Modelling*
James Scicluna and Colin de la Higuera

Session 6p: Computational Psycholinguistics, Text Mining and NLP Applications

15:40–17:20 *Poster Session*
Multiple presenters

Can characters reveal your native language? A language-independent approach to native language identification
Radu Tudor Ionescu, Marius Popescu and Aoife Cahill

Formalizing Word Sampling for Vocabulary Prediction as Graph-based Active Learning
Yo Ehara, Yusuke Miyao, Hidekazu Oiwa, Issei Sato and Hiroshi Nakagawa

Language Transfer Hypotheses with Linear SVM Weights
Shervin Malmasi and Mark Dras

Predicting Dialect Variation in Immigrant Contexts Using Light Verb Constructions
A. Seza Dogruoz and Preslav Nakov

Device-Dependent Readability for Improved Text Understanding
A-Yeong Kim, Hyun-Je Song, Seong-Bae Park and Sang-Jo Lee

Predicting Chinese Abbreviations with Minimum Semantic Unit and Global Constraints
Longkai Zhang, li li, Houfeng WANG and Xu Sun

Tuesday, October 28, 2014

08:00–17:00 ***Registration***

08:00–09:00 ***Refreshments***

Plenary Session

09:00–09:05 *Best Paper Award*
Bo Pang and Walter Daelemans

09:05–09:30 *Language Modeling with Power Low Rank Ensembles*
Ankur P. Parikh, Avneesh Saluja, Chris Dyer and Eric Xing

09:30–09:55 *Modeling Biological Processes for Reading Comprehension*
Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang and Christopher D. Manning

10:00–10:30 ***Coffee Break***

Session 7a: Semantics

10:30–10:55 *Sensicon: An Automatically Constructed Sensorial Lexicon*
Serra Sinem Tekiroglu, Gözde Özbal and Carlo Strapparava

10:55–11:20 *Word Semantic Representations using Bayesian Probabilistic Tensor Factorization*
Jingwei Zhang, Jeremy Salwen, Michael Glass and Alfio Gliozzo

11:20–11:45 *Glove: Global Vectors for Word Representation*
Jeffrey Pennington, Richard Socher and Christopher Manning

11:45–12:10 *Jointly Learning Word Representations and Composition Functions Using Predicate-Argument Structures*
Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa and Yoshimasa Tsuruoka

Session 7b: Information Extraction

10:30–10:55 *Combining Distant and Partial Supervision for Relation Extraction*
Gabor Angeli, Julie Tibshirani, Jean Wu and Christopher D. Manning

10:55–11:20 *Typed Tensor Decomposition of Knowledge Bases for Relation Extraction*
Kai-Wei Chang, Wen-tau Yih, Bishan Yang and Christopher Meek

11:20–11:45 *A convex relaxation for weakly supervised relation extraction*
Edouard Grave

11:45–12:10 *Knowledge Graph and Text Jointly Embedding*
Zhen Wang, Jianwen Zhang, Jianlin Feng and Zheng Chen

Session 7c: Sentiment Analysis and NLP Applications

10:30–10:55 *Abstractive Summarization of Product Reviews Using Discourse Structure*
Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T. Ng and Bita Nejat

10:55–11:20 *Clustering Aspect-related Phrases by Leveraging Sentiment Distribution Consistency*
Li Zhao, Minlie Huang, Haiqiang Chen, Junjun Cheng and Xiaoyan Zhu

11:20–11:45 *Automatic Generation of Related Work Sections in Scientific Papers: An Optimization Approach*
Yue Hu and Xiaojun Wan

11:45–12:10 *Fast and Accurate Misspelling Correction in Large Corpora*
Octavian Popescu and Ngoc Phuoc An Vo

Session 7p: Machine Translation and Machine Learning

10:30–12:10 *Poster Session*
Multiple presenters

Assessing the Impact of Translation Errors on Machine Translation Quality with Mixed-effects Models
Marcello Federico, Matteo Negri, Luisa Bentivogli and Marco Turchi

Refining Word Segmentation Using a Manually Aligned Corpus for Statistical Machine Translation
Xiaolin Wang, Masao Utiyama, Andrew Finch and Eiichiro Sumita

Improving Pivot-Based Statistical Machine Translation by Pivoting the Co-occurrence Count of Phrase Pairs
Xiaoning Zhu, Zhongjun He, Hua Wu, Conghui Zhu, Haifeng Wang and Tiejun Zhao

Word Translation Prediction for Morphologically Rich Languages with Bilingual Neural Networks
Ke M. Tran, Arianna Bisazza and Christof Monz

Dependency-Based Bilingual Language Models for Reordering in Statistical Machine Translation
Ekaterina Garmash and Christof Monz

Combining String and Context Similarity for Bilingual Term Alignment from Comparable Corpora
Georgios Kontonatsios, Ioannis Korkontzelos, Jun'ichi Tsujii and Sophia Ananiadou

Random Manhattan Integer Indexing: Incremental L1 Normed Vector Space Construction
Behrang Q. Zadeh and Siegfried Handschuh

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation
Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk and Yoshua Bengio

Type-based MCMC for Sampling Tree Fragments from Forests
Xiaochang Peng and Daniel Gildea

Convolutional Neural Networks for Sentence Classification
Yoon Kim

Sometimes Average is Best: The Importance of Averaging for Prediction using MCMC Inference in Topic Modeling
Viet-An Nguyen, Jordan Boyd-Graber and Philip Resnik

Large-scale Reordering Model for Statistical Machine Translation using Dual Multinomial Logistic Regression
Abdullah Alrajeh and Mahesan Niranjan

Dynamic Language Models for Streaming Text
Dani Yogatama, Chong Wang, Bryan Routledge, Noah A. Smith, Eric P. Xing

Improved Decipherment of Homophonic Ciphers
Malte Nuhn, Julian Schamper and Hermann Ney

Cipher Type Detection
Malte Nuhn and Kevin Knight

12:10–13:30 ***Lunch Break***

Session 8sa: Segmentation and Tagging / Spoken Language / Semantics

13:30–13:50 *Joint Learning of Chinese Words, Terms and Keywords*
Ziqiang Cao, Sujian Li and Heng Ji

13:50–14:10 *Cross-Lingual Part-of-Speech Tagging through Ambiguous Learning*
Guillaume Wisniewski, Nicolas Pécheux, Souhir Gahbiche-Braham and François Yvon

14:10–14:30 *Comparing Representations of Semantic Roles for String-To-Tree Decoding*
Marzieh Bazrafshan and Daniel Gildea

14:30–14:50 *Detecting Non-compositional MWE Components using Wiktionary*
Bahar Salehi, Paul Cook and Timothy Baldwin

14:50–15:10 *(Empty slot)*
(No presentation)

Session 8sb: Sentiment Analysis / Social / Computational Psycholinguistics / Text Classification

13:30–13:50 *Joint Emotion Analysis via Multi-task Gaussian Processes*
Daniel Beck, Trevor Cohn and Lucia Specia

13:50–14:10 *Detecting Latent Ideology in Expert Text: Evidence From Academic Papers in Economics*
Zubin Jelveh, Bruce Kogut and Suresh Naidu

14:10–14:30 *A Model of Individual Differences in Gaze Control During Reading*
Niels Landwehr, Sebastian Arzt, Tobias Scheffer and Reinhold Kliegl

14:30–14:50 *Muli-label Text Categorization with Hidden Components*
li li, Longkai Zhang and Houfeng WANG

14:50–15:10 *#TagSpace: Semantic Embeddings from Hashtags*
Jason Weston, Sumit Chopra and Keith Adams

Session 8sc: Summarization / Machine Translation / Information Extraction

13:30–13:50 *Joint Decoding of Tree Transduction Models for Sentence Compression*
Jin-ge Yao, Xiaojun Wan and Jianguo Xiao

13:50–14:10 *Dependency-based Discourse Parser for Single-Document Summarization*
Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao and Masaaki Nagata

14:10–14:30 *Improving Word Alignment using Word Similarity*
Theerawat Songyot and David Chiang

14:30–14:50 *Constructing Information Networks Using One Single Model*
Qi Li, Heng Ji, Yu HONG and Sujian Li

14:50–15:10 *Event Role Extraction using Domain-Relevant Word Representations*
Emanuela Boros, Romaric Besançon, Olivier Ferret and Brigitte Grau

Session 8p: Information Extraction

13:30–15:10 *Poster Session*
Multiple presenters

Modeling Joint Entity and Relation Extraction with Table Representation
Makoto Miwa and Yutaka Sasaki

ZORE: A Syntax-based System for Chinese Open Relation Extraction
Likun Qiu and Yue Zhang

Coarse-grained Candidate Generation and Fine-grained Re-ranking for Chinese Abbreviation Prediction
Longkai Zhang, Houfeng WANG and Xu Sun

Type-Aware Distantly Supervised Relation Extraction with Linked Arguments
Mitchell Koch, John Gilmer, Stephen Soderland and Daniel S. Weld

Automatic Inference of the Tense of Chinese Events Using Implicit Linguistic Information
Yuchen Zhang and Nianwen Xue

Joint Inference for Knowledge Base Population
Liwei Chen, Yansong Feng, Jinghui Mo, Songfang Huang and Dongyan Zhao

Combining Visual and Textual Features for Information Extraction from Online Flyers
Emilia Apostolova and Noriko Tomuro

CTPs: Contextual Temporal Profiles for Time Scoping Facts using State Change Detection
Derry Tanti Wijaya, Ndapandula Nakashole and Tom M. Mitchell

Noisy Or-based model for Relation Extraction using Distant Supervision
Ajay Nagesh, Gholamreza Haffari and Ganesh Ramakrishnan

15:10–15:40 ***Coffee Break***

Tuesday, October 28, 2014 (continued)

Session 9a: Machine Learning and Machine Translation

15:40–16:05 *Search-Aware Tuning for Machine Translation*
Lemao Liu and Liang Huang

16:05–16:30 *Latent-Variable Synchronous CFGs for Hierarchical Translation*
Avneesh Saluja, Chris Dyer and Shay B. Cohen

16:30–16:55 *Dynamically Shaping the Reordering Search Space of Phrase-Based Statistical Machine Translation*
Arianna Bisazza and Marcello Federico

16:55–17:20 *(Empty slot)*
(No presentation)

Session 9b: NLP for the Web and Social Media

15:40–16:05 *Gender and Power: How Gender and Gender Environment Affect Manifestations of Power*
Vinodkumar Prabhakaran, Emily E. Reid and Owen Rambow

16:05–16:30 *Online topic model for Twitter considering dynamics of user interests and topic trends*
Kentaro Sasaki, Tomohiro Yoshikawa and Takeshi Furuhashi

16:30–16:55 *Self-disclosure topic model for classifying and analyzing Twitter conversations*
JinYeong Bak, Chin-Yew Lin and Alice Oh

16:55–17:20 *Major Life Event Extraction from Twitter based on Congratulations/Condolences Speech Acts*
Jiwei Li, Alan Ritter, Claire Cardie and Eduard Hovy

Tuesday, October 28, 2014 (continued)

Session 9c: Semantics

15:40–16:05 *Brighter than Gold: Figurative Language in User Generated Comparisons*
Vlad Niculae and Cristian Danescu-Niculescu-Mizil

16:05–16:30 *Classifying Idiomatic and Literal Expressions Using Topic Models and Intensity of Emotions*
Jing Peng, Anna Feldman and Ekaterina Vylomova

16:30–16:55 *TREETALK: Composition and Compression of Trees for Image Descriptions*
Polina Kuznetsova, Vicente Ordonez, Tamara Berg, Yejin Choi

16:55–17:20 *Learning Spatial Knowledge for Text to 3D Scene Generation*
Angel Chang, Manolis Savva and Christopher D. Manning

Session 9p: Discourse, Dialogue and Pragmatics

15:40–17:20 *Poster Session*
Multiple presenters

A Model of Coherence Based on Distributed Sentence Representation
Jiwei Li and Eduard Hovy

Discriminative Reranking of Discourse Parses Using Tree Kernels
Shafiq Joty and Alessandro Moschitti

Recursive Deep Models for Discourse Parsing
Jiwei Li, Rumeng Li and Eduard Hovy

Recall Error Analysis for Coreference Resolution
Sebastian Martschat and Michael Strube

A Rule-Based System for Unrestricted Bridging Resolution: Recognizing Bridging Anaphora and Finding Links to Antecedents
Yufang Hou, Katja Markert and Michael Strube

Resolving Referring Expressions in Conversational Dialogs for Natural User Interfaces
Asli Celikyilmaz, Zhaleh Feizollahi, Dilek Hakkani-Tur and Ruhi Sarikaya

Building Chinese Discourse Corpus with Connective-driven Dependency Tree Structure
Yancui Li, wenhe feng, jing sun, Fang Kong and Guodong Zhou

Prune-and-Score: Learning for Greedy Coreference Resolution
Chao Ma, Janardhan Rao Doppa, J. Walker Orr, Prashanth Mannem, Xiaoli Fern, Tom Dietterich and Prasad Tadepalli

Summarizing Online Forum Discussions – Can Dialog Acts of Individual Messages Help?
Sumit Bhatia, Prakhar Biyani and Prasenjit Mitra

Closing Session

17:20–17:40 *Final Thanks and EMNLP 2015 Preview*
Alessandro Moschitti

IBM Cognitive Computing - An NLP Renaissance!

Salim Roukos
Senior Manager of Multi-Lingual NLP and CTO for Translation Technologies
IBM T. J. Watson Research Center
roukos@us.ibm.com

Abstract

Electronically available multi-modal data (primarily text and meta-data) is unprecedented in terms of its volume, variety, velocity, (and veracity). The increased interest and investment in cognitive computing for building systems and solutions that enable and support richer human-machine interactions presents a unique opportunity for novel statistical models for natural language processing.

In this talk, I will describe a journey at IBM during the past three decades in developing novel statistical models for NLP covering statistical parsing, machine translation, and question-answering systems. Along with a discussion of some of the recent successes, I will discuss some difficult challenges that need to be addressed to achieve more effective cognitive systems and applications.

About the Speaker

Salim Roukos is Senior Manager of Multi-Lingual NLP and CTO for Translation Technologies at IBM T. J. Watson Research Center. Dr. Roukos received his B.E. from the American University of Beirut, in 1976, his M.Sc. and Ph.D. from the University of Florida, in 1978 and 1980, respectively. He joined Bolt Beranek and Newman from 1980 through 1989, where he was a Senior Scientist in charge of projects in speech compression, time scale modification, speaker identification, word spotting, and spoken language understanding. He was an Adjunct Professor at Boston University in 1988 before joining IBM in 1989. Dr. Roukos has served as Chair of the IEEE Digital Signal Processing Committee in 1988.

Salim Roukos currently leads a group at IBM T.J. Watson research Center that focuses on various problems using machine learning techniques for natural language processing. The group pioneered many of the statistical methods for NLP from statistical parsing, to natural language understanding, to statistical machine translation and machine translation evaluation metrics (BLEU metric). Roukos has over a 150 publications in the speech and language areas and over two dozen patents. Roukos was the lead of the group which introduced the first commercial statistical language understanding system for conversational telephony systems (IBM ViaVoice Telephony) in 2000 and the first statistical machine translation product for Arabic-English translation in 2003. He has recently lead the effort to create IBM's offering of IBM Real-Time Translation Services (RTTS) a platform for enabling real-time translation applications such as multilingual chat and on-demand document translation.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), page 1,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Modeling Interestingness with Deep Neural Networks

Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, Li Deng
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
{jfgao,ppantel,mgamon,xiaohe,deng}@microsoft.com

Abstract

This paper presents a deep semantic similarity model (DSSM), a special type of deep neural networks designed for text analysis, for recommending *target* documents to be of interest to a user based on a *source* document that she is reading. We observe, identify, and detect naturally occurring signals of *interestingness* in click transitions on the Web between source and target documents, which we collect from commercial Web browser logs. The DSSM is trained on millions of Web transitions, and maps source-target document pairs to feature vectors in a latent space in such a way that the distance between source documents and their corresponding interesting targets in that space is minimized. The effectiveness of the DSSM is demonstrated using two *interestingness* tasks: automatic highlighting and contextual entity search. The results on large-scale, real-world datasets show that the semantics of documents are important for modeling interestingness and that the DSSM leads to significant quality improvement on both tasks, outperforming not only the classic document models that do not use semantics but also state-of-the-art topic models.

1 Introduction

Tasks of predicting what interests a user based on the document she is reading are fundamental to many online recommendation systems. A recent survey is due to Ricci et al. (2011). In this paper, we exploit the use of a deep semantic model for two such *interestingness* tasks in which document semantics play a crucial role: automatic highlighting and contextual entity search.

Automatic Highlighting. In this task we want a recommendation system to automatically discover the entities (e.g., a person, location, organization etc.) that interest a user when reading a document and to highlight the corresponding text spans, referred to as *keywords* afterwards. We show in this study that document semantics are among the most important factors that influence what is perceived as interesting to the user. For example, we observe in Web browsing logs that when a user reads an article about a movie, she is more likely to browse to an article about an actor or character than to another movie or the director.

Contextual entity search. After identifying the keywords that represent the entities of interest to the user, we also want the system to recommend new, interesting documents by searching the Web for supplementary information about these entities. The task is challenging because the same keywords often refer to different entities, and interesting supplementary information to the highlighted entity is highly sensitive to the semantic context. For example, "Paul Simon" can refer to many people, such as the singer and the senator. Consider an article about the music of Paul Simon and another about his life. Related content about his upcoming concert tour is much more interesting in the first context, while an article about his family is more interesting in the second.

At the heart of these two tasks is the notion of interestingness. In this paper, we model and make use of this notion of interestingness with a deep semantic similarity model (DSSM). The model, extending from the deep neural networks shown recently to be highly effective for speech recognition (Hinton et al., 2012; Deng et al., 2013) and computer vision (Krizhevsky et al., 2012; Markoff, 2014), is *semantic* because it maps documents to feature vectors in a latent semantic space, also known as semantic representations. The model is *deep* because it employs a neural network with several hidden layers including a special convolutional-pooling structure to identify keywords and extract hidden semantic features at different levels of abstractions, layer by layer. The semantic representation is computed through a deep neural network after its training by back-propagation with respect to an objective tailored

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2–13,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

to the respective interestingness tasks. We obtain naturally occurring "interest" signals by observing Web browser transitions, from a source document to a target document, in Web usage logs of a commercial browser. Our training data is sampled from these transitions.

The use of the DSSM to model interestingness is motivated by the recent success of applying related deep neural networks to computer vision (Krizhevshy et al. 2012; Markoff, 2014), speech recognition (Hinton et al. 2012), text processing (Collobert et al. 2011), and Web search (Huang et al. 2013). Among them, (Huang et al. 2013) is most relevant to our work. They also use a deep neural network to map documents to feature vectors in a latent semantic space. However, their model is designed to represent the *relevance* between queries and documents, which differs from the notion of interestingness between documents studied in this paper. It is often the case that a user is interested in a document because it provides supplementary information about the entities or concepts she encounters when reading another document although the overall contents of the second documents is not highly relevant. For example, a user may be interested in knowing more about the history of University of Washington after reading the news about President Obama's visit to Seattle. To better model interestingness, we extend the model of Huang et al. (2013) in two significant aspects. First, while Huang et al. treat a document as a bag of words for semantic mapping, the DSSM treats a document as a sequence of words and tries to discover prominent keywords. These keywords represent the entities or concepts that might interest users, via the convolutional and max-pooling layers which are related to the deep models used for computer vision (Krizhevsky et al., 2013) and speech recognition (Deng et al., 2013a) but are not used in Huang et al.'s model. The DSSM then forms the high-level semantic representation of the whole document based on these keywords. Second, instead of directly computing the document relevance score using cosine similarity in the learned semantic space, as in Huang et al. (2013), we feed the features derived from the semantic representations of documents to a ranker which is trained in a supervised manner. As a result, a document that is not highly relevant to another document a user is reading (i.e., the distance between their derived feature vectors is big) may still have a high score of interestingness because the former provides useful information about an entity mentioned in the latter. Such information and entity are encoded, respectively, by (some subsets of) the semantic features in their corresponding documents. In Sections 4 and 5, we empirically demonstrate that the aforementioned two extensions lead to significant quality improvements for the two interestingness tasks presented in this paper.

Before giving a formal description of the DSSM in Section 3, we formally define the *interestingness* function, and then introduce our data set of naturally occurring interest signals.

2 The Notion of Interestingness

Let D be the set of all documents. Following Gamon et al. (2013), we formally define the *interestingness* modeling task as learning the mapping function:

$$\sigma: D \times D \rightarrow \mathbb{R}^{+}$$

where the function $\sigma(s,t)$ is the quantified degree of interest that the user has in the target document $t \in D$ after or while reading the source document $s \in D$.

Our notion of a document is meant in its most general form as a string of raw unstructured text. That is, the interestingness function should not rely on any document structure such as title tags, hyperlinks, etc., or Web interaction data. In our tasks, documents can be formed either from the plain text of a webpage or as a text span in that plain text, as will be discussed in Sections 4 and 5.

2.1 Data

We can observe many naturally occurring manifestations of interestingness on the Web. For example, on Twitter, users follow shared links embedded in tweets. Arguably the most frequent signal, however, occurs in Web browsing events where users click from one webpage to another via hyperlinks. When a user clicks on a hyperlink, it is reasonable to assume that she is interested in learning more about the anchor, modulo cases of erroneous clicks. Aggregate clicks can therefore serve as a proxy for interestingness. That is, for a given source document, target documents that attract the most clicks are more interesting than documents that attract fewer clicks[1].

[1] We stress here that, although the click signal is available to form a dataset and a gold standard ranker (to be described in Section 4), our task is to model interestingness between unstructured documents, i.e., without access to any document structure or Web interaction data. Thus, in our experiments,

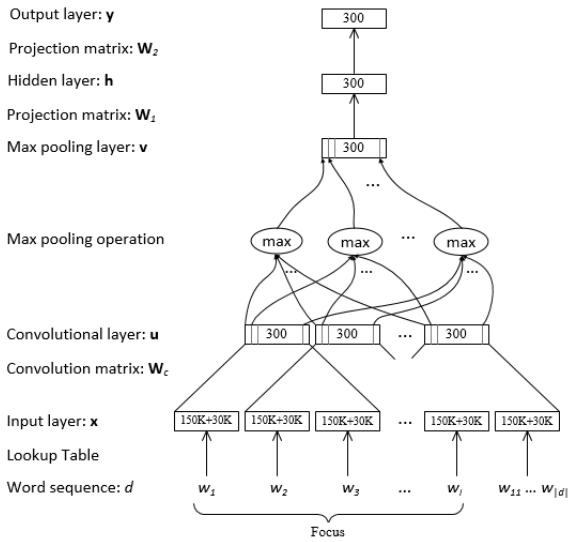


Figure 1: Illustration of the network architecture and information flow of the DSSM

We collect a large dataset of user browsing events from a commercial Web browser. Specifically, we sample 18 million occurrences of a user click from one Wikipedia page to another during a one year period. We restrict our browsing events to Wikipedia since its pages tend to contain many anchors (79 on average, where on average 42 have a unique target URL). Thus, they attract enough traffic for us to obtain robust browsing transition data[2]. We group together all transitions originating from the same page and randomly hold out 20% of the transitions for our evaluation data (**EVAL**), 20% for training the DSSM described in Section 3.2 (**TRAIN_1**), and the remaining 60% for training our task specific rankers described in Section 3.3 (**TRAIN_2**). In our experiments, we used different settings for the two interestingness tasks. Thus, we postpone the detailed description of these datasets and other task-specific datasets to Sections 4 and 5.

3 A Deep Semantic Similarity Model (DSSM)

This section presents the architecture of the DSSM, describes the parameter estimation, and the way the DSSM is used in our tasks.

3.1 Network Architecture

The heart of the DSSM is a deep neural network with convolutional structure, as shown in Figure 1. In what follows, we use lower-case bold letters, such as $\mathbf{x}$, to denote column vectors, $x(i)$ to denote the i^{th} element of $\mathbf{x}$, and upper-case letters, such as $\mathbf{W}$, to denote matrices.

Input Layer x. It takes two steps to convert a document d, which is a sequence of words, into a vector representation $\mathbf{x}$ for the input layer of the network: (1) convert each word in d to a word vector, and (2) build $\mathbf{x}$ by concatenating these word vectors. To convert a word w into a word vector, we first represent w by a one-hot vector using a vocabulary that contains N high frequent words ($N = 150$K in this study). Then, following Huang et al. (2013), we map w to a separate tri-letter vector. Consider the word "#dog#", where # is a word boundary symbol. The nonzero elements in its tri-letter vector are "#do", "dog", and "og#". We then form the word vector of w by concatenating its one-hot vector and its tri-letter vector. It is worth noting that the tri-letter vector complements the one-hot vector representation in two aspects. First, different OOV (out of vocabulary) words can be represented by tri-letter vectors with few collisions. Second, spelling variations of the same word can be mapped to the points that are close to each other in the tri-letter space. Although the number of unique English words on the Web is extremely large, the total number of distinct tri-letters in English is limited (restricted to the most frequent 30K in this study). As a result, incorporating tri-letter vectors substantially improves the representation power of word vectors while keeping their size small.

To form our input layer $\mathbf{x}$ using word vectors, we first identify a text span with a high degree of relevance, called *focus*, in d using task-specific heuristics (see Sections 4 and 5 respectively). Second, we form $\mathbf{x}$ by concatenating each word vector in the focus and a vector that is the summation of all other word vectors, as shown in Figure 1. Since the length of the focus is much smaller than that of its document, $\mathbf{x}$ is able to capture the contextual information (for the words in the focus)

we remove all structural information (e.g., hyperlinks and XML tags) in our documents, except that in the highlighting experiments (Section 4) we use anchor texts to simulate the candidate keywords to be highlighted. We then convert each Web document into plain text, which is white-space tokenized and lowercased. Numbers are retained and no stemming is performed.

[2] We utilize the May 3, 2013 English Wikipedia dump consisting of roughly 4.1 million articles from http://dumps.wikimedia.org.

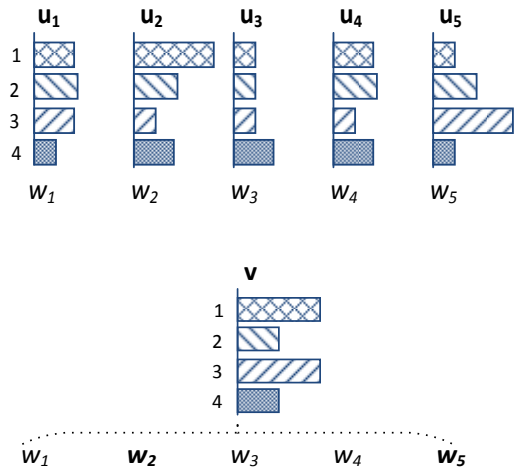


Figure 2: Toy example of (upper) a 5-word document and its local feature vectors extracted using a convolutional layer, and (bottom) the global feature vector of the document generated after max-pooling.

useful to the corresponding tasks, with a manageable vector size.

Convolutional Layer u. A convolutional layer extracts local features around each word w_i in a word sequence of length I as follows. We first generate a contextual vector $\mathbf{c}_i$ by concatenating the word vectors of w_i and its surrounding words defined by a window (the window size is set to 3 in this paper). Then, we generate for each word a local feature vector $\mathbf{u}_i$ using a tanh activation function and a linear projection matrix $\mathbf{W}_c$, which is the same across all windows i in the word sequence, as:

$$\mathbf{u}_i = \tanh\left(\mathbf{W}_c^{\mathrm{T}}\mathbf{c}_i\right), \text{where } i = 1 \dots I \quad (1)$$

Max-pooling Layer v. The size of the output $\mathbf{u}$ depends on the number of words in the word sequence. Local feature vectors have to be combined to obtain a global feature vector, with a fixed size independent of the document length, in order to apply subsequent standard affine layers. We design $\mathbf{v}$ by adopting the max operation over each "time" i of the sequence of vectors computed by (1), which forces the network to retain only the most useful, partially invariant local features produced by the convolutional layer:

$$v(j) = \max_{i=1,\dots,I}\{\mathrm{u}_i(j)\} \quad (2)$$

where the max operation is performed for each dimension of $\mathbf{u}$ across $i = 1, \dots, I$ respectively.

That convolutional and max-pooling layers are able to discover prominent keywords of a document can be demonstrated using the procedure in Figure 2 using a toy example. First, the convolutional layer of (1) generates for each word in a 5-word document a 4-dimensional local feature vector, which represents a distribution of four *topics*. For example, the most prominent topic of w_2 within its three word context window is the first topic, denoted by $u_2(1)$, and the most prominent topic of w_5 is $u_5(3)$. Second, we use max-pooling of (2) to form a global feature vector, which represents the topic distribution of the whole document. We see that $v(1)$ and $v(3)$ are two prominent topics. Then, for each prominent topic, we trace back to the local feature vector that survives max-pooling:

$$v(1) = \max_{i=1,\dots,5}\{u_i(1)\} = u_2(1)$$

$$v(3) = \max_{i=1,\dots,5}\{u_i(3)\} = u_5(3).$$

Finally, we label the corresponding words of these local feature vectors, w_2 and w_5, as keywords of the document.

Figure 3 presents a sample of document snippets and their keywords detected by the DSSM according to the procedure elaborated in Figure 2. It is interesting to see that many names are identified as keywords although the DSSM is not designed explicitly for named entity recognition.

Fully-Connected Layers h and y. The fixed sized global feature vector $\mathbf{v}$ of (2) is then fed to several standard affine network layers, which are stacked and interleaved with nonlinear activation functions, to extract highly non-linear features $\mathbf{y}$ at the output layer. In our model, shown in Figure 1, we have:

$$\mathbf{h} = \tanh(\mathbf{W}_1^{\mathrm{T}}\mathbf{v}) \quad (3)$$

$$\mathbf{y} = \tanh(\mathbf{W}_2^{\mathrm{T}}\mathbf{h}) \quad (4)$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are learned linear projection matrices.

3.2 Training the DSSM

To optimize the parameters of the DSSM of Figure 1, i.e., $\boldsymbol{\theta} = \{\mathbf{W}_c, \mathbf{W}_1, \mathbf{W}_2\}$, we use a pair-wise rank loss as objective (Yih et al. 2011). Consider a source document s and two candidate target documents t_1 and t_2, where t_1 is more interesting than t_2 to a user when reading s. We construct two pairs of documents (s, t_1) and (s, t_2), where the former is preferred and should have a higher

… the **comedy festival** formerly known as the us **comedy arts** festival is a comedy festival held each year in **las vegas nevada** from its 1985 inception to 2008 . it was held annually at the **wheeler opera house** and other venues in **aspen colorado** . the primary sponsor of the festival was hbo with co-sponsorship by caesars palace . the primary venue tbs **geico insurance** twix candy bars and **smirnoff vodka hbo** exited the festival business in 2007 and tbs became the primary sponsor the festival includes standup comedy performances appearances by the casts of television shows…

… **bad samaritans** is an american **comedy** series produced by **walt becker kelly** hayes and **ross putman** . it premiered on **netflix** on march 31 2013 cast and characters . the show focuses on a community service parole group and their parole officer **brian kubach** as **jake gibson** an aspiring professional **starcraft** player who gets sentenced to 2000 hours of community service for starting a **forest fire** during his **breakup** with **drew** prior to community service he had no real ambition in life other than to be a professional gamer and become wealthy overnight like **mark zuckerberg** as in life his goal during …

Figure 3: A sample of document snippets and the keywords (in **bold**) detected by the DSSM.

interestingness score. Let Δ be the difference of their interestingness scores: $\Delta = \sigma(s, t_1) - \sigma(s, t_2)$, where σ is the interestingness score, computed as the cosine similarity:

$$\sigma(s,t) \equiv \text{sim}_{\boldsymbol{\theta}}(s,t) = \frac{\mathbf{y}_s^{\mathrm{T}}\mathbf{y}_t}{\|\mathbf{y}_s\|\|\mathbf{y}_t\|} \qquad (5)$$

where $\mathbf{y}_s$ and $\mathbf{y}_t$ are the feature vectors of s and t, respectively, which are generated using the DSSM, parameterized by $\boldsymbol{\theta}$. Intuitively, we want to learn $\boldsymbol{\theta}$ to maximize Δ. That is, the DSSM is learned to represent documents as points in a hidden interestingness space, where the similarity between a document and its interesting documents is maximized.

We use the following logistic loss over Δ, which can be shown to upper bound the pairwise accuracy:

$$\mathcal{L}(\Delta; \boldsymbol{\theta}) = \log(1 + \exp(-\gamma\Delta)) \qquad (6)$$

The loss function in (6) has a shape similar to the hinge loss used in SVMs. Because of the use of the cosine similarity function, we add a scaling factor γ that magnifies Δ from [-2, 2] to a larger range. Empirically, the value of γ makes no difference as long as it is large enough. In the experiments, we set $\gamma = 10$. Because the loss function is differentiable, optimizing the model parameters can be done using gradient-based methods. Due to space limitations, we omit the derivation of the gradient of the loss function, for which readers are referred to related derivations (e.g., Collobert et al. 2011; Huang et al. 2013; Shen et al. 2014).

In our experiments we trained DSSMs using mini-batch Stochastic Gradient Descent. Each mini-batch consists of 256 source-target document pairs. For each source document s, we randomly select from that batch four target documents which are not paired with s as negative training samples[3]. The DSSM trainer is implemented using a GPU-accelerated linear algebra library, which is developed on CUDA 5.5. Given the training set (**TRAIN_1** in Section 2), it takes approximately 30 hours to train a DSSM as shown in Figure 1, on a Xeon E5-2670 2.60GHz machine with one Tesla K20 GPU card.

In principle, the loss function of (6) can be further regularized (e.g. by adding a term of $L2$ norm) to deal with overfitting. However, we did not find a clear empirical advantage over the simpler early stop approach in a pilot study, hence we adopted the latter in the experiments in this paper. Our approach adjusts the learning rate η during the course of model training. Starting with $\eta = 1.0$, after each epoch (a pass over the entire training data), the learning rate is adjusted as $\eta = 0.5 \times \eta$ if the loss on validation data (held-out from **TRAIN_1**) is not reduced. The training stops if either η is smaller than a preset threshold (0.0001) or the loss on training data can no longer be reduced significantly. In our experiments, the DSSM training typically converges within 20 epochs.

3.3 Using the DSSM

We experiment with two ways of using the DSSM for the two interestingness tasks. First, we use the DSSM as a feature generator. The output layer of the DSSM can be seen as a set of semantic features, which can be incorporated in a boosted tree

[3] In our experiments, we observed better results by sampling more negative training examples (e.g., up to 100) although this makes the training much slower. An alternative approach is to approximate the partition function using Noise Contrastive Estimation (Gutmann and Hyvarinen 2010). We leave it to future work.

	#	Models	HEAD			TORSO			TAIL		
			@1	@5	@10	@1	@5	@10	@1	@5	@10
src only	1	**RAND**	0.041	0.062	0.081	0.036	0.076	0.109	0.062	0.195	0.258
	2	**1stK**	0.010	0.177	0.243	0.072	0.171	0.240	0.091	0.274	0.348
	3	**LastK**	0.170	0.022	0.027	0.022	0.044	0.062	0.058	0.166	0.219
	4	**NSF**	0.215	0.253	0.295	0.139	0.229	0.282	0.109	0.293	0.365
	5	**NSF+WCAT**	0.438	0.424	0.463	0.194	0.290	0.346	0.118	0.317	0.386
	6	**NSF+JTT**	0.220	0.302	0.343	0.141	0.241	0.295	0.111	0.300	0.369
	7	**NSF+DSSM_BOW**	0.312	0.351	0.391	0.162	0.258	0.313	0.110	0.299	0.372
	8	**NSF+DSSM**	0.362	0.386	0.421	0.178	0.275	0.330	0.116	0.312	0.382
src+tar	9	**NSF+WCAT**	0.505	0.475	0.501	0.224	0.304	0.356	0.129	0.324	0.391
	10	**NSF+JTT**	0.345	0.380	0.418	0.183	0.280	0.332	0.131	0.321	0.390
	11	**NSF+DSSM_BOW**	0.416	0.393	0.428	0.197	0.274	0.325	0.123	0.311	0.380
	12	**NSF+DSSM**	**0.554**	**0.524**	**0.547**	**0.241**	**0.317**	**0.367**	**0.135**	**0.329**	**0.398**

Table 1: Highlighting task performance (NDCG @ K) of interest models over HEAD, TORSO and TAIL test sets. Bold indicates statistical significance over all non-shaded results using *t*-test ($p = 0.05$).

based ranker (Friedman 1999) trained discriminatively on the task-specific data. Given a source-target document pair (s, t), the DSSM generates 600 features (300 from the output layers $\mathbf{y}_s$ and $\mathbf{y}_t$ for each s and t, respectively).

Second, we use the DSSM as a direct implementation of the interestingness function σ. Recall from Section 3.2 that in model training, we measure the interestingness score for a document pair using the cosine similarity between their corresponding feature vectors ($\mathbf{y}_s$ and $\mathbf{y}_t$). Similarly at runtime, we define $\sigma = \text{sim}_{\boldsymbol{\theta}}(s, t)$ as (5).

4 Experiments on Highlighting

Recall from Section 1 that in this task, a system must select k most interesting keywords in a document that a user is reading. To evaluate our models using the click transition data described in Section 2, we simulate the task as follows. We use the set of anchors in a source document s to simulate the set of candidate keywords that may be of interest to the user while reading s, and treat the text of a document that is linked by an anchor in s as a target document t. As shown in Figure 1, to apply DSSM to a specific task, we need to define the focus in source and target documents. In this task, the *focus* in *s* is defined as the anchor text, and the *focus* in *t* is defined as the first 10 tokens in *t*.

We evaluate the performance of a highlighting system against a gold standard interestingness function σ' which scores the interestingness of an anchor as the number of user clicks on t from the anchor in s in our data. We consider the ideal selection to then consist of the k most interesting anchors according to σ'. A natural metric for this task is Normalized Discounted Cumulative Gain (NDCG) (Jarvelin and Kekalainen 2000).

We evaluate our models on the **EVAL** dataset described in Section 2. We utilize the transition distributions in **EVAL** to create three other test sets, following the stratified sampling methodology commonly employed in the IR community, for the frequently, less frequently, and rarely viewed source pages, referred to as **HEAD**, **TORSO**, and **TAIL**, respectively. We obtain these sets by first sorting the unique source documents according to their frequency of occurrence in **EVAL**. We then partition the set so that **HEAD** corresponds to all transitions from the source pages at the top of the list that account for 20% of the transitions in **EVAL**; **TAIL** corresponds to the transitions at the bottom also accounting for 20% of the transitions in **EVAL**; and **TORSO** corresponds to the remaining transitions.

4.1 Main Results

Table 1 summarizes the results of various models over the three test sets using NDCG at truncation levels 1, 5, and 10.

Rows 1 to 3 are simple heuristic baselines. **RAND** selects k random anchors, **1stK** selects the first k anchors and **LastK** the last k anchors.

The other models in Table 1 are boosted tree based rankers trained on **TRAIN_2** described in Section 2. They vary only in their features. The ranker in Row 4 uses Non-Semantic Features (**NSF**) only. These features are derived from the

source document *s* and from user session information in the browser log. The document features include: position of the anchor in the document, frequency of the anchor, and anchor density in the paragraph.

The rankers in Rows 5 to 12 use the NSF and the semantic features computed from source and target documents of a browsing transition. We compare semantic features derived from three different sources. The first feature source comes from our DSSMs (**DSSM** and **DSSM_BOW**) using the output layers as feature generators as described in Section 3.3. **DSSM** is the model described in Section 3 and **DSSM_BOW** is the model proposed by Huang et al. (2013) where documents are view as bag of words (BOW) and the convolutional and max-pooling layers are not used. The two other sources of semantic features are used as a point of comparison to the DSSM. One is a generative semantic model (Joint Transition Topic model, or **JTT**) (Gamon et al. 2013). **JTT** is an LDA-style model (Blei et al. 2003) that is trained jointly on source and target documents linked by browsing transitions. **JTT** generates a total of 150 features from its latent variables, 50 each for the source topic model, the target topic model and the transition model. The other semantic model of contrast is a manually defined one, which we use to assess the effectiveness of automatically learned models against human modelers. To this effect, we use the page categories that editors assign in Wikipedia as semantic features (**WCAT**). These features number in the multiple thousands. Using features such as **WCAT** is not a viable solution in general since Wikipedia categories are not available for all documents. As such, we use it solely as a point of comparison against **DSSM** and **JTT**.

We also distinguish between two types of learned rankers: those which draw their features only from the *source* (**src only**) document and those that draw their features from both the *source* and *target* (**src+tar**) documents. Although our task setting allows access to the content of both source and target documents, there are practical scenarios where a system should predict what interests the user without looking at the target document because the extra step of identifying a suitable target document for each candidate concept or entity of interest is computationally expensive.

4.2 Analysis of Results

As shown in Table 1, **NSF+DSSM**, which incorporates our DSSM, is the overall best performing system across test sets. The task is hard as evidenced by the weak baseline scores. One reason is the large average number of candidates per page. On **HEAD**, we found an average of 170 anchors (of which 95 point to a unique target URL). For **TORSO** and **TAIL**, we found the average number of anchors to be 94 (52 unique targets) and 41 (19 unique targets), respectively.

Clearly, the semantics of the documents form important signals for this task: **WCAT**, **JTT**, **DSSM_BOW**, and **DSSM** all significantly boost the performance over NSF alone. There are two interesting comparisons to consider: (a) manual semantics vs. learned semantics; and (b) deep semantic models vs. generative topic models. On (a), we observe somewhat surprisingly that the learned DSSM produces features that outperform the thousands of features coming from manually (editor) assigned Wikipedia category features (**WCAT**), in all but the TAIL where the two perform statistically the same. In contrast, features from the generative model (**JTT**) perform worse than **WCAT** across the board except on **TAIL** where **JTT** and **WCAT** are statistically tied. On (b), we observe that DSSM outperforms a state-of-the-art generative model (**JTT**) on **HEAD** and **TORSO**. On **TAIL**, they are statistically indistinguishable.

We turn now to inspecting the scenario where features are only drawn from the source document (Rows 1-8 in Table 1). Again we observe that semantic features significantly boost the performance against **NSF** alone, however they significantly deteriorate when compared to using features from both source and target documents. In this scenario, the manual semantics from **WCAT** outperform all other models, but with a diminishing effect as we move from **HEAD** through **TORSO** to **TAIL**. **DSSM** is the best performing learned semantic model.

Finally, we present the results to justify the two modifications we made to extend the model of Huang et al. (2013) to the DSSM, as described in Section 1. First, we see in Table 1 that **DSSM_BOW**, which has the same network structure of Huang et al.'s model, is much weaker than **DSSM**, demonstrating the benefits of using convolutional and max-pooling layers to extract semantic features for the highlighting task. Second, we conduct several experiments by using the cosine scores between the output layers of **DSSM** for s and t as features (following the procedure in Section 3.3 for using the DSSM as a direct implementation of σ). We found that adding the cosine

#	Models	@1	@3	AUC
1	**BM25 (entity)**	0.133	0.195	0.583
2	**BM25**	0.142	0.227	0.675
3	**WTM**	0.191	0.287	0.678
4	**BLTM**	0.214	0.306	0.704
5	**DSSM**	0.259*	0.356*	0.711*
6	**DSSM_BOW**	0.223	0.322	0.699
7	**Baseline ranker**	0.283	0.360	0.723
8	**7 + DSSM(1)**	0.301#	0.385#	0.758#
9	**7 + DSSM(600)**	0.327##	0.402##	0.782##

Table 2: Contextual entity search task performance (NDCG @ K and AUC). * indicates statistical significance over all non-shaded single model results (Rows 1 to 6) using *t*-test ($p < 0.05$). # indicates statistical significance over results in Row 7. ## indicates statistical significance over results in Rows 7 and 8.

features to **NSF+DSSM** does not lead to any improvement. We also combined **NSF** with solely the cosine features from **DSSM** (i.e., without the other semantic features drawn from its output layers). But we still found no improvement over using **NSF** alone. Thus, we conclude that for this task it is much more effective to feed the features derived from **DSSM** to a supervised ranker than directly computing the interestingness score using cosine similarity in the learned semantic space, as in Huang et al. (2013).

5 Experiments on Entity Search

We construct the evaluation data set for this second task by randomly sampling a set of documents from a traffic-weighted set of Web documents. In a second step, we identify the entity names in each document using an in-house named entity recognizer. We issue each entity name as a query to a commercial search engine, and retain up to the top-100 retrieved documents as candidate *target* documents. We form for each entity a *source* document which consists of the entity text and its surrounding text defined by a 200-word window. We define the *focus* (as in Figure 1) in s as the entity text, and the *focus* in t as the first 10 tokens in t.

The final evaluation data set contains 10,000 source documents. On average, each source document is associated with 87 target documents. Finally, the source-target document pairs are labeled in terms of interestingness by paid annotators. The label is on a 5-level scale, 0 to 4, with 4 meaning the target document is the most interesting to the source document and 0 meaning the target is of no interest.

We test our models on two scenarios. The first is a ranking scenario where k interesting documents are displayed to the user. Here, we select the top-k ranked documents according to their interestingness scores. We measure the performance via NDCG at truncation levels 1 and 3. The second scenario is to display to the user all interesting results. In this scenario, we select all target documents with an interestingness score exceeding a predefined threshold. We evaluate this scenario using ROC analysis and, specifically, the area under the curve (AUC).

5.1 Main Results

The main results are summarized in Table 2. Rows 1 to 6 are **single model** results, where each model is used as a direct implementation of the interestingness function σ. Rows 7 to 9 are **ranker** results, where σ is defined as a boosted tree based ranker that incorporates different sets of features extracted from source and target documents, including the features derived from single models. As in the highlighting experiments, all the machine-learned single models, including the DSSM, are trained on **TRAIN_1**, and all the rankers are trained on **TRAIN_2**.

5.2 Analysis of Results

BM25 (Rows 1 and 2 in Table 2) is the classic document model (Robertson and Zaragoza 2009). It uses the bag-of-words document representation and the BM25 term weighting function. In our setting, we define the interestingness score of a document pair as the dot product of their BM25-weighted term vectors. To verify the importance of using contextual information, we compare two different ways of forming the term vector of a source document. The first only uses the entity text (Row 1). The second (Row 2) uses both the entity text and and its surrounding text in a 200-word window (i.e., the entire *source* document). Results show that the model using contextual information is significantly better. Therefore, all the other models in this section use both the entity texts and their surrounding text.

WTM (Row 3) is our implementation of the word translation model for IR (Berger and Lafferty 1999; Gao et al. 2010). **WTM** defines the interestingness score as:

$$\sigma(s,t) = \prod_{w_t \in t} \sum_{w_s \in s} P(w_t|w_s) P(w_s|s),$$

where $P(w_s|s)$ is the unigram probability of word w_s in s, and $P(w_t|w_s)$ is the probability of *translating* w_s into w_t, trained on source-target document pairs using EM (Brown et al. 1993). The translation-based approach allows any pair of non-identical but semantically related words to have a nonzero matching score. As a result, it significantly outperforms **BM25**.

BTLM (Row 4) follows the best performing bilingual topic model described in Gao et al. (2011), which is an extension of PLSA (Hofmann 1999). The model is trained on source-target document pairs using the EM algorithm with a constraint enforcing a source document s and its target document t to not only share the same prior topic distribution, but to also have similar fractions of words assigned to each topic. **BLTM** defines the interestingness score between s and t as:

$$\sigma(s,t) = \prod_{w_t \in t} \sum_z P(w_t|\phi_z) P(z|\theta^s).$$

The model assumes the following story of generating t from s. First, for each topic z a word distribution ϕ_z is selected from a Dirichlet prior with concentration parameter β. Second, given s, a topic distribution θ^s is drawn from a Dirichlet prior with parameter α. Finally, t is generated word by word. Each word w_t is generated by first selecting a topic z according to θ^s, and then drawing a word from ϕ_z. We see that **BLTM** models interestingness by taking into account the semantic topic distribution of the entire documents. Our results in Table 2 show that **BLTM** outperforms **WTM** by a significant margin in both NDCG and AUC.

DSSM (Row 5) outperforms all the competing single models, including the state-of-the-art topic model **BLTM**. Now, we inspect the difference between **DSSM** and **BLTM** in detail. Although both models strive to generate the semantic representation of a document, they use different modeling approaches. **BLTM** by nature is a generative model. The semantic representation in **BLTM** is a distribution of hidden semantic topics. Such a distribution is learned using Maximum Likelihood Estimation in an unsupervised manner, i.e., maximizing the log-likelihood of the source-target document pairs in the training data. On the other hand, **DSSM** represents documents as points in a hidden semantic space using a supervised learning method, i.e., paired documents are closer in that latent space than unpaired ones. We believe that the superior performance of **DSSM** is largely due to the fact that the model parameters are discriminatively trained using an objective that is tailored to the interestingness task.

In addition to the difference in training methods, **DSSM** and **BLTM** also use different model structures. **BLTM** treats a document as a bag of words (thus losing some important contextual information such as word order and inter-word dependencies), and generates semantic representations of documents using linear projection. **DSSM**, on the other hand, treats text as a sequence of words and better captures local and global context, and generates highly non-linear semantic features via a deep neural network. To further verify our analysis, we inspect the results of a variant of **DSSM**, denoted as **DSSM_BOW** (Row 6), where the convolution and max-pooling layers are removed. This model treats a document as a bag of words, just like **BLTM**. These results demonstrate that the effectiveness of **DSSM** can also be attributed to the convolutional architecture in the neural network, in addition to being deep and being discriminative.

We turn now to discussing the ranker results in Rows 7 to 9. The baseline ranker (Row 7) uses 158 features, including many counts and single model scores, such as **BM25** and **WMT**. **DSSM** (Row 5) alone is quite effective, being close in performance to the baseline ranker with non-DSSM features. Integrating the DSSM score computed in (5) as one single feature into the ranker (Row 8) leads to a significant improvement over the baseline. The best performing combination (Row 9) is obtained by incorporating the DSSM feature vectors of source and target documents (i.e., 600 features in total) in the ranker.

We thus conclude that on both tasks, automatic highlighting and contextual entity search, features drawn from the output layers of our deep semantic model result in significant gains after being added to a set of non-semantic features, and in comparison to other types of semantic models used in the past.

6 Related Work

In addition to the notion of relevance as described in Section 1, related to interestingness is also the notion of *salience* (also called *aboutness*) (Gamon et al. 2013; 2014; Parajpe 2009; Yih et al. 2006). Salience is the centrality of a term to the content of a document. Although salience and interestingness interact, the two are not the same. For example, in a news article about President Obama's visit to Seattle, Obama is salient, yet the average user would probably not be interested in learning more about Obama while reading that article.

There are many systems that identify popular content in the Web or recommend content (e.g., Bandari et al. 2012; Lerman and Hogg 2010; Szabo and Huberman 2010), which is closely related to the highlighting task. In contrast to these approaches, we strive to predict what term a user is likely to be interested in when reading content, which may or may not be the same as the most popular content that is related to the current document. It has empirically been demonstrated in Gamon et al. (2013) that popularity is in fact a rather poor predictor for interestingness. The task of contextual entity search, which is formulated as an information retrieval problem in this paper, is also related to research on entity resolution (Stefanidis et al. 2013).

Latent Semantic Analysis (Deerwester et al. 1990) is arguably the earliest semantic model designed for IR. Generative topic models widely used for IR include PLSA (Hofmann 1990) and LDA (Blei et al. 2003). Recently, these models have been extended to handle cross-lingual cases, where there are pairs of corresponding documents in different languages (e.g., Dumais et al. 1997; Gao et al. 2011; Platt et al. 2010; Yih et al. 2011).

By exploiting deep architectures, deep learning techniques are able to automatically discover from training data the hidden structures and the associated features at different levels of abstraction useful for a variety of tasks (e.g., Collobert et al. 2011; Hinton et al. 2012; Socher et al. 2012; Krizhevsky et al., 2012; Gao et al. 2014). Hinton and Salakhutdinov (2010) propose the most original approach based on an unsupervised version of the deep neural network to discover the hierarchical semantic structure embedded in queries and documents. Huang et al. (2013) significantly extends the approach so that the deep neural network can be trained on large-scale query-document pairs giving much better performance. The use of the convolutional neural network for text processing, central to our DSSM, was also described in Collobert et al. (2011) and Shen et al. (2014) but with very different applications. The DSSM described in Section 3 can be viewed as a variant of the deep neural network models used in these previous studies.

7 Conclusions

Modeling interestingness is fundamental to many online recommendation systems. We obtain naturally occurring interest signals by observing Web browsing transitions where users click from one webpage to another. We propose to model this "interestingness" with a deep semantic similarity model (DSSM), based on deep neural networks with special convolutional-pooling structure, mapping source-target document pairs to feature vectors in a latent semantic space. We train the DSSM using browsing transitions between documents. Finally, we demonstrate the effectiveness of our model on two interestingness tasks: automatic highlighting and contextual entity search. Our results on large-scale, real-world datasets show that the semantics of documents computed by the DSSM are important for modeling interestingness and that the new model leads to significant improvements on both tasks. DSSM is shown to outperform not only the classic document models that do not use (latent) semantics but also state-of-the-art topic models that do not have the deep and convolutional architecture characterizing the DSSM.

One area of future work is to extend our method to model interestingness given an entire user session, which consists of a sequence of browsing events. We believe that the prior browsing and interaction history recorded in the session provides additional signals for predicting interestingness. To capture such signals, our model needs to be extended to adequately represent time series (e.g., causal relations and consequences of actions). One potentially effective model for such a purpose is based on the architecture of recurrent neural networks (e.g., Mikolov et al. 2010; Chen and Deng, 2014), which can be incorporated into the deep semantic model proposed in this paper.

Additional Authors

Yelong Shen (Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA, email: yeshen@microsoft.com).

Acknowledgments

The authors thank Johnson Apacible, Pradeep Chilakamarri, Edward Guo, Bernhard Kohlmeier, Xiaolong Li, Kevin Powell, Xinying Song and Ye-Yi Wang for their guidance and valuable discussions. We also thank the three anonymous reviewers for their comments.

References

Bandari, R., Asur, S., and Huberman, B. A. 2012. The pulse of news in social media: forecasting popularity. In *ICWSM*.

Bengio, Y., 2009. Learning deep architectures for AI. *Fundamental Trends in Machine Learning*, 2(1):1–127.

Berger, A., and Lafferty, J. 1999. Information retrieval as statistical translation. In *SIGIR*, pp. 222-229.

Blei, D. M., Ng, A. Y., and Jordan, M. J. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3.

Broder, A., Fontoura, M., Josifovski, V., and Riedel, L. 2007. A semantic approach to contextual advertising. In *SIGIR*.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263-311.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, and Hullender, G. 2005. Learning to rank using gradient descent. In *ICML*, pp. 89-96.

Chen, J. and Deng, L. 2014. A primal-dual method for training recurrent neural networks constrained by the echo-state property. In *ICLR*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P., 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, vol. 12.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T., and Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6): 391-407

Deng, L., Hinton, G., and Kingsbury, B. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *ICASSP*.

Deng, L., Abdel-Hamid, O., and Yu, D., 2013a. A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion. In *ICASSP*.

Dumais, S. T., Letsche, T. A., Littman, M. L., and Landauer, T. K. 1997. Automatic cross-linguistic information retrieval using latent semantic indexing. In *AAAI-97 Spring Symposium Series: Cross-Language Text and Speech Retrieval*.

Friedman, J. H. 1999. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29:1189-1232.

Gamon, M., Mukherjee, A., Pantel, P. 2014. Predicting interesting things in text. In *COLING*.

Gamon, M., Yano, T., Song, X., Apacible, J. and Pantel, P. 2013. Identifying salient entities in web pages. In *CIKM*.

Gao, J., He, X., and Nie, J-Y. 2010. Clickthrough-based translation models for web search: from word models to phrase models. In *CIKM*. pp. 1139-1148.

Gao, J., He, X., Yih, W-t., and Deng, L. 2014. Learning continuous phrase representations for translation modeling. In *ACL*.

Gao, J., Toutanova, K., Yih., W-T. 2011. Clickthrough-based latent semantic models for web search. In *SIGIR*. pp. 675-684.

Graves, A., Mohamed, A., and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *ICASSP*.

Gutmann, M. and Hyvarinen, A. 2010. Noise-contrastive estimation: a new estimation principle for unnormalized statistical models. In *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS2010)*.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B., 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29:82-97.

Hinton, G., and Salakhutdinov, R., 2010. Discovering binary codes for documents by learning deep generative models. *Topics in Cognitive Science*, pp. 1-18.

Hofmann, T. 1999. Probabilistic latent semantic indexing. In *SIGIR*. pp. 50-57.

Huang, P., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*.

Jarvelin, K. and Kekalainen, J. 2000. IR evaluation methods for retrieving highly relevant documents. In *SIGIR*. pp. 41-48.

Krizhevsky, A., Sutskever, I. and Hinton, G. 2012. ImageNet classification with deep convolutional neural networks. In *NIPS*.

Lerman, K., and Hogg, T. 2010. Using a model of social dynamics to predict popularity of news. In *WWW*. pp. 621-630.

Markoff, J. 2014. Computer eyesight gets a lot more accurate. In New York Times.

Mikolov, T.. Karafiat, M., Burget, L., Cernocky, J., and Khudanpur, S. 2010. Recurrent neural network based language model. In *INTERSPEECH*. pp. 1045-1048.

Paranjpe, D. 2009. Learning document aboutness from implicit user feedback and document structure. In *CIKM*.

Platt, J., Toutanova, K., and Yih, W. 2010. Translingual document representations from discriminative projections. In *EMNLP*. pp. 251-261.

Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B. (eds) 2011. *Recommender System Handbook*, Springer.

Robertson, S., and Zaragoza, H. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333-389.

Shen, Y., He, X., Gao. J., Deng, L., and Mesnil, G. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM*.

Socher, R., Huval, B., Manning, C., Ng, A., 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*.

Stefanidis, K., Efthymiou, V., Herschel, M., and Christophides, V. 2013. Entity resolution in the web of data. *CIKM'13 Tutorial*.

Szabo, G., and Huberman, B. A. 2010. Predicting the popularity of online content. *Communications of the ACM*, 53(8).

Wu, Q., Burges, C.J.C., Svore, K., and Gao, J. 2009. Adapting boosting for information retrieval measures. *Journal of Information Retrieval*, 13(3):254-270.

Yih, W., Goodman, J., and Carvalho, V. R. 2006. Finding advertising keywords on web pages. In *WWW*.

Yih, W., Toutanova, K., Platt, J., and Meek, C. 2011. Learning discriminative projections for text similarity measures. In *CoNLL*.

Translation Modeling with Bidirectional Recurrent Neural Networks

Martin Sundermeyer[1], **Tamer Alkhouli**[1], **Joern Wuebker**[1], **and Hermann Ney**[1,2]
[1]Human Language Technology and Pattern Recognition Group
RWTH Aachen University, Aachen, Germany
[2]Spoken Language Processing Group
Univ. Paris-Sud, France and LIMSI/CNRS, Orsay, France
{surname}@cs.rwth-aachen.de

Abstract

This work presents two different translation models using recurrent neural networks. The first one is a word-based approach using word alignments. Second, we present phrase-based translation models that are more consistent with phrase-based decoding. Moreover, we introduce bidirectional recurrent neural models to the problem of machine translation, allowing us to use the full source sentence in our models, which is also of theoretical interest. We demonstrate that our translation models are capable of improving strong baselines already including recurrent neural language models on three tasks: IWSLT 2013 German→English, BOLT Arabic→English and Chinese→English. We obtain gains up to 1.6% BLEU and 1.7% TER by rescoring 1000-best lists.

1 Introduction

Neural network models have recently experienced unprecedented attention in research on statistical machine translation (SMT). Several groups have reported strong improvements over state-of-the-art baselines using feedforward neural network-based language models (Schwenk et al., 2006; Vaswani et al., 2013), as well as translation models (Le et al., 2012; Schwenk, 2012; Devlin et al., 2014). Different from the feedforward design, *recurrent* neural networks (RNNs) have the advantage of being able to take into account an unbounded history of previous observations. In theory, this enables them to model long-distance dependencies of arbitrary length. However, while previous work on translation modeling with recurrent neural networks shows its effectiveness on standard baselines, so far no notable gains have been presented on top of recurrent language models (Auli et al., 2013; Kalchbrenner and Blunsom, 2013; Hu et al., 2014).

In this work, we present two novel approaches to recurrent neural translation modeling: word-based and phrase-based. The word-based approach assumes one-to-one aligned source and target sentences. We evaluate different ways of resolving alignment ambiguities to obtain such alignments. The phrase-based RNN approach is more closely tied to the underlying translation paradigm. It models actual phrasal translation probabilities while avoiding sparsity issues by using single words as input and output units. Furthermore, in addition to the unidirectional formulation, we are the first to propose a bidirectional architecture which can take the full source sentence into account for all predictions. Our experiments show that these models can improve state-of-the-art baselines containing a recurrent language model on three tasks. For our competitive IWSLT 2013 German→English system, we observe gains of up to 1.6% BLEU and 1.7% TER. Improvements are also demonstrated on top of our evaluation systems for BOLT Arabic→English and Chinese→English, which also include recurrent neural language models.

The rest of this paper is structured as follows. In Section 2 we review related work and in Section 3 an overview of long short-term memory (LSTM) neural networks, a special type of recurrent neural networks we make use of in this work, is given. Section 4 describes our novel translation models. Finally, experiments are presented in Section 5 and we conclude with Section 6.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 14–25,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

2 Related Work

In this Section we contrast previous work to ours, where we design RNNs to model bilingual dependencies, which are applied to rerank n-best lists after decoding.

To the best of our knowledge, the earliest attempts to apply neural networks in machine translation (MT) are presented in (Castaño et al., 1997; Castaño and Casacuberta, 1997; Castaño and Casacuberta, 1999), where they were used for example-based MT.

Recently, Le et al. (2012) presented translation models using an output layer with classes and a shortlist for rescoring using feedforward networks. They compare between word-factored and tuple-factored n-gram models, obtaining their best results using the word-factored approach, which is less amenable to data sparsity issues. Both of our word-based and phrase-based models eventually work on the word level. Kalchbrenner and Blunsom (2013) use recurrent neural networks with full source sentence representations. The continuous representations are obtained by applying a sequence of convolutions, and the result is fed into the hidden layer of a recurrent language model. Rescoring results indicate no improvements over the state of the art. Auli et al. (2013) also include source sentence representations built either using Latent Semantic Analysis or by concatenating word embeddings. This approach produced no notable gain over systems using a recurrent language model. On the other hand, our proposed bidirectional models include the full source sentence relying on recurrency, yielding improvements over competitive baselines already including a recurrent language model.

RNNs were also used with minimum translation units (Hu et al., 2014), which are phrase pairs undergoing certain constraints. At the input layer, each of the source and target phrases are modeled as a bag of words, while the output phrase is predicted word-by-word assuming conditional independence. The approach seeks to alleviate data sparsity problems that would arise if phrases were to be uniquely distinguished. Our proposed phrase-based models maintain word order within phrases, but the phrases are processed in a word-pair manner, while the phrase boundaries remain implicitly encoded in the way the words are presented to the network. Schwenk (2012) proposed a feedforward network that predicts phrases of a fixed maximum length, such that all phrase words are predicted at once. The prediction is conditioned on the source phrase. Since our phrase-based model predicts one word at a time, it does not assume any phrase length. Moreover, our model's predictions go beyond phrase boundaries and cover unbounded history and future contexts.

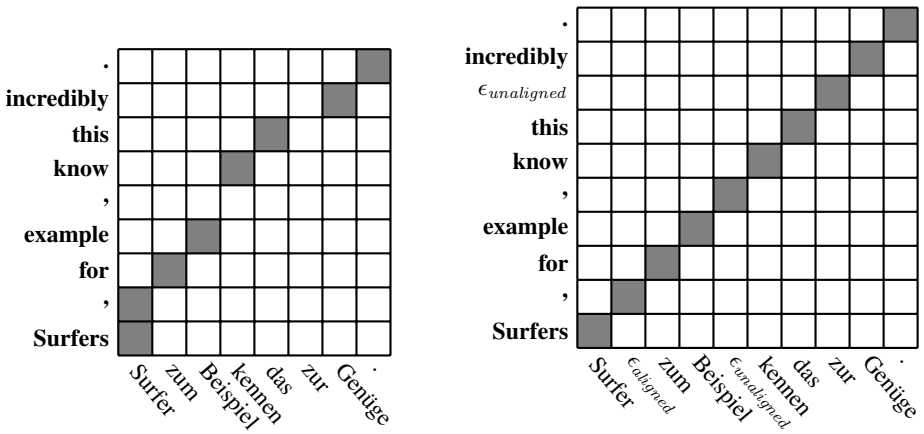
Using neural networks during decoding requires tackling the costly output normalization step. Vaswani et al. (2013) avoid this step by training feedforward neural language models using *noise contrastive estimation*, while Devlin et al. (2014) augment the training objective function to produce approximately normalized scores directly. The latter work makes use of translation and joint models, and pre-computes the first hidden layer beforehand, resulting in large speedups. They report major improvements over strong baselines. The speedups achieved by both works allowed to integrate feedforward neural networks into the decoder.

3 LSTM Recurrent Neural Networks

Our work is based on recurrent neural networks. In related fields like e. g. language modeling, this type of neural network has been shown to perform considerably better than standard feedforward architectures (Mikolov et al., 2011; Arisoy et al., 2012; Sundermeyer et al., 2013; Liu et al., 2014).

Most commonly, recurrent neural networks are trained with stochastic gradient descent (SGD), where the gradient of the training criterion is computed with the backpropagation through time algorithm (Rumelhart et al., 1986; Werbos, 1990; Williams and Zipser, 1995). However, the combination of RNN networks with conventional backpropagation training leads to conceptual difficulties which are known as the vanishing (or exploding) gradient problem, described e. g. in (Bengio et al., 1994). To remedy this problem, in (Hochreiter and Schmidhuber, 1997) it was suggested to modify the architecture of a standard RNN in such a way that vanishing and exploding gradients are avoided during backpropagation. In particular, no modification of the training algorithm is necessary. The resulting architecture is referred to as long short-term memory (LSTM) neural network.

Bidirectional recurrent neural networks (BRNNs) were first proposed in (Schuster and Paliwal, 1997) and applied to speech recognition tasks. They have been since applied to different



(a) Original

(b) One-to-one alignment

Figure 1: Example sentence from the German→English IWSLT data. The one-to-one alignment is created by introducing $\epsilon_{aligned}$ and $\epsilon_{unaligned}$ tokens.

tasks like parsing (Henderson, 2004) and spoken language understanding (Mesnil et al., 2013). Bidirectional long short-term memory (BLSTM) networks are BRNNs using LSTM hidden layers (Graves and Schmidhuber, 2005). This work introduces BLSTMs to the problem of machine translation, allowing powerful models that employ unlimited history and future information to make predictions.

While the proposed models do not make any assumptions about the type of RNN used, all of our experiments make use of recurrent LSTM neural networks, where we include later LSTM extensions proposed in (Gers et al., 2000; Gers et al., 2003). The cross-entropy error criterion is used for training. Further details on LSTM neural networks can be found in (Graves and Schmidhuber, 2005; Sundermeyer et al., 2012).

4 Translation Modeling with RNNs

In the following we describe our word- and phrase-based translation models in detail. We also show how bidirectional RNNs can enable such models to include full source information.

4.1 Resolving Alignment Ambiguities

Our word-based recurrent models are only defined for one-to-one-aligned source-target sentence pairs. In this work, we always evaluate the model in the order of the target sentence. However, we experiment with several different ways to resolve ambiguities due to unaligned or multiply aligned words. To that end, we introduce two additional tokens, $\epsilon_{aligned}$ and $\epsilon_{unaligned}$. Unaligned words are either removed or aligned to an extra $\epsilon_{unaligned}$ token on the opposite side. If an $\epsilon_{unaligned}$ is introduced on the target side, its position is determined by the aligned source word that is closest to the unaligned source word in question, preferring left to right. To resolve one-to-many alignments, we use an IBM-1 translation table to decide for one of the alignment connections to be kept. The remaining words are also either deleted or aligned to additionally introduced $\epsilon_{aligned}$ tokens on the opposite side. Fig. 1 shows an example sentence from the IWSLT data, where all ϵ tokens are introduced.

In a short experiment, we evaluated 5 different setups with our unidirectional RNN translation model (cf. next Section): without any ϵ tokens, without $\epsilon_{unaligned}$, source identity, target identity and using all ϵ tokens. Source identity means we

	dev		test	
	BLEU	TER	BLEU	TER
baseline	33.5	45.8	30.9	48.4
w/o ϵ	34.2	45.3	31.8	47.7
w/o $\epsilon_{unaligned}$	34.4	44.8	31.7	47.4
source identity	34.5	45.0	31.9	47.5
target identity	34.5	44.6	31.9	47.0
all ϵ	34.6	44.5	32.0	47.1

Table 1: Comparison of including different sets of ϵ tokens into the one-to-one alignment on the IWSLT 2013 German→English task using the unidirectional RNN translation model.

introduce no ϵ tokens on source side, but all on target side. Target identity is defined analogously. The results can be found in Tab. 1. We use the setup with all ϵ tokens in all following experiments, which showed the best BLEU performance.

4.2 Word-based RNN Models

Given a pair of source sequence $f_1^I = f_1 \dots f_I$ and target sequence $e_1^I = e_1 \dots e_I$, where we assume a direct correspondence between f_i and e_i, we define the posterior translation probability by factorizing on the target words:

$$p(e_1^I|f_1^I) = \prod_{i=1}^{I} p(e_i|e_1^{i-1}, f_1^I) \quad (1)$$

$$\approx \prod_{i=1}^{I} p(e_i|e_1^{i-1}, f_1^{i+d}) \quad (2)$$

$$\approx \prod_{i=1}^{I} p(e_i|f_1^{i+d}). \quad (3)$$

We denote the formulation (1) as the bidirectional joint model (BJM). This model can be simplified by several independence assumptions. First, we drop the dependency on the future source information, receiving what we denote as the unidirectional joint model (JM) in (2). Here, $d \in \mathbb{N}_0$ is a delay parameter, which is set to $d = 0$ for all experiments, except for the comparative results reported in Fig. 7. Finally, assuming conditional independence from the previous target sequence, we receive the unidirectional translation model (TM) in (3). Analogously, we can define a bidirectional translation model (BTM) by keeping the dependency on the full source sentence f_1^I, but dropping the previous target sequence e_1^{i-1}:

$$p(e_1^I|f_1^I) \approx \prod_{i=1}^{I} p(e_i|f_1^I). \quad (4)$$

Fig. 2 shows the dependencies of the word-based neural translation and joint models. The alignment points are traversed in target order and at each time step one target word is predicted. The pure translation model (TM) takes only source words as input, while the joint model (JM) takes the preceding target words as an additional input. A delay of $d > 0$ is implemented by shifting the target sequence by d time steps and filling the first d target positions and the last d source positions with a dedicated $\epsilon_{padding}$ symbol. The RNN architecture for the unidirectional word-based models

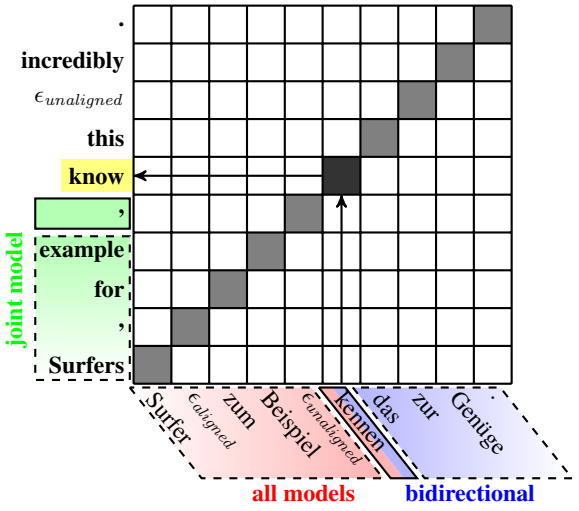


Figure 2: Dependencies modeled within the word-based RNN models when predicting the target word 'know'. Directly processed information is depicted with solid rectangles, and information available through recurrent connections is marked with dashed rectangles.

is illustrated in Fig. 3, which corresponds to the following set of equations:

$$\begin{aligned} y_i &= A_1 \hat{f}_i + A_2 \hat{e}_{i-1} \\ z_i &= \xi(y_i; A_3, y_1^{i-1}) \\ p\big(c(e_i)|e_1^{i-1}, f_1^i\big) &= \varphi_{c(e_i)}(A_4 z_i) \\ p\big(e_i|c(e_i), e_1^{i-1}, f_1^i\big) &= \varphi_{e_i}(A_{c(e_i)} z_i) \\ p(e_i|e_1^{i-1}, f_1^i) &= p\big(e_i|c(e_i), e_1^{i-1}, f_1^i\big) \cdot \\ &\quad p\big(c(e_i)|e_1^{i-1}, f_1^i\big) \end{aligned}$$

Here, by $\hat{f}_i$ and $\hat{e}_{i-1}$ we denote the one-hot encoded vector representations of the source and target words f_i and e_{i-1}. The outgoing activation values of the projection layer and the LSTM layer are y_i and z_i, respectively. The matrices A_j contain the weights of the neural network layers. By $\xi(\cdot\,; A_3, y_1^{i-1})$ we denote the LSTM formalism that we plug in at the third layer. As the LSTM layer is recurrent, we explicitly include the dependence on the previous layer activations y_1^{i-1}. Finally, φ is the widely-used softmax function to obtain normalized probabilities, and c denotes a word class mapping from any target word to its unique word class. For the bidirectional model, the equations can be defined analogously.

Due to the use of word classes, the output layer consists of two parts. The class probability $p\big(c(e_i)|e_1^{i-1}, f_1^i\big)$ is computed first, and then

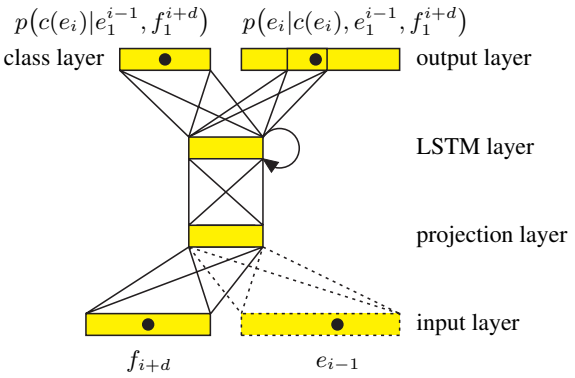


Figure 3: Architecture of a recurrent unidirectional translation model. By including the dashed parts, a *joint* model is obtained.

the word probability $p(e_i|c(e_i), e_1^{i-1}, f_1^i)$ is obtained given the word class. This trick helps avoiding the otherwise computationally expensive normalization sum, which would be carried out over all words in the target vocabulary. In a class-factorized output layer where each word belongs to a single class, the normalization is carried out over all classes, whose number is typically much less than the vocabulary size. The other normalization sum needed to produce the word probability is limited to the words belonging to the same class (Goodman, 2001; Morin and Bengio, 2005).

4.3 Phrase-based RNN Models

One of the conceptual disadvantages of word-based modeling as introduced in the previous section is that there is a mismatch between training and testing conditions: During neural network training, the vocabulary has to be extended by additional ϵ tokens, and a one-to-one alignment is used which does not reflect the situation in decoding. In phrase-based machine translation, more complex alignments in terms of multiple words on both the source and the target sides are used, which allow the decoder to make use of richer short-distance dependencies and are crucial for the performance of the resulting system.

From this perspective, it seems interesting to standardize the alignments used in decoding, and in training the neural network. However, it is difficult to use the phrases themselves as the vocabulary of the RNN. Usually, the huge number of potential phrases in comparison to the relatively small amount of training data makes the learning of continuous phrase representations difficult

Surfers Surfer	, for example , zum Beispiel	know kennen	this das	incredibly zur Genüge	. .

Figure 4: Example phrase alignment for a sentence from the IWSLT training data.

due to data sparsity. This is confirmed by results presented in (Le et al., 2012), which show that a word-factored translation model outperforms the phrase-factored version. Therefore, in this work we continue relying on source and target word vocabularies for building our phrase representations. However, we no longer use a direct correspondence between a source and a target word, as enforced in our word-based models.

Fig. 4 shows an example phrase alignment, where a sequence of source words $\tilde{f}_i$ is directly mapped to a sequence of target words $\tilde{e}_i$ for $1 \leq i \leq \tilde{I}$. By $\tilde{I}$, we denote the number of phrases in the alignment. We decompose the target sentence posterior probability in the following way:

$$p(e_1^I|f_1^J) = \prod_{i=1}^{\tilde{I}} p(\tilde{e}_i|\tilde{e}_1^{i-1}, \tilde{f}_1^{\tilde{I}}) \quad (5)$$

$$\approx \prod_{i=1}^{\tilde{I}} p(\tilde{e}_i|\tilde{e}_1^{i-1}, \tilde{f}_1^{i}) \quad (6)$$

where the joint model in Eq. 5 would correspond to a bidirectional RNN, and Eq. 6 only requires a unidirectional RNN. By leaving out the conditioning on the target side, we obtain a phrase-based *translation* model.

As there is no one-to-one correspondence between the words within a phrase, the basic idea of our phrase-based approach is to let the neural network learn the dependencies itself, and present the full source side of the phrase to the network before letting it predict target side words. Then the probability for the target side of a phrase can be computed, in case of Eq. 6, by:

$$p(\tilde{e}_i|\tilde{e}_1^{i-1}, \tilde{f}_1^{\tilde{I}}) = \prod_{j=1}^{|\tilde{e}_i|} p\big((\tilde{e}_i)_j|(\tilde{e}_i)_1^{j-1}, \tilde{e}_1^{i-1}, \tilde{f}_1^{i}\big),$$

and analogously for the case of Eq. 5. Here, $(\tilde{e}_i)_j$ denotes the j-th word of the i-th aligned target phrase.

We feed the source side of a phrase into the neural network one word at a time. Only when the

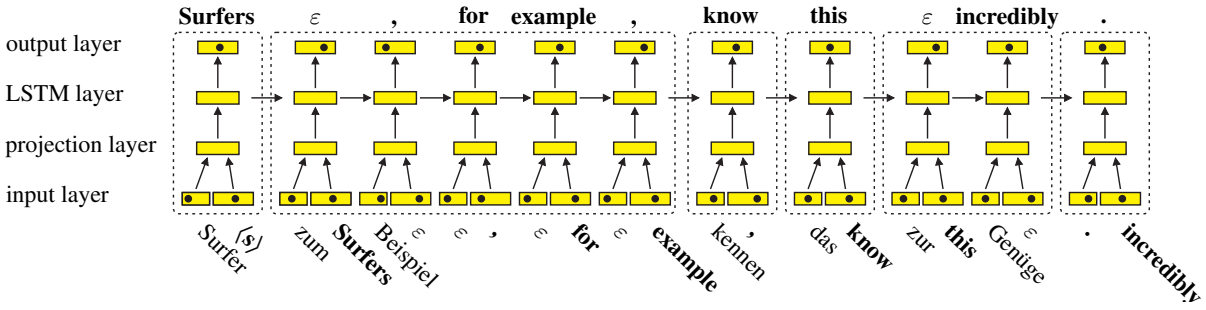


Figure 5: A recurrent phrase-based joint translation model, unfolded over time. Source words are printed in normal face, while target words are printed in bold face. Dashed lines indicate phrases from the example sentence. For brevity, we omit the precise handling of sentence begin and end tokens.

presentation of the source side is finished we start estimating probabilities for the target side. Therefore, we do not let the neural network learn a target distribution until the very last source word is considered. In this way, we break up the conventional RNN training scheme where an input sample is directly followed by its corresponding teacher signal. Similarly, the presentation of the source side of the next phrase only starts after the prediction of the current target side is completed.

To this end, we introduce a no-operation token, denoted by ε, which is not part of the vocabulary (which means it cannot be input to or predicted by the RNN). When the ε token occurs as input, it indicates that no input needs to be processed by the RNN. When the ε token occurs as a teacher signal for the RNN, the output layer distribution is ignored, and does not even have to be computed. In both cases, all the other layers are still processed during forward and backward passes such that the RNN state can be advanced even without additional input or output.

Fig. 5 depicts the evaluation of a phrase-based joint model for the example alignment from Fig. 4. For a source phrase $\tilde{f}_i$, we include $(|\tilde{e}_i|-1)$ many ε symbols at the end of the phrase. Conversely, for a target phrase $\tilde{e}_i$, we include $(|\tilde{f}_i| - 1)$ many ε symbols at the beginning of the phrase.

E. g., in the figure, the second dashed rectangle from the left depicts the training of the English phrase "*, for example ,*" and its German translation "*zum Beispiel*". At the input layer, we feed in the source words one at a time, while we present ε tokens at the target side input layer and the output layer (with the exception of the very first time step, where we still have the last target word from the previous phrase as input instead of ε). With the last word of the source phrase "*Beispiel*" being presented to the network, the full source phrase is stored in the hidden layer, and the neural network is then trained to predict the target phrase words at the output layer. Subsequently, the source input is ε, and the target input is the most recent target side history word.

To obtain a phrase-aligned training sequence for the phrase-based RNN models, we force-align the training data with the application of leave-one-out as described in (Wuebker et al., 2010).

4.4 Bidirectional RNN Architecture

While the unidirectional RNNs include an unbounded sentence history, they are still limited in the number of future source words they include. Bidirectional models provide a flexible means to also include an unbounded future context, which, unlike the delayed unidirectional models, require no tuning to determine the amount of delay.

Fig. 6 illustrates the bidirectional model architecture, which is an extension of the unidirectional model of Fig. 3. First, an additional recurrent hidden layer is added in parallel to the existing one. This layer will be referred to as the backward layer, since it processes information in backward time direction. This hidden layer receives source word input only, while target words in the case of a joint model are fed to the forward layer as in the unidirectional case. Due to the backward recurrency, the backward layer will make the information f_i^I available when predicting the target word e_i, while the forward layer takes care of the source history f_1^i. Jointly, the forward and backward branches include the full source sentence f_1^I, as indicated in Fig. 2. Fig. 6 shows the "deep" variant of the bidirectional model, where the for-

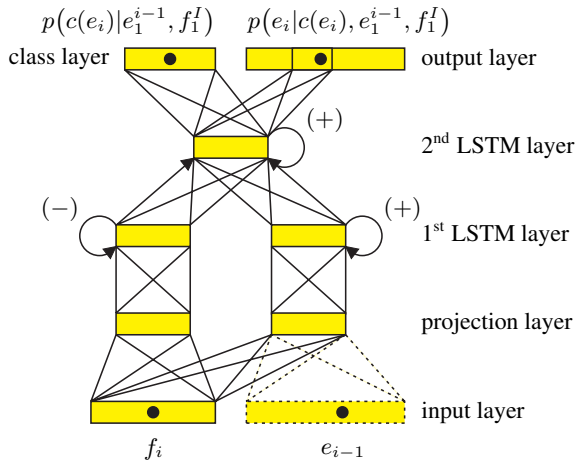


Figure 6: Architecture of a recurrent bidirectional translation model. By $(+)$ and $(-)$, we indicate a processing in forward and backward time directions, respectively. The inclusion of the dashed parts leads to a bidirectional *joint* model. One source projection matrix is used for the forward and backward branches.

ward and backward layers converge into a hidden layer. A shallow variant can be obtained if the parallel layers converge into the output layer directly[1].

Due to the full dependence on the source sequence, evaluating bidirectional networks requires computing the forward pass of the forward and backward layers for the full sequence, before being able to evaluate the next layers. In the backward pass of backpropagation, the forward and backward recurrent layers are processed in decreasing and increasing time order, respectively.

5 Experiments

5.1 Setup

All translation experiments are performed with the *Jane* toolkit (Vilar et al., 2010; Wuebker et al., 2012). The largest part of our experiments is carried out on the IWSLT 2013 German→English shared translation task.[2] The baseline system is trained on all available bilingual data, 4.3M sentence pairs in total, and uses a 4-gram LM with modified Kneser-Ney smoothing (Kneser and Ney, 1995; Chen and Goodman, 1998), trained with the SRILM toolkit (Stolcke, 2002). As additional data sources for the LM we selected parts of the Shuffled News and LDC English Gigaword corpora based on cross-entropy difference (Moore and Lewis, 2010), resulting in a total of 1.7 billion running words for LM training. The state-of-the-art baseline is a standard phrase-based SMT system (Koehn et al., 2003) tuned with MERT (Och, 2003). It contains a hierarchical reordering model (Galley and Manning, 2008) and a 7-gram word cluster language model (Wuebker et al., 2013). Here, we also compare against a feed-forward joint model as described by Devlin et al. (2014), with a source window of 11 words and a target history of three words, which we denote as *BBN-JM*. Instead of POS tags, we predict word classes trained with `mkcls`. We use a shortlist of size 16K and 1000 classes for the remaining words. All neural networks are trained on the TED portion of the data (138K segments) and are applied in a rescoring step on 1000-best lists.

To confirm our results, we run additional experiments on the Arabic→English and Chinese→English tasks of the DARPA BOLT project. In both cases, the neural network models are added on top of our most competitive evaluation system. On Chinese→English, we use a hierarchical phrase-based system trained on 3.7M segments with 22 dense features, including an advanced orientation model (Huck et al., 2013). For the neural network training, we selected a subset of 9M running words. The Arabic→English system is a standard phrase-based decoder trained on 6.6M segments, using 17 dense features. The neural network training was performed using a selection amounting to 15.5M running words. For both tasks we apply the neural networks by rescoring 1000-best lists and evaluate results on two data sets from the 'discussion forum' domain, `test1` and `test2`. The sizes of the data sets for the Arabic→English system are: 1219 (`dev`), 1510 (`test1`), and 1137 (`test2`) segments, and for the Chinese→English system are: 5074 (`dev`), 1844 (`test1`), and 1124 (`test2`) segments. All results are measured in case-insensitive BLEU [%] (Papineni et al., 2002) and TER [%] (Snover et al., 2006) on a single reference.

5.2 Results

Our results on the IWSLT German→English task are summarized in Tab. 2. At this point, we do not include a recurrent neural network lan-

[1] In our implementation, the forward and backward layers converge into an intermediate identity layer, and the aggregate is weighted and fed to the next layer.

[2] `http://www.iwslt2013.org`

	dev		test	
	BLEU	TER	BLEU	TER
baseline	33.5	45.8	30.9	48.4
TM	34.6	44.5	32.0	47.1
JM	34.7	44.7	31.8	47.4
BTM	34.7	44.9	32.3	47.0
BTM (deep)	34.8	44.3	**32.5**	46.7
BJM	34.7	44.5	32.1	47.0
BJM (deep)	**34.9**	**44.1**	32.2	**46.6**
PTM	34.3	44.9	32.1	47.5
PJM	34.3	45.0	32.0	47.5
PJM (10-best)	34.4	44.8	32.0	47.3
PJM (deep)	34.6	44.7	32.0	47.6
PBJM (deep)	34.8	44.9	31.9	47.5
BBN-JM	34.4	44.9	31.9	47.6

Table 2: Results for the IWSLT 2013 German→English task with different RNN models. T: translation, J: joint, B: bidirectional, P: phrase-based.

guage model yet. Here, the delay parameter d from Equations 2 and 3 is set to zero. We observe that for all recurrent translation models, we achieve substantial improvements over the baseline on the test data, ranging from 0.9 BLEU up to 1.6 BLEU. These results are also consistent with the improvements in terms of TER, where we achieve reductions by 0.8 TER up to 1.8 TER.

These numbers can be directly compared to the case of feedforward neural network-based translation modeling as proposed in (Devlin et al., 2014) which we include in the very last row of the table. Nearly all of our recurrent models outperform the feedforward approach, where the RNN model performing best on the dev data is better on test by 0.3 BLEU and 1.0 TER.

Interestingly, for the recurrent word-based models, on the test data it can be seen that TMs perform better than JMs, even though TMs do not take advantage of the target side history words. However, exploiting this extra information does not always need to result in a better model, as the target side words are only derived from the given source side, which is available to both TMs and JMs. On the other hand, including future source words in a bidirectional model clearly improves the performance further. By adding another LSTM

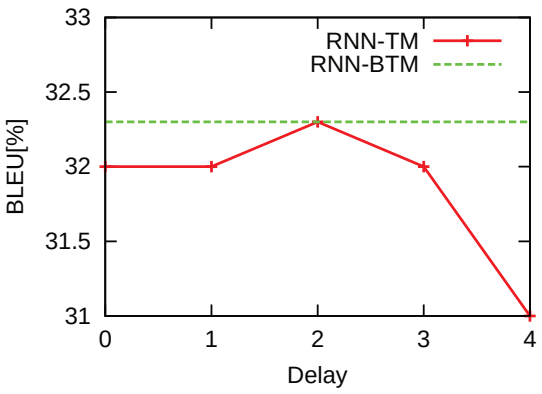


Figure 7: BLEU scores on the IWSLT test set with different delays for the unidirectional RNN-TM and the bidirectional RNN-BTM.

layer that combines forward and backward time directions (indicated as 'deep' in the table), we obtain our overall best model.

In Fig. 7 we compare the word-based bidirectional TM with a unidirectional TM that uses different time delays $d = 0, \ldots, 4$. For a delay $d = 2$, the same performance is obtained as with the bidirectional model, but this comes at the price of tuning the delay parameter.

In comparison to the unidirectional word-based models, phrase-based models perform similarly. In the tables, we include those phrase-based variants which perform best on the dev data, where phrase-based JMs always are at least as good or better than the corresponding TMs in terms of BLEU. Therefore, we mainly report JM results for the phrase-based networks. A phrase-based model can also be trained on multiple variants for the phrase alignment. For our experiments, we tested 10-best alignments against the single best alignment, which resulted in a small improvement of 0.2 TER on both dev and test. We did not observe consistent gains by using an additional hidden layer or bidirectional models. To some extent, future information is already considered in unidirectional phrase-based models by feeding the complete source side before predicting the target side.

Tab. 3 shows different model combination results for the IWSLT task, where a recurrent language model is included in the baseline. Adding a deep bidirectional TM or JM to the recurrent language model improves the RNN-LM baseline by 1.2 BLEU or 1.1 BLEU, respectively. A phrase-based model substantially improves over

	dev		eval11		test	
	BLEU[%]	TER[%]	BLEU[%]	TER[%]	BLEU[%]	TER[%]
baseline (w/ RNN-LM)	34.3	44.8	36.4	42.9	31.5	47.8
BTM (deep)	34.9	**43.7**	**37.6**	**41.5**	**32.7**	**46.1**
BJM (deep)	**35.0**	44.4	37.4	41.9	32.6	46.5
PBJM (deep)	34.8	44.6	36.9	42.6	32.3	47.2
4 RNN models	**35.2**	**43.4**	**38.0**	**41.2**	**32.7**	**46.0**

Table 3: Results for the IWSLT 2013 German→English task with different RNN models. All results include a recurrent language model. T: translation, J: joint, B: bidirectional, P: phrase-based.

the RNN-LM baseline, but performs not as good as its word-based counterparts. By adding four different translation models, including models in reverse word order and reverse translation direction, we are able to improve these numbers even further. However, especially on the test data, the gains from model combination saturate quickly.

Apart from the IWSLT track, we also analyze the performance of our translation models on the BOLT Chinese→English and Arabic→English translation tasks. Due to the large amount of training data, we concentrate on models of high performance in the IWSLT experiments. The results can be found in Tab. 4 and 5. In both cases, we see consistent improvements over the recurrent neural network language model baseline, improving the Arabic→English system by 0.6 BLEU and 0.5 TER on test1. This can be compared to the rescoring results for the same task reported by (Devlin et al., 2014), where they achieved 0.3 BLEU, despite the fact that they used multiple references for scoring, whereas in our experiments we rely on a single reference only. The models are also able to improve the Chinese→English system by 0.5 BLEU and 0.5 TER on test2.

	test1		test2	
	BLEU	TER	BLEU	TER
baseline	25.2	57.4	26.8	57.3
BTM (deep)	25.6	**56.6**	26.8	56.7
BJM (deep)	25.9	56.9	27.4	56.7
RNN-LM	25.6	57.1	27.5	56.7
+ BTM (deep)	25.9	56.7	27.3	56.8
+ BJM (deep)	**26.2**	**56.6**	**27.9**	**56.5**

Table 4: Results for the BOLT Arabic→English task with different RNN models. The "+" sign in the last two rows indicates that either of the corresponding deep models (BTM and BJM) are added to the baseline including the recurrent language model (i.e. they are not applied at the same time). T: translation, J: joint, B: bidirectional.

5.3 Analysis

To investigate whether bidirectional models benefit from future source information, we compare the single-best output of a system reranked with a unidirectional model to the output reranked with a bidirectional model. We choose the models to be translation models in both cases, as they predict target words independent of previous predictions, given the source information (cf. Eqs. (3, 4)). This makes it easier to detect the effect of including future source information or the lack thereof. The examples are taken from the IWSLT task, where we include the one-to-one source information, reordered according to the target side.

source: nicht **so wie** ich
reference: not **like** me
Hypothesis 1:
1-to-1 source: **so** ich ϵ nicht **wie**
1-to-1 target: **so** I do n't **like**
Hypothesis 2:
1-to-1 source: nicht **so wie** ich
1-to-1 target: not ϵ **like** me

In this example, the German phrase "*so wie*" translates to "*like*" in English. The bidirectional model prefers hypothesis 2, making use of the future word "*wie*" when translating the German word "*so*" to ϵ, because it has future insight that this move will pay off later when translating

	BLEU	TER	BLEU	TER
baseline	18.3	63.6	16.7	63.0
BTM (deep)	18.7	63.3	17.1	62.6
BJM (deep)	18.5	**63.1**	17.2	**62.3**
RNN-LM	18.8	63.3	17.2	62.8
+ BTM (deep)	**18.9**	**63.1**	**17.7**	**62.3**
+ BJM (deep)	18.8	63.3	17.5	62.5

Table 5: Results for the BOLT Chinese→English task with different RNN models. The "+" sign in the last two rows indicates that either of the corresponding deep models (BTM and BJM) are added to the baseline including the recurrent language model (i.e. they are not applied at the same time). T: translation, B: bidirectional.

the rest of the sentence. This information is not available to the unidirectional model, which prefers hypothesis 1 instead.

> *source*: das taten wir dann auch und verschafften uns so **eine Zeit lang** einen Wettbewerbs Vorteil .
> *reference*: and we actually did that and it gave us a competitive advantage **for a while** .
> **Hypothesis 1**:
> *1-to-1 source*: das ϵ ϵ ϵ wir dann auch taten und verschafften uns so **eine Zeit lang** einen Wettbewerbs Vorteil .
> *1-to-1 target*: that 's just what we ϵ ϵ did and gave us ϵ a **time ,** a competitive advantage .
> **Hypothesis 2**:
> *1-to-1 source*: das ϵ ϵ ϵ wir dann auch taten und verschafften uns so einen Wettbewerbs Vorteil ϵ **eine Zeit lang** .
> *1-to-1 target*: that 's just what we ϵ ϵ did and gave us ϵ a competitive advantage **for a ϵ while** .

Here, the German phrase "*eine Zeit lang*" translates to "*for a while*" in English. Bidirectional scoring favors hypothesis 2, while unidirectional scoring favors hypothesis 1. It seems that the unidirectional model translates "*Zeit*" to "*time*" as the object of the verb "*give*" in hypothesis 1, being blind to the remaining part "*lang*" of the phrase which changes the meaning. The bidirectional model, to its advantage, has the full source information, allowing it to make the correct prediction.

6 Conclusion

We developed word- and phrase-based RNN translation models. The former is simple and performs well in practice, while the latter is more consistent with the phrase-based paradigm. The approach inherently evades data sparsity problems as it works on words in its lowest level of processing. Our experiments show the models are able to achieve notable improvements over baselines containing a recurrent LM.

In addition, and for the first time in statistical machine translation, we proposed a bidirectional neural architecture that allows modeling past and future dependencies of any length. Besides its good performance in practice, the bidirectional architecture is of theoretical interest as it allows the exact modeling of posterior probabilities.

Acknowledgments

This material is partially based upon work supported by the DARPA BOLT project under Contract No. HR0011- 12-C-0015. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA. The research leading to these results has also received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreements n° 287658 and n° 287755. Experiments were performed with computing resources granted by JARA-HPC from RWTH Aachen University under project 'jara0085'. We would like to thank Jan-Thorsten Peter for providing the *BBN-JM* system.

References

Ebru Arisoy, Tara N. Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. 2012. Deep neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 20–28. Association for Computational Linguistics.

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint Language and Translation Modeling with Recurrent Neural Networks. In *Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, USA, October.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gra-

dient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Maria Asunción Castaño and Francisco Casacuberta. 1997. A connectionist approach to machine translation. In *5th International Conference on Speech Communication and Technology (EUROSPEECH-97)*, Rhodes, Greece.

Maria Asunción Castaño and Francisco Casacuberta. 1999. Text-to-text machine translation using the RECONTRA connectionist model. In *Lecture Notes in Computer Science (IWANN 99)*, volume 1607, pages 683–692, Alicante, Spain.

Maria Asunción Castaño, Francisco Casacuberta, and Enrique Vidal. 1997. Machine translation using neural networks and finite-state models. In *7th International Conference on Theoretical and Methodological Issues in Machine Translation. TMI'97*, pages 160–167, Santa Fe, USA.

Stanley F. Chen and Joshua Goodman. 1998. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, Cambridge, MA, August.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *52nd Annual Meeting of the Association for Computational Linguistics*, page to appear, Baltimore, MD, USA, June.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 848–856, Stroudsburg, PA, USA. Association for Computational Linguistics.

Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10):2451–2471.

Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. 2003. Learning precise timing with lstm recurrent networks. *The Journal of Machine Learning Research*, 3:115–143.

Joshua Goodman. 2001. Classes for fast maximum entropy training. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 561–564. IEEE.

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

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

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

Yuening Hu, Michael Auli, Qin Gao, and Jianfeng Gao. 2014. Minimum translation modeling with recurrent neural networks. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 20–29, Gothenburg, Sweden, April. Association for Computational Linguistics.

Matthias Huck, Joern Wuebker, Felix Rietig, and Hermann Ney. 2013. A phrase orientation model for hierarchical machine translation. In *ACL 2013 Eighth Workshop on Statistical Machine Translation*, pages 452–463, Sofia, Bulgaria, August.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, October. Association for Computational Linguistics.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for M-gram language modeling. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processingw*, volume 1, pages 181–184, May.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*, pages 127–133, Edmonton, Alberta.

Hai Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous Space Translation Models with Neural Networks. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 39–48, Montreal, Canada, June.

Xunying Liu, Yongqiang Wang, Xie Chen, Mark J. F. Gales, and Phil C. Woodland. 2014. Efficient lattice rescoring using recurrent neural network language models. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 4941–4945. IEEE.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*, pages 3771–3775.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, JH Cernocky, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing*

(ICASSP), 2011 IEEE International Conference on, pages 5528–5531. IEEE.

Robert C. Moore and William Lewis. 2010. Intelligent Selection of Language Model Training Data. In *ACL (Short Papers)*, pages 220–224, Uppsala, Sweden, July.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of the 41th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. *Learning Internal Representations by Error Propagation. In: J. L. McClelland, D. E. Rumelhart, and The PDP Research Group: "Parallel Distributed Processing, Volume 1: Foundations"*. The MIT Press.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Holger Schwenk, Daniel Déchelotte, and Jean-Luc Gauvain. 2006. Continuous Space Language Models for Statistical Machine Translation. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 723–730, Sydney, Australia, July.

Holger Schwenk. 2012. Continuous Space Translation Models for Phrase-Based Statistical Machine Translation. In *25th International Conference on Computational Linguistics (COLING)*, pages 1071–1080, Mumbai, India, December.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts, USA, August.

Andreas Stolcke. 2002. SRILM – An Extensible Language Modeling Toolkit. In *Proc. of the Int. Conf. on Speech and Language Processing (ICSLP)*, volume 2, pages 901–904, Denver, CO, September.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Interspeech*, Portland, OR, USA, September.

Martin Sundermeyer, Ilya Oparin, Jean-Luc Gauvain, Ben Freiberg, Ralf Schlüter, and Hermann Ney. 2013. Comparison of feedforward and recurrent neural network language models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 8430–8434, Vancouver, Canada, May.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, Seattle, Washington, USA, October. Association for Computational Linguistics.

David Vilar, Daniel Stein, Matthias Huck, and Hermann Ney. 2010. Jane: Open source hierarchical translation, extended with reordering and lexicon models. In *ACL 2010 Joint Fifth Workshop on Statistical Machine Translation and Metrics MATR*, pages 262–270, Uppsala, Sweden, July.

Paul J. Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Ronald J. Williams and David Zipser. 1995. *Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity. In: Yves Chauvain and David E. Rumelhart: "Back-Propagation: Theory, Architectures and Applications"*. Lawrence Erlbaum Publishers.

Joern Wuebker, Arne Mauser, and Hermann Ney. 2010. Training phrase translation models with leaving-one-out. In *Proceedings of the 48th Annual Meeting of the Assoc. for Computational Linguistics*, pages 475–484, Uppsala, Sweden, July.

Joern Wuebker, Matthias Huck, Stephan Peitz, Malte Nuhn, Markus Freitag, Jan-Thorsten Peter, Saab Mansour, and Hermann Ney. 2012. Jane 2: Open source phrase-based and hierarchical statistical machine translation. In *International Conference on Computational Linguistics*, pages 483–491, Mumbai, India, December.

Joern Wuebker, Stephan Peitz, Felix Rietig, and Hermann Ney. 2013. Improving statistical machine translation with word class models. In *Conference on Empirical Methods in Natural Language Processing*, pages 1377–1381, Seattle, USA, October.

A Neural Network Approach to Selectional Preference Acquisition

Tim Van de Cruys
IRIT & CNRS
Toulouse, France
tim.vandecruys@irit.fr

Abstract

This paper investigates the use of neural networks for the acquisition of selectional preferences. Inspired by recent advances of neural network models for NLP applications, we propose a neural network model that learns to discriminate between felicitous and infelicitous arguments for a particular predicate. The model is entirely unsupervised – preferences are learned from unannotated corpus data. We propose two neural network architectures: one that handles standard two-way selectional preferences and one that is able to deal with multi-way selectional preferences. The model's performance is evaluated on a pseudo-disambiguation task, on which it is shown to achieve state of the art performance.

1 Introduction

Predicates often have a semantically motivated preference for particular arguments. Compare for example the sentences in (1) and (2).

(1) The vocalist sings a ballad.

(2) The exception sings a tomato.

Most language users would have no problems accepting the first sentence as well-formed: a vocalist can be expected to sing, and a ballad is something that can be sung. The same language users, however, would likely consider the second sentence to be ill-formed: an exception is not supposed to sing, nor is a tomato something that is typically sung. Within the field of natural language processing, this inclination of predicates to select for particular arguments is known as *selectional preference*.

The automatic acquisition of selectional preferences has been a popular research subject within the field of natural language processing. An automatically acquired selectional preference resource is a versatile tool for numerous NLP applications, such as semantic role labeling (Gildea and Jurafsky, 2002), word sense disambiguation (McCarthy and Carroll, 2003), and metaphor processing (Shutova et al., 2013).

Models for selectional preference need to adequately deal with the consequences of Zipf's law: language is inherently sparse, and the majority of language utterances occur very infrequently. As a consequence, models that are based on corpus data need to properly generalize beyond the mere co-occurrence frequencies of sparse corpus data, taking into account the semantic similarity of both predicates and arguments. Researchers have come up with various approaches to this generalization step. Earlier approaches to selectional preference acquisition mostly rely on hand-crafted resources such as WordNet (Resnik, 1996; Li and Abe, 1998; Clark and Weir, 2001), while later approaches tend to take advantage of unsupervised learning machinery, such as latent variable models (Rooth et al., 1999; Ó Séaghdha, 2010) and distributional similarity metrics (Erk, 2007; Padó et al., 2007).

This paper investigates the use of neural networks for the acquisition of selectional preferences. Inspired by recent advances of neural network models for NLP applications (Collobert and Weston, 2008; Mikolov et al., 2013), we propose a neural network model that learns to discriminate between felicitous and infelicitous arguments for a particular predicate. The model is entirely unsupervised – preferences are learned from unannotated corpus data. Positive training instances are constructed from attested corpus data, while negative instances are constructed from randomly corrupted instances. We propose two neural network architectures: one that handles standard two-way selectional preferences and one that is able to deal with multi-way selectional preferences, where the interaction be-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 26–35,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

tween multiple verb arguments is taken into account. The model's performance is evaluated on a pseudo-disambiguation task, on which it is shown to achieve state of the art performance.

The contributions of this paper are twofold. First of all, we apply and evaluate a neural network approach to the problem of standard (two-way) selectional preference acquisition. Selectional preference acquisition using neural networks has not yet been explored in the literature. Secondly, we propose a novel network architecture and training objective for the acquisition of multi-way selectional preferences, where the interaction between a verb and its various arguments is captured at the same time.

The remainder of this paper is as follows. Section 2 first discusses related work with respect to selectional preference acquisition and neural network modeling. Section 3 describes our neural network architecture and its training procedure. Section 4 evaluates the model's performance, comparing it to other existing models for selectional preference acquisition. Finally, section 5 concludes and indicates a number of avenues for future work.

2 Related Work

2.1 Selectional preferences

One of the first approaches to the automatic induction of selectional preferences from corpora was the one by Resnik (1996). Resnik (1996) relies on WordNet synsets in order to generate generalized noun clusters. The *selectional preference strength* of a specific verb v in a particular relation is calculated by computing the Kullback-Leibler divergence between the cluster distribution of the verb and the prior cluster distribution.

$$S_{R(v)} = \sum_c \mathrm{p}(c|v) \log \frac{\mathrm{p}(c|v)}{\mathrm{p}(c)} \quad (1)$$

where c stands for a noun cluster, and R stands for a given predicate-argument relation. The *selectional association* of a particular noun cluster is then the contribution of that cluster to the verb's preference strength.

$$A_{R(v,c)} = \frac{\mathrm{p}(c|v) \log \frac{\mathrm{p}(c|v)}{\mathrm{p}(c)}}{S_{R(v)}} \quad (2)$$

The model's generalization relies entirely on WordNet, and there is no generalization among the verbs.

Other researchers have equally relied on WordNet in order to generalize over arguments. Li and Abe (1998) use the principle of Minimum Description Length in order to find a suitable generalization level within the lexical WordNet hierarchy. A same intuition is used by Clark and Weir (2001), but they use hypothesis testing instead to find the appropriate level of generalization. A recent approach that makes use of WordNet (in combination with Bayesian modeling) is the one by Ó Séaghdha and Korhonen (2012).

Most researchers, however, acknowledge the shortcomings of hand-crafted resources, and focus on the acquisition of selectional preferences from corpus data. Rooth et al. (1999) propose an Expectation-Maximization (EM) clustering algorithm for selectional preference acquisition based on a probabilistic latent variable model. The idea is that both predicate v and argument o are generated from a latent variable c, where the latent variables represent clusters of tight verb-argument interactions.

$$\mathrm{p}(v,o) = \sum_{c \in C} \mathrm{p}(c,v,o) = \sum_{c \in C} \mathrm{p}(c)\mathrm{p}(v|c)\mathrm{p}(o|c) \quad (3)$$

The use of latent variables allows the model to generalize to predicate-argument tuples that have not been seen during training. The latent variable distribution – and the probabilities of predicates and argument given the latent variables – are automatically induced from data using EM. We will compare against their model for evaluation purposes.

Erk (2007) and Erk et al. (2010) describe a method that uses corpus-driven distributional similarity metrics for the induction of selectional preferences. The key idea is that a predicate-argument tuple (v,o) is felicitous if the predicate v appears in the training corpus with arguments o' that are similar to o, i.e.

$$S(v,o) = \sum_{o' \in O_v} \frac{wt(v,o')}{Z(v)} \cdot sim(o,o') \quad (4)$$

where O_v represents the set of arguments that have been attested with predicate v, $wt(\cdot)$ represents an appropriate weighting function (such as the frequency of the (v,o') tuple), and Z is a normalization factor. We equally compare to their model for evaluation purposes.

Bergsma et al. (2008) present a discriminative approach to selectional preference acquisition. Positive examples are taken from observed predicate-

argument pairs, while negative examples are constructed from unobserved combinations. An SVM classifier is used to distinguish the positive from the negative instances. The training procedure used in their model is based on an intuition that is similar to ours, although it is implemented using different techniques.

A number of researchers presented models that are based on the framework of topic modeling. Ó Séaghdha (2010) describes three models for selectional preference induction based on Latent Dirichlet Allocation, which model the selectional preference of a predicate and a single argument. Ritter et al. (2010) equally present a selectional preference model based on topic modeling, but they tackle multi-way selectional preferences (of transitive predicates, which take two arguments) instead.

Finally, in previous work (Van de Cruys, 2009) we presented a model for multi-way selectional preference induction based on tensor factorization. Three-way co-occurrences of subjects, verbs, and objects are represented as a three-way tensor (the generalization of a matrix), and a latent factorization model is applied in order to generalize to unseen instances. We will compare our neural network based-approach for multi-way selectional preference acquisition to this tensor-based factorization model.

2.2 Neural networks

In the last few years, neural networks have become increasingly popular in NLP applications. In particular, neural language models (Bengio et al., 2003; Mnih and Hinton, 2007; Collobert and Weston, 2008) have demonstrated impressive performance at the task of language modeling. By incorporating distributed representations for words that model their similarity, neural language models are able to overcome the problem of data sparseness that standard n-gram models are confronted with. Also related to our work is the approach by Tsubaki et al. (2013), who successfully use a neural network to model co-compositionality.

Our model for selectional preference acquisition uses a network architecture that is similar to the abovementioned models. Its training objective is also similar to the ranking-loss training objective proposed by Collobert and Weston (2008), but we present a novel, modified version in order to deal with multi-way selectional preferences.

3 Methodology

3.1 Neural network architecture

Our model computes the score for a predicate i and an argument j as follows. First, the selectional preference tuple (i, j) is represented as the concatenation of the vectors $\mathbf{v}_i$ and $\mathbf{o}_j$, i.e.

$$\mathbf{x} = [\mathbf{v}_i, \mathbf{o}_j] \tag{5}$$

Vectors $\mathbf{v}_i$ and $\mathbf{o}_j$ are extracted from two embedding matrices, $\mathbf{V} \in \mathbb{R}^{N \times I}$ (the predicate matrix, where I represents the number of elements in the predicate vocabulary) and $\mathbf{O} \in \mathbb{R}^{N \times J}$ (the argument matrix, where J represents the number of elements in the argument vocabulary). N is a parameter setting of the model, representing the vector size of the embeddings. Matrices $\mathbf{V}$ and $\mathbf{O}$ will be automatically learned during training.

Vector $\mathbf{x}$ then serves as input vector to our neural network. We use a feed-forward neural network architecture with one hidden layer:

$$\mathbf{a}_1 = f(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \tag{6}$$

$$y = \mathbf{W}_2\mathbf{a}_1 \tag{7}$$

where $\mathbf{x} \in \mathbb{R}^{2N}$ is our input vector, $\mathbf{a}_1 \in \mathbb{R}^H$ represents the activation of the hidden layer with H hidden nodes, $\mathbf{W}_1 \in \mathbb{R}^{H \times 2N}$ and $\mathbf{W}_2 \in \mathbb{R}^{1 \times H}$ respectively represent the first and second layer weights, $\mathbf{b}_1$ represents the first layer's bias, $f(\cdot)$ represents the element-wise activation function tanh, and y is our final selectional preference score. The left-hand picture of figure 1 gives a graphical representation of our standard neural network architecture.

3.2 Training the network

A proper estimation of a neural network's parameters requires a large amount of training data. To be able to use non-annotated corpus data for training, we use the method proposed by Collobert and Weston (2008). The authors present a method for training a neural network language model from unlabeled data by corrupting actual attested n-grams with a random word. They then define a ranking-type cost function, which allows the network to learn to discriminate between good and bad word sequences. We adopt the same method for our selectional preference model as follows.

Let (i, j) be our proper, attested predicate-argument tuple. The goal of our model is to discriminate the correct tuple (i, j) from other, non-attested tuples (i, j'), in which the correct predicate

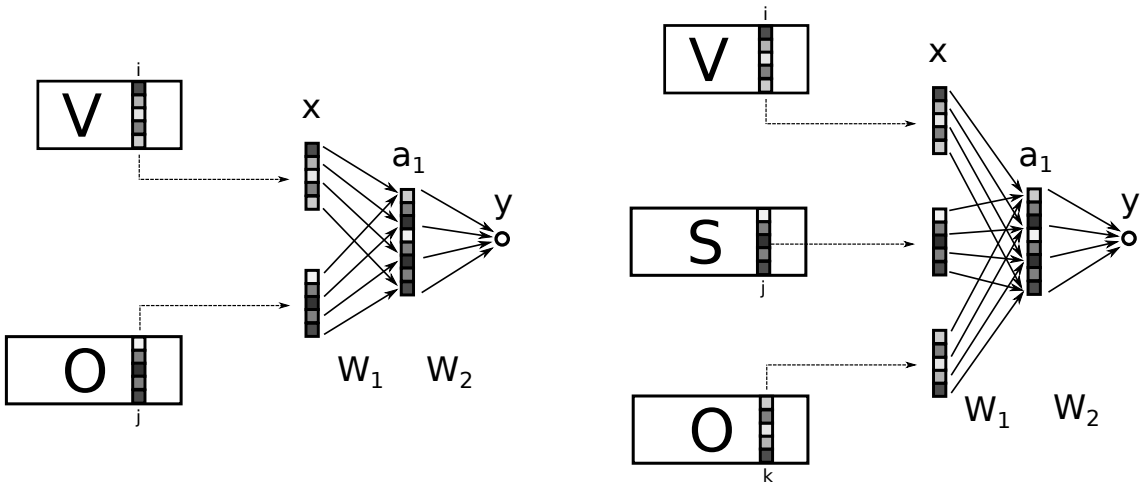


Figure 1: Neural network architectures for selectional preference acquisition. The left-hand picture shows the architecture for two-way selectional preferences, the right-hand picture shows the architecture for three-way selectional preferences. In both cases, vector **x** is constructed from the appropriate predicate and argument vectors from the embedding matrices, and fed forward through the network to yield a preference score y.

j has been replaced with a random predicate j'. We require the score for the correct tuple to be larger than the score for the corrupt tuple by a margin of one. For one tuple (i, j), this corresponds to minimizing the objective function in (8)

$$\sum_{j' \in J} \max(0, 1 - g[(i, j)] + g[(i, j')]) \qquad (8)$$

where J represents the predicate vocabulary, and $g[\cdot]$ represents our neural network scoring function presented in the previous section.

In line with Collobert and Weston (2008), the gradient of the objective function is sampled by randomly picking one corrupt argument j' from the argument vocabulary for each attested predicate-argument tuple (i, j). The derivative of the cost with respect to the model's parameters (weight matrices $\mathbf{W}_1$ and $\mathbf{W}_2$, bias vector $\mathbf{b}_1$, and embedding matrices $\mathbf{V}$ and $\mathbf{O}$) is computed, and the appropriate parameters are updated through backpropagation.

3.3 Multi-way selectional preferences

The model presented in the previous section is only able to deal with two-way selectional preferences. In this section, we present an extension of the model that is able to handle multi-way selectional preferences.[1]

[1]We exemplify the model using three-way selectional preferences for transitive predicates, but the model can be straightforwardly generalized to other multi-way selectional preferences.

In order to model the selectional preference of a transitive verb for its subject and direct object, we start out in a similar fashion to the two-way case. Instead of having only one embedding matrix, we now have two embedding matrices $\mathbf{S} \in \mathbb{R}^{N \times J}$ and $\mathbf{O} \in \mathbb{R}^{N \times K}$, representing the two different argument slots of a transitive predicate. Our input vector can now be represented as

$$\mathbf{x} = (\mathbf{v}_i, \mathbf{s}_j, \mathbf{o}_k) \qquad (9)$$

Note that $\mathbf{x} \in \mathbb{R}^{3N}$ and $\mathbf{W}_1 \in \mathbb{R}^{H \times 3N}$. The rest of our neural network architecture stays exactly the same. The right-hand picture of figure 1 presents a graphical representation.

For the multi-way case, we present an adapted version of the training objective. Given an attested subject-verb-object tuple (i, j, k), the goal of our network is now to discriminate this correct tuple from other, corrupted tuples (i, j, k'), (i, j', k) and (i, j', k'), where the correct arguments have been replaced by random subjects j' and random objects k'. Note that we do not only want the network to learn the infelicity of tuples in which both the subject and object slot are corrupted; we also want our network to learn the infelicity of tuples in which either the subject or object slot is corrupt, while the other slot contains the correct, attested argument. This leads us to the objective function represented in (10).

$$\sum_{k' \in K} \max(0, 1 - g[(i,j,k)] + g[(i,j,k')])$$
$$+ \sum_{j' \in J} \max(0, 1 - g[(i,j,k)] + g[(i,j',k)])$$
$$+ \sum_{\substack{j' \in J \\ k' \in K}} \max(0, 1 - g[(i,j,k)] + g[(i,j',k')]) \quad (10)$$

As in the two-way case, the gradient of the objective function is sampled by randomly picking one corrupted subject j' and one corrupted object k' for each tuple (i, j, k). All of the model's parameters are again updated through backpropagation.

4 Evaluation

4.1 Implementational details

We evaluate our neural network approach to selectional preference acquisition using verb-object tuples for the two-way model, and subject-verb-object tuples for the multi-way model.

Our model has been applied to English, using the UKWaC corpus (Baroni et al., 2009), which covers about 2 billion words of web text. The corpus has been part of speech tagged and lemmatized with Stanford Part-Of-Speech Tagger (Toutanova et al., 2003), and parsed with MaltParser (Nivre et al., 2006), so that dependency tuples could be extracted.

For the two-way model, we select all verbs and objects that appear within a predicate-argument relation with a frequency of at least 50. This gives us a total of about 7K verbs and 30K objects. For the multi-way model, we select the 2K most frequent verbs, together with the 10K most frequent subjects and the 10K most frequent objects (that appear within a transitive frame).

All words are converted to lowercase. We use the lemmatized forms, and only keep those forms that contain alphabetic characters. Furthermore, we require each tuple to appear at least three times in the corpus.

We set N, the size of our embedding matrices, to 50, and H, the number of units in the hidden layer, to 100. Following Huang et al. (2012), we use mini-batch L-BFGS (Liu and Nocedal, 1989) with 1000 pairs of good and corrupt tuples per batch for training, and train for 10 epochs.

4.2 Evaluation Setup

4.2.1 Task

Our models are quantitatively evaluated using a pseudo-disambiguation task (Rooth et al., 1999), which bears some resemblance to our training procedure. The task provides an adequate test of the generalization capabilities of our models. For the two-way case, the task is to judge which object (o or o') is more likely for a particular verb v, where (v, o) is a tuple attested in the corpus, and o' is a direct object randomly drawn from the object vocabulary. The tuple is considered correct if the model prefers the attested tuple (v, o) over (v, o'). For the three-way case, the task is to judge which subject (s or s') and direct object (o or o') are more likely for a particular verb v, where (v, s, o) is the attested tuple, and s' and o' are a random subject and object drawn from their respective vocabularies. The tuple is considered correct if the model prefers the attested tuple (v, s, o) over the alternatives (v, s, o'), (v, s', o), and (v, s', o'). Tables 1 and 2 respectively show a number of examples from the two-way and three-way pseudo-disambiguation task.

v	o	o'
perform	*play*	*geometry*
buy	*wine*	*renaissance*
read	*introduction*	*peanut*

Table 1: Pseudo-disambiguation examples for two-way verb-object tuples

v	s	o	s'	o'
win	*team*	*game*	*diversity*	*egg*
publish	*government*	*document*	*grid*	*priest*
develop	*company*	*software*	*breakfast*	*landlord*

Table 2: Pseudo-disambiguation examples for three-way subject-verb-object tuples

The models are evaluated using 10-fold cross validation. All tuples from our corpus are randomly divided into 10 equal parts. Next, for each fold, 9 parts are used for training, and the remaining part is used for testing. In order to properly test the generalization capability of our models, we make sure that all instances of a particular tuple appear in one part only. This way, we make sure that tuples used for testing are never seen during training.

For the two-way model, our corpus consists of about 70M tuple instances (1.9M types), so in each

fold, about 63M tuple instances are used for training and about 7M (190K types) are used for testing. For the three-way model, our corpus consists of about 5,5M tuple instances (750K types), so in each fold, about 5M tuples are used for training and about 500K (75K types) are used for testing. Note that our training procedure is instance-based, while our evaluation is type-based: during training, the neural network sees a tuple as many times as it appears in the training set, while for testing each individual tuple is only evaluated once.

4.2.2 Comparison models

We compare our neural network model to a number of other models for selectional preference acquisition.

For the two-way case, we compare our model to the EM-based clustering technique presented by Rooth et al. (1999),[2] and to Erk et al.'s (2010) similarity-based model. For Rooth et al.'s model, we set the number of latent factors to 50. Using a larger number of latent factors does not increase performance. For Erk et al.'s model, we create a dependency-based similarity model from the UKWaC corpus using our 30K direct objects as instances and 100K dependency relations as features. The resulting matrix is weighted using pointwise mutual information (Church and Hanks, 1990). Similarity values are computed using cosine. Furthermore, we use a sampling procedure in the testing phase: we sample 5000 predicate-argument pairs for each fold, as testing Erk et al.'s model on the complete test sets proved prohibitively expensive.

For the three-way case, we compare our model to the tensor factorization model we developed in previous work (Van de Cruys, 2009). We set the number of latent factors to 300.[3]

4.3 Results

4.3.1 Two-way model

Table 3 compares the results of our neural network architecture for two-way selectional preferences to the results of Rooth et al.'s (1999) model and Erk et al.'s (2010) model.

[2]Our own implementation of Rooth et al.'s (1999) algorithm is based on non-negative matrix factorization (Lee and Seung, 2000). Non-negative matrix factorization with Kullback-Leibler divergence has been shown to minimize the same objective function as EM (Li and Ding, 2006).

[3]The best scoring model presented by Van de Cruys (2009) also uses 300 latent factors; using more factors does not improve the results.

model	accuracy ($\mu \pm \sigma$)
Rooth et al. (1999)	.720 $\pm$ .002
Erk et al. (2010)	**.887** $\pm$ .004
2-way neural network	.880 $\pm$ .001

Table 3: Comparison of model results for two-way selectional preference acquisition – mean accuracy and standard deviations of 10-fold cross-validation results

The results indicate that our neural network approach outperforms Rooth et al.'s (1999) method by a large margin (16%). Clearly, the neural network architecture is able to model selectional preferences more profoundly than Rooth et al.'s latent variable approach. The difference between the models is highly statistically significant (paired t-test, $p < .01$), as the standard deviations already indicate.

Erk et al.'s model reaches a slightly better score than our model, and this result is also statistically significant (paired t-test, $p < .01$). However, Erk et al.'s model does not provide full coverage, whereas the other two models are able to compute scores for all pairs in the test set. In addition, Erk et al.'s model is much more expensive to compute. Our model computes selectional preference scores for the test set in a matter of seconds, whereas for Erk et al.'s model, we ended up sampling from the test set, as computing preference values for the complete test set proved prohibitively expensive.

4.3.2 Three-way model

Table 4 compares the results of our neural network architecture for three-way selectional preference acquisition to the results of the tensor-based factorization method (Van de Cruys, 2009).

model	accuracy ($\mu \pm \sigma$)
Van de Cruys (2009)	.874 $\pm$ .001
3-way neural network	**.889** $\pm$ .001

Table 4: Comparison of model results for three-way selectional preference acquisition – mean accuracy and standard deviations of 10-fold cross-validation results

The results indicate that the neural network approach slightly outperforms the tensor-based factorization method. Again the model difference is sta-

tistically significant (paired t-test, $p < 0.01$). Using our adapted training objective, the neural network is clearly able to learn a rich model of three-way selectional preferences, reaching state of the art performance.

4.4 Examples

We conclude our results section by briefly presenting a number of examples that illustrate the kind of semantics present in our models. Similar to neural language models, the predicate and argument embedding matrices of our neural network contain distributed word representations, that capture the similarity of predicates and arguments to other words.

Tables 5 and 6 contain a number of nearest neighbour similarity examples for predicate and arguments from our two-way neural network model. The nearest neighbours were calculated using standard cosine similarity.

DRINK	PROGRAM	INTERVIEW	FLOOD
SIP	RECOMPILE	RECRUIT	INUNDATE
BREW	UNDELETE	PERSUADE	RAVAGE
MINCE	CODE	INSTRUCT	SUBMERGE
FRY	IMPORT	PESTER	COLONIZE

Table 5: Nearest neighbours of 4 verbs, calculated using the distributed word representations of embedding matrix **V** from our two-way neural network model

Table 5 indicates that the network is effectively able to capture a semantics for verbs. The first column – verbs similar to DRINK – all have to do with food consumption. The second column contains verbs related to computer programming. The third column is related to human communication; and the fourth column seems to illustrate the network's comprehension of FLOOD having to do with invasion and water.

PAPER	RASPBERRY	SECRETARY	DESIGNER
BOOK	COURGETTE	PRESIDENT	PLANNER
JOURNAL	LATTE	MANAGER	PAINTER
ARTICLE	LEMONADE	POLICE	SPECIALIST
CODE	OATMEAL	EDITOR	SPEAKER

Table 6: Nearest neighbours of 4 direct objects, calculated using the distributed word representations of embedding matrix **O** from our two way neural network model

Similarly, table 6 shows the network's ability to capture the meaning of nouns that appear as direct objects to the verbs. Column one contains things that can be read. Column two contains things that can be consumed. Column three seems to hint at supervising professions, while column four seems to capture creative professions.

A similar kind of semantics is present in the embedding matrices of the three-way neural network model. Tables 7, 8, and 9 again illustrate this using word similarity calculations.

SEARCH	DIMINISH	CONFIGURE	PROSECUTE
CLICK	LESSEN	AUTOMATE	CRITICISE
BROWSE	DISTORT	SCROLL	URGE
SCROLL	HEIGHTEN	PROGRAM	DEPLORE
UPLOAD	DEGRADE	INSTALL	CONDEMN

Table 7: Nearest neighbours of 4 verbs, calculated using the distributed word representations of embedding matrix **V** from our three-way neural network model

Table 7 shows the network's verb semantics for the three-way case. The first column is related to internet usage, the second column contains verbs of scalar change, column three is again related to computer usage, and column four seems to capture 'mending' verbs.

FLOWER	COLLEGE	PRESIDENT	SONG
FISH	UNIVERSITY	BUSH	FILM
BIRD	INSTITUTE	BLAIR	ALBUM
SUN	DEPARTMENT	MP	PLAY
TREE	CENTRE	CHAIRMAN	MUSIC

Table 8: Nearest neighbours of 4 subjects, calculated using the distributed word representations of embedding matrix **S** from our three way neural network model

Table 8 illustrates the semantics for the subject slot of our three-way model. The first column captures nature terms, the second column contains university-related terms, the third column contains politicians/government terms, and the fourth column contains art expressions.

Finally, table 9 demonstrates the semantics of our three-way model's object slot. Column one generally contains housing terms, column two contains various locations, column three contains dining occasions, and column four contains textual expressions.

WALL	PARK	LUNCH	THESIS
FLOOR	STUDIO	DINNER	QUESTIONNAIRE
CEILING	VILLAGE	MEAL	DISSERTATION
ROOF	HALL	BUFFET	PERIODICAL
METRE	MUSEUM	BREAKFAST	DISCOURSE

Table 9: Nearest neighbours of 4 direct objects, calculated using the distributed word representations of embedding matrix **O** from our three way neural network model

Note that the embeddings for the subject and the object slot is different, although they mostly contain the same words. This allows the model to capture specific semantic characteristics for words given their argument position. *Virus*, for example, is in subject position more similar to active words like *animal*, whereas in object position, it is more similar to passive words like *cell*, *device*. Similarly, *mouse* in subject position tends to be similar to words like *animal*, *rat* whereas in object position it is similar to words like *web*, *browser*.

These examples, although anecdotal, illustrate that our neural network model is able to capture a rich semantics for predicates and arguments, which subsequently allows the network to make accurate predictions with regard to selectional preference.

5 Conclusion and future work

In this paper, we presented a neural network approach to the acquisition of selectional preferences. Inspired by recent work on neural language models, we proposed a neural network model that learns to discriminate between felicitous and infelicitous arguments for a particular predicate. The model is entirely unsupervised, as preferences are learned from unannotated corpus data. Positive training instances are constructed from attested corpus data, while negative instances are constructed from randomly corrupted instances. Using designated network architectures, we are able to handle standard two-way selectional preferences as well as multi-way selectional preferences. A quantitative evaluation on a pseudo-disambiguation task shows that our models achieve state of the art performance. The results for our two-way neural network are on a par with Erk et al.'s (2010) similarity-based approach, while our three-way neural network slightly outperforms the tensor-based factorization model (Van de Cruys, 2009) for multi-way selectional preference induction.

We conclude with a number of issues for future work. First of all, we would like to investigate how our neural network approach might be improved by incorporating information from other sources. In particular, we think of initializing our embedding matrices with distributed representations that come from a large-scale neural language model (Mikolov et al., 2013). We also want to further investigate the advantages and disadvantages of having different embedding matrices for different argument positions in our multi-way neural network. In our results section, we demonstrated that such an approach allows for more flexibility, but it also adds a certain level of redundancy. We want to investigate the benefit of our approach, compared to a model that shares the distributed word representation among different argument positions. Finally, we want to investigate more advanced neural network architectures for the acquisition of selectional preferences. In particular, neural tensor networks (Yu et al., 2013) have recently demonstrated impressive results in related fields like speech recognition, and might provide the necessary machinery to model multi-way selectional preferences in a more profound way.

References

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Shane Bergsma, Dekang Lin, and Randy Goebel. 2008. Discriminative learning of selectional preference from unlabeled text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 59–68. Association for Computational Linguistics.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information & lexicography. *Computational Linguistics*, 16(1):22–29.

Stephen Clark and David Weir. 2001. Class-based probability estimation using a semantic hierarchy. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 95–102. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep

neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

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

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 216–223, Prague, Czech Republic, June. Association for Computational Linguistics.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13*, pages 556–562.

Hang Li and Naoki Abe. 1998. Generalizing case frames using a thesaurus and the MDL principle. *Computational linguistics*, 24(2):217–244.

Tao Li and Chris Ding. 2006. The relationships among various nonnegative matrix factorization methods for clustering. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 362–371. IEEE.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Diana McCarthy and John Carroll. 2003. Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654.

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

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*, pages 2216–2219.

Diarmuid Ó Séaghdha and Anna Korhonen. 2012. Modelling selectional preferences in a lexical hierarchy. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 170–179. Association for Computational Linguistics.

Diarmuid Ó Séaghdha. 2010. Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 435–444. Association for Computational Linguistics.

Sebastian Padó, Ulrike Padó, and Katrin Erk. 2007. Flexible, corpus-based modelling of human plausibility judgements. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 400–409, Prague, Czech Republic, June. Association for Computational Linguistics.

Philip Resnik. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, 61:127–159, November.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Uppsala, Sweden, July. Association for Computational Linguistics.

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via em-based clustering. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 104–111. Association for Computational Linguistics.

Ekaterina Shutova, Simone Teufel, and Anna Korhonen. 2013. Statistical metaphor processing. *Computational Linguistics*, 39(2):301–353.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259.

Masashi Tsubaki, Kevin Duh, Masashi Shimbo, and Yuji Matsumoto. 2013. Modeling and learning semantic co-compositionality through prototype projections and neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 130–140, Seattle, Washington, USA, October. Association for Computational Linguistics.

Tim Van de Cruys. 2009. A non-negative tensor factorization model for selectional preference induction.

In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 83–90, Athens, Greece, March. Association for Computational Linguistics.

Dong Yu, Li Deng, and Frank Seide. 2013. The deep tensor neural network with applications to large vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(2):388–396.

Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics

Douwe Kiela*
University of Cambridge
Computer Laboratory
douwe.kiela@cl.cam.ac.uk

Léon Bottou
Microsoft Research
New York
leon@bottou.org

Abstract

We construct multi-modal concept representations by concatenating a skip-gram linguistic representation vector with a visual concept representation vector computed using the feature extraction layers of a deep convolutional neural network (CNN) trained on a large labeled object recognition dataset. This transfer learning approach brings a clear performance gain over features based on the traditional bag-of-visual-word approach. Experimental results are reported on the WordSim353 and MEN semantic relatedness evaluation tasks. We use visual features computed using either ImageNet or ESP Game images.

1 Introduction

Recent works have shown that multi-modal semantic representation models outperform unimodal linguistic models on a variety of tasks, including modeling semantic relatedness and predicting compositionality (Feng and Lapata, 2010; Leong and Mihalcea, 2011; Bruni et al., 2012; Roller and Schulte im Walde, 2013; Kiela et al., 2014). These results were obtained by combining linguistic feature representations with robust visual features extracted from a set of images associated with the concept in question. This extraction of visual features usually follows the popular computer vision approach consisting of computing local features, such as SIFT features (Lowe, 1999), and aggregating them as bags of visual words (Sivic and Zisserman, 2003).

Meanwhile, deep transfer learning techniques have gained considerable attention in the computer vision community. First, a deep convolutional neural network (CNN) is trained on a large labeled dataset (Krizhevsky et al., 2012). The convolutional layers are then used as mid-level feature extractors on a variety of computer vision tasks (Oquab et al., 2014; Girshick et al., 2013; Zeiler and Fergus, 2013; Donahue et al., 2014). Although transferring convolutional network features is not a new idea (Driancourt and Bottou, 1990), the simultaneous availability of large datasets and cheap GPU co-processors has contributed to the achievement of considerable performance gains on a variety computer vision benchmarks: *"SIFT and HOG descriptors produced big performance gains a decade ago, and now deep convolutional features are providing a similar breakthrough"* (Razavian et al., 2014).

This work reports on results obtained by using CNN-extracted features in multi-modal semantic representation models. These results are interesting in several respects. First, these superior features provide the opportunity to increase the performance gap achieved by augmenting linguistic features with multi-modal features. Second, this increased performance confirms that the multimodal performance improvement results from the information contained in the images and not the information used to select which images to use to represent a concept. Third, our evaluation reveals an intriguing property of the CNN-extracted features. Finally, since we use the skip-gram approach of Mikolov et al. (2013) to generate our linguistic features, we believe that this work represents the first approach to multimodal distributional semantics that exclusively relies on deep learning for both its linguistic and visual components.

2 Related work

2.1 Multi-Modal Distributional Semantics

Multi-modal models are motivated by parallels with human concept acquisition. Standard se-

* This work was carried out while Douwe Kiela was an intern at Microsoft Research, New York.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 36–45,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

mantic space models extract meanings solely from linguistic data, even though we know that human semantic knowledge relies heavily on perceptual information (Louwerse, 2011). That is, there exists substantial evidence that many concepts are *grounded* in the perceptual system (Barsalou, 2008). One way to do this grounding in the context of distributional semantics is to obtain representations that combine information from linguistic corpora with information from another modality, obtained from e.g. property norming experiments (Silberer and Lapata, 2012; Roller and Schulte im Walde, 2013) or from processing and extracting features from images (Feng and Lapata, 2010; Leong and Mihalcea, 2011; Bruni et al., 2012). This approach has met with quite some success (Bruni et al., 2014).

2.2 Multi-modal Deep Learning

Other examples that apply multi-modal deep learning use restricted Boltzmann machines (Srivastava and Salakhutdinov, 2012; Feng et al., 2013), auto-encoders (Wu et al., 2013) or recursive neural networks (Socher et al., 2014). Multimodal models with deep learning components have also successfully been employed in cross-modal tasks (Lazaridou et al., 2014). Work that is closely related in spirit to ours is by Silberer and Lapata (2014). They use a stacked auto-encoder to learn combined embeddings of textual and visual input. Their visual inputs consist of vectors of visual attributes obtained from learning SVM classifiers on attribute prediction tasks. In contrast, our work keeps the modalities separate and follows the standard multi-modal approach of concatenating linguistic and visual representations in a single semantic space model. This has the advantage that it allows for separate data sources for the individual modalities. We also learn visual representations directly from the images (i.e., we apply deep learning directly to the images), as opposed to taking a higher-level representation as a starting point. Frome et al. (2013) jointly learn multimodal representations as well, but apply them to a visual object recognition task instead of concept meaning.

2.3 Deep Convolutional Neural Networks

A flurry of recent results indicates that image descriptors extracted from deep convolutional neural networks (CNNs) are very powerful and consistently outperform highly tuned state-of-the-art systems on a variety of visual recognition tasks (Razavian et al., 2014). Embeddings from state-of-the-art CNNs (such as Krizhevsky et al. (2012)) have been applied successfully to a number of problems in computer vision (Girshick et al., 2013; Zeiler and Fergus, 2013; Donahue et al., 2014). This contribution follows the approach described by Oquab et al. (2014): they train a CNN on 1512 ImageNet synsets (Deng et al., 2009), use the first seven layers of the trained network as feature extractors on the Pascal VOC dataset, and achieve state-of-the-art performance on the Pascal VOC classification task.

3 Improving Multi-Modal Representations

Figure 1 illustrates how our system computes multi-modal semantic representations.

3.1 Perceptual Representations

The perceptual component of standard multi-modal models that rely on visual data is often an instance of the bag-of-visual-words (BOVW) representation (Sivic and Zisserman, 2003). This approach takes a collection of images associated with words or tags representing the concept in question. For each image, keypoints are laid out as a dense grid. Each keypoint is represented by a vector of robust local visual features such as SIFT (Lowe, 1999), SURF (Bay et al., 2008) and HOG (Dalal and Triggs, 2005), as well as pyramidal variants of these descriptors such as PHOW (Bosch et al., 2007). These descriptors are subsequently clustered into a discrete set of "visual words" using a standard clustering algorithm like k-means and quantized into vector representations by comparing the local descriptors with the cluster centroids. Visual representations are obtained by taking the average of the BOVW vectors for the images that correspond to a given word. We use BOVW as a baseline.

Our approach similarly makes use of a collection of images associated with words or tags representing a particular concept. Each image is processed by the first seven layers of the convolutional network defined by Krizhevsky et al. (2012) and adapted by Oquab et al. (2014)[1]. This network takes 224×224 pixel RGB images and applies five successive convolutional layers followed by three fully connected layers. Its eighth and last

[1] http://www.di.ens.fr/willow/research/cnn/

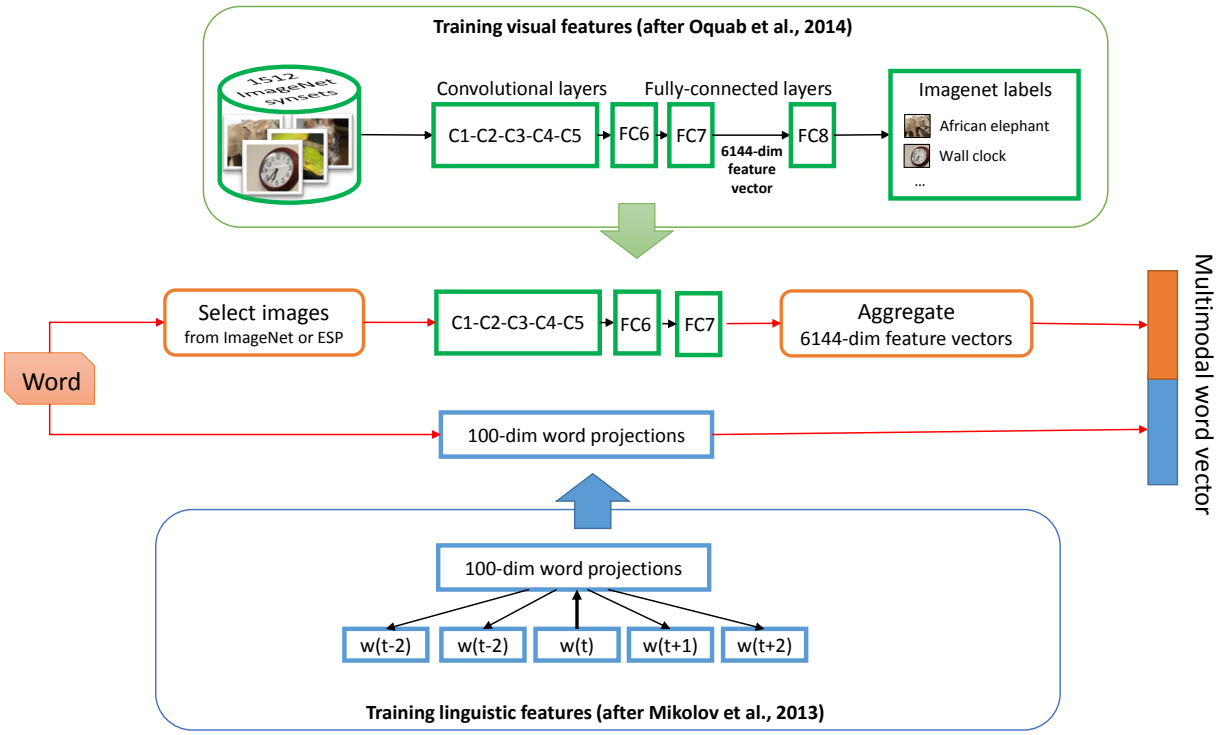


Figure 1: Computing word feature vectors.

layer produces a vector of 1512 scores associated with 1000 categories of the ILSVRC-2012 challenge and the 512 additional categories selected by Oquab et al. (2014). This network was trained using about 1.6 million ImageNet images associated with these 1512 categories. We then freeze the trained parameters, chop the last network layer, and use the remaining seventh layer as a filter to compute a 6144-dimensional feature vector on arbitrary 224×224 input images.

We consider two ways to aggregate the feature vectors representing each image.

1. The first method (**CNN-Mean**) simply computes the average of all feature vectors.

2. The second method (**CNN-Max**) computes the component-wise maximum of all feature vectors. This approach makes sense because the feature vectors extracted from this particular network are quite sparse (about 22% non-zero coefficients) and can be interpreted as bags of visual properties.

3.2 Linguistic representations

For our linguistic representations we extract 100-dimensional continuous vector representations using the log-linear skip-gram model of Mikolov et al. (2013) trained on a corpus consisting of the 400M word Text8 corpus of Wikipedia text[2] together with the 100M word British National Corpus (Leech et al., 1994). We also experimented with dependency-based skip-grams (Levy and Goldberg, 2014) but this did not improve results. The skip-gram model learns high quality semantic representations based on the distributional properties of words in text, and outperforms standard distributional models on a variety of semantic similarity and relatedness tasks. However we note that Bruni et al. (2014) have recently reported an even better performance for their linguistic component using a standard distributional model, although this may have been tuned to the task.

3.3 Multi-modal Representations

Following Bruni et al. (2014), we construct multi-modal semantic representations by concatenating the centered and L_2-normalized linguistic and perceptual feature vectors $\vec{v}_{ling}$ and $\vec{v}_{vis}$,

$$\vec{v}_{concept} = \alpha \times \vec{v}_{ling} \,||\, (1-\alpha) \times \vec{v}_{vis}\,, \quad (1)$$

where $||$ denotes the concatenation operator and α is an optional tuning parameter.

[2] http://mattmahoney.net/dc/textdata.html

Figure 2: Examples of *dog* in the ESP Game dataset.

Figure 3: Examples of *golden retriever* in ImageNet.

4 Experimental Setup

We carried out experiments using visual representations computed using two canonical image datasets. The resulting multi-modal concept representations were evaluated using two well-known semantic relatedness datasets.

4.1 Visual Data

We carried out experiments using two distinct sources of images to compute the visual representations.

The **ImageNet** dataset (Deng et al., 2009) is a large-scale ontology of images organized according to the hierarchy of WordNet (Fellbaum, 1999). The dataset was constructed by manually re-labelling candidate images collected using web searches for each WordNet synset. The images tend to be of high quality with the designated object roughly centered in the image. Our copy of ImageNet contains about 12.5 million images organized in 22K synsets. This implies that ImageNet covers only a small fraction of the existing 117K WordNet synsets.

The **ESP Game** dataset (Von Ahn and Dabbish, 2004) was famously collected as a "game with a purpose", in which two players must independently and rapidly agree on a correct word label for randomly selected images. Once a word label has been used sufficiently frequently for a given image, that word is added to the image's tags. This dataset contains 100K images, but with every image having on average 14 tags, that amounts to a coverage of 20,515 words. Since players are encouraged to produce as many terms per image, the dataset's increased coverage is at the expense of accuracy in the word-to-image mapping: a dog in a field with a house in the background might be a *golden retriever* in ImageNet and could have tags *dog, golden retriever, grass, field, house, door* in the ESP Dataset. In other words, images in the ESP dataset do not make a distinction between objects in the foreground and in the background, or between the relative size of the objects (tags for images are provided in a random order, so the top tag is not necessarily the best one).

Figures 2 and 3 show typical examples of images belonging to these datasets. Both datasets have attractive properties. On the one hand, ImageNet has higher quality images with better labels. On the other hand, the ESP dataset has an interesting coverage because the MEN task (see section 4.4) was specifically designed to be covered by the ESP dataset.

4.2 Image Selection

Since ImageNet follows the WordNet hierarchy, we would have to include almost all images in the dataset to obtain representations for high-level concepts such as *entity*, *object* and *animal*. Doing so is both computationally expensive and unlikely to improve the results. For this reason, we randomly sample up to N distinct images from the subtree associated with each concept. When this returns less than N images, we attempt to increase coverage by sampling images from the subtree of the concept's hypernym instead. In order to allow for a fair comparison, we apply the same method of sampling up to N on the ESP Game dataset. In all following experiments, $N = 1.000$. We used the WordNet lemmatizer from NLTK (Bird et al., 2009) to lemmatize tags and concept words so as to further improve the dataset's coverage.

4.3 Image Processing

The ImageNet images were preprocessed as described by (Krizhevsky et al., 2012). The largest centered square contained in each image is resam-

pled to form a 256×256 image. The CNN input is then formed by cropping 16 pixels off each border and subtracting 128 to the image components. The ESP Game images were preprocessed slightly differently because we do not expect the objects to be centered. Each image was rescaled to fit inside a 224×224 rectangle. The CNN input is then formed by centering this image into the 224×224 input field, subtracting 128 to the image components, and zero padding.

The BOVW features were obtained by computing DSIFT descriptors using VLFeat (Vedaldi and Fulkerson, 2008). These descriptors were subsequently clustered using mini-batch k-means (Sculley, 2010) with 100 clusters. Each image is then represented by a bag of clusters (visual words) quantized as a 100-dimensional feature vector. These vectors were then combined into visual concept representations by taking their mean.

4.4 Evaluation

We evaluate our multi-modal word representations using two semantic relatedness datasets widely used in distributional semantics (Agirre et al., 2009; Feng and Lapata, 2010; Bruni et al., 2012; Kiela and Clark, 2014; Bruni et al., 2014).

WordSim353 (Finkelstein et al., 2001) is a selection of 353 concept pairs with a similarity rating provided by human annotators. Since this is probably the most widely used evaluation dataset for distributional semantics, we include it for comparison with other approaches. WordSim353 has some known idiosyncracies: it includes named entities, such as *OPEC*, *Arafat*, and *Maradona*, as well as abstract words, such as *antecedent* and *credibility*, for which it may be hard to find corresponding images. Multi-modal representations are often evaluated on an unspecified subset of WordSim353 (Feng and Lapata, 2010; Bruni et al., 2012; Bruni et al., 2014), making it impossible to compare the reported scores. In this work, we report scores on the full WordSim353 dataset (**W353**) by setting the visual vector $\vec{v}_{vis}$ to zero for concepts without images. We also report scores on the subset (**W353-Relevant**) of pairs for which both concepts have both ImageNet and ESP Game images using the aforementioned selection procedure.

MEN (Bruni et al., 2012) was in part designed to alleviate the WordSim353 problems. It was constructed in such a way that only frequent words with at least 50 images in the ESP Game dataset were included in the evaluation pairs. The MEN dataset has been found to mirror the aggregate score over a variety of tasks and similarity datasets (Kiela and Clark, 2014). It is also much larger, with 3000 words pairs consisting of 751 individual words. Although MEN was constructed so as to have at least a minimum amount of images available in the ESP Game dataset for each concept, this is not the case for ImageNet. Hence, similarly to WordSim353, we also evaluate on a subset (**MEN-Relevant**) for which images are available in both datasets.

We evaluate the models in terms of their Spearman ρ correlation with the human relatedness ratings. The similarity between the representations associated with a pair of words is calculated using the cosine similarity:

$$\cos(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\| \, \|v_2\|} \tag{2}$$

5 Results

We evaluate on the two semantic relatedness datasets using solely linguistic, solely visual and multi-modal representations. In the case of MEN-Relevant and W353-Relevant, we report scores for BOVW, CNN-Mean and CNN-Max visual representations. For all datasets we report the scores obtained by BOVW, CNN-Mean and CNN-Max multi-modal representations. Since we have full coverage with the ESP Game dataset on MEN, we are able to report visual representation scores for the entire dataset as well. The results can be seen in Table 1.

There are a number of questions to ask. First of all, do CNNs yield better visual representations? Second, do CNNs yield better multi-modal representations? And third, is there a difference between the high-quality low-coverage ImageNet and the low-quality higher-coverage ESP Game dataset representations?

5.1 Visual Representations

In all cases, CNN-generated visual representations perform better or as good as BOVW representations (we report results for BOVW-Mean, which performs slightly better than taking the element-wise maximum). This confirms the motivation outlined in the introduction: by applying state-of-the-art approaches from computer vision to multi-modal semantics, we obtain a signficant perfor-

Dataset	Linguistic	Visual			Multi-modal		
		BOVW	CNN-Mean	CNN-Max	BOVW	CNN-Mean	CNN-Max
ImageNet visual features							
MEN	0.64	-	-	-	0.64	**0.70**	0.67
MEN-Relevant	0.62	0.40	0.64	0.63	0.64	**0.72**	0.71
W353	0.57	-	-	-	0.58	0.59	**0.60**
W353-Relevant	0.51	0.30	0.32	0.30	0.55	0.56	**0.57**
ESP game visual features							
MEN	0.64	0.17	0.51	0.20	0.64	**0.71**	0.65
MEN-Relevant	0.62	0.35	0.58	0.57	0.63	0.69	**0.70**
W353	0.57	-	-	-	0.58	0.59	**0.60**
W353-Relevant	0.51	0.38	0.44	0.56	0.52	0.55	**0.61**

Table 1: Results (see sections 4 and 5).

mance increase over standard multi-modal models.

5.2 Multi-modal Representations

Higher-quality perceptual input leads to better-performing multi-modal representations. In all cases multi-modal models with CNNs outperform multi-modal models with BOVW, occasionally by quite a margin. In all cases, multi-modal representations outperform purely linguistic vectors that were obtained using a state-of-the-art system. This re-affirms the importance of multi-modal representations for distributional semantics.

5.3 The Contribution of Images

Since the ESP Game images come with a multitude of word labels, one could question whether a performance increase of multi-modal models based on that dataset comes from the images themselves, or from overlapping word labels. It might also be possible that similar concepts are more likely to occur in the same image, which encodes relatedness information without necessarily taking the image data itself into account. In short, it is a natural question to ask whether the performance gain is due to image data or due to word label associations? We conclusively show that the image data matters in two ways: (a) using a different dataset (ImageNet) also results in a performance boost, and (b) using higher-quality image features on the ESP game images increases the performance boost without changing the association between word labels.

5.4 Image Datasets

It is important to ask whether the source image dataset has a large impact on performance. Although the scores for the visual representation in some cases differ, performance of multi-modal representations remains close for both image datasets. This implies that our method is robust over different datasets. It also suggests that it is beneficial to train on high-quality datasets like ImageNet and to subsequently generate embeddings for other sets of images like the ESP Game dataset that are more noisy but have better coverage. The results show the benefit of transfering convolutional network features, corroborating recent results in computer vision.

5.5 Semantic Similarity/Relatedness Datasets

There is an interesting discrepancy between the two types of network with respect to dataset performance: CNN-Mean multi-modal models tend to perform best on MEN and MEN-Relevant, while CNN-Max multi-modal models perform better on W353 and W353-Relevant. There also appears to be some interplay between the source corpus, the evaluation dataset and the best performing CNN: the performance leap on W353-

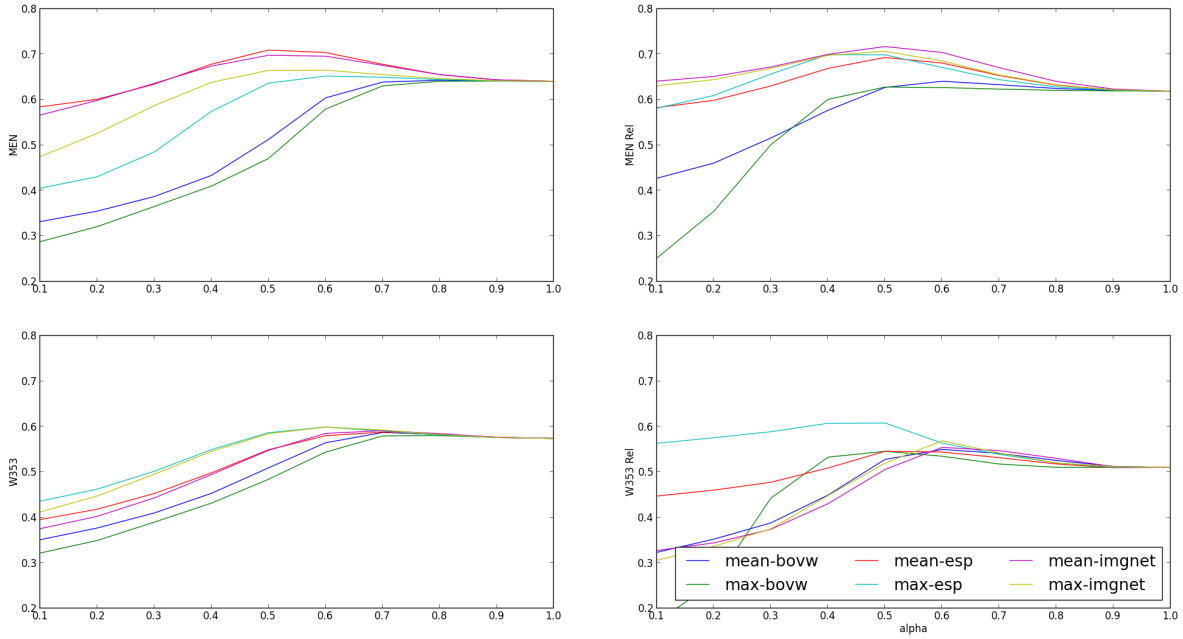


Figure 4: Varying the α parameter for MEN, MEN-Relevant, WordSim353 and WordSim353-Relevant, respectively.

Relevant for CNN-Max is much larger using ESP Game images than with ImageNet images.

We speculate that this is because CNN-Max performs better than CNN-Mean on a somewhat different type of similarity. It has been noted (Agirre et al., 2009) that WordSim353 captures both similarity (as in *tiger-cat*, with a score of 7.35) as well as relatedness (as in *Maradona-football*, with a score of 8.62). MEN, however, is explicitly designed to capture semantic relatedness only (Bruni et al., 2012). CNN-Max using sparse feature vectors means that we treat the dominant components as definitive of the concept class, which is more suited to similarity. CNN-Mean averages over all the feature components, and as such might be more suited to relatedness. We conjecture that the performance increase on WordSim353 is due to increased performance on the similarity subset of that dataset.

5.6 Tuning

The concatenation scheme in Equation 1 allows for a tuning parameter α to weight the relative contribution of the respective modalities. Previous work on MEN has found that the optimal parameter for that dataset is close to 0.5 (Bruni et al., 2014). We have found that this is indeed the case. On WordSim353, however, we have found the parameter for optimal performance to be shifted to the right, meaning that optimal performance is achieved when we include less of the visual input compared to the linguistic input. Figure 4 shows what happens when we vary alpha over the four datasets. There are a number of observations to be made here.

First of all, we can see that the performance peak for the MEN datastes is much higher than for the WordSim353 ones, and that its peak is relatively higher as well. This indicates that MEN is in a sense a more balanced dataset. There are two possible explanations: as indicated earlier, WordSim353 contains slightly idiosyncratic word pairs which may have a detrimental effect on performance; or, WordSim353 was not constructed with multi-modal semantics in mind, and contains a substantial amount of abstract words that would not benefit at all from including visual information.

Due to the nature of the datasets and the tasks at hand, it is arguably much more important that CNNs beat standard bag-of-visual-words representations on MEN than on W353, and indeed we see that there exists no α for which BOVW would beat any of the CNN networks.

6 Error Analysis

Table 2 shows the top 5 best and top 5 worst scoring word pairs for the two datasets using CNN-

W353-Relevant

ImageNet				ESP Game			
word1	word2	system score	gold standard	word1	word2	system score	gold standard
tiger	tiger	1.00	1.00	tiger	tiger	1.00	1.00
man	governor	0.53	0.53	man	governor	0.53	0.53
stock	phone	0.15	0.16	stock	phone	0.15	0.16
football	tennis	0.68	0.66	football	tennis	0.68	0.66
man	woman	0.85	0.83	man	woman	0.85	0.83
cell	phone	0.27	0.78	law	lawyer	0.33	0.84
discovery	space	0.10	0.63	monk	slave	0.58	0.09
closet	clothes	0.22	0.80	gem	jewel	0.41	0.90
king	queen	0.26	0.86	stock	market	0.33	0.81
wood	forest	0.13	0.77	planet	space	0.32	0.79

MEN-Relevant

ImageNet				ESP Game			
word1	word2	system score	gold standard	word1	word2	system score	gold standard
beef	potatoes	0.35	0.35	beef	potatoes	0.35	0.35
art	work	0.35	0.35	art	work	0.35	0.35
grass	stop	0.06	0.06	grass	stop	0.06	0.06
shade	tree	0.45	0.45	shade	tree	0.45	0.45
blonde	rock	0.07	0.07	blonde	rock	0.07	0.07
bread	potatoes	0.88	0.34	bread	dessert	0.78	0.24
fruit	potatoes	0.80	0.26	jacket	shirt	0.89	0.34
dessert	sandwich	0.76	0.23	fruit	nuts	0.88	0.33
pepper	tomato	0.79	0.27	dinner	lunch	0.93	0.37
dessert	tomato	0.66	0.14	dessert	soup	0.81	0.23

Table 2: The top 5 best and top 5 worst scoring pairs with respect to the gold standard.

Mean multi-modal vectors. The most accurate pairs are consistently the same across the two image datasets. There are some clear differences between the least accurate pairs, however. The MEN words *potatoes* and *tomato* probably have low quality ImageNet-derived representations, because they occur often in the bottom pairs for that dataset. The MEN words *dessert*, *bread* and *fruit* occur in the bottom 5 for both image datasets, which implies that their linguistic representations are probably not very good. For WordSim353, the bottom pairs on ImageNet could be said to be similarity mistakes; while the ESP Game dataset contains more relatedness mistakes (*king* and *queen* would evaluate similarity, while *stock* and *market* would evaluate relatedness). It is difficult to say anything conclusive about this discrepancy, but it is clearly a direction for future research.

7 Image embeddings

To facilitate further research on image embeddings and multi-modal semantics, we publicly release embeddings for all the image labels occurring in the ESP Game dataset. Please see the following web page: `http://www.cl.cam.ac.uk/~dk427/imgembed.html`

8 Conclusion

We presented a novel approach to improving multi-modal representations using deep convolutional neural network-extracted features. We reported high results on two well-known and widely-used semantic relatedness benchmarks, with increased performance both in the separate visual representations and in the combined multi-modal representations. Our results indicate that such multi-modal representations outperform both linguistic and standard bag-of-visual-words multi-modal representations. We have shown that our approach is robust and that CNN-extracted features from separate image datasets can succesfully be applied to semantic relatedness.

In addition to improving multi-modal representations, we have shown that the source of this improvement is due to image data and is not simply a result of word label associations. We have shown this by obtaining performance improvements on two different image datasets, and by obtaining

higher performance with higher-quality image features on the ESP game images, without changing the association between word labels.

In future work, we will investigate whether our system can be further improved by including concreteness information or a substitute metric such as image dispersion, as has been suggested by other work on multi-modal semantics (Kiela et al., 2014). Furthermore, a logical next step to increase performance would be to jointly learn multi-modal representations or to learn weighting parameters. Another interesting possibility would be to examine multi-modal distributional compositional semantics, where multi-modal representations are composed to obtain phrasal representations.

Acknowledgments

We would like to thank Maxime Oquab for providing the feature extraction code.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 19–27, Boulder, Colorado.

Lawrence W. Barsalou. 2008. Grounded cognition. *Annual Review of Psychology*, 59:617—845.

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. 2008. SURF: Speeded Up Robust Features. In *Computer Vision and Image Understanding (CVIU)*, volume 110, pages 346–359.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Anna Bosch, Andrew Zisserman, and Xavier Munoz. 2007. Image classification using random forests and ferns. In *Proceedings of ICCV*.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

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

Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *International Conference on Machine Learning (ICML 2014)*.

Xavier Driancourt and Léon Bottou. 1990. TDNN-extracted features. In *Proceedings of Neuro Nimes 90*, Nimes, France. EC2.

Christiane Fellbaum. 1999. *WordNet*. Wiley Online Library.

Yansong Feng and Mirella Lapata. 2010. Visual information in semantic representation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 91–99. Association for Computational Linguistics.

Fangxiang Feng, Ruifan Li, and Xiaojie Wang. 2013. Constructing hierarchical image-tags bimodal representations for word tags alternative choice. *CoRR*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Andrea Frome, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, MarcÁurelio Ranzato, and Tomas Mikolov. 2013. DeViSE: A Deep Visual-Semantic Embedding Model. In *NIPS*.

R. Girshick, J. Donahue, T. Darrell, and J. Malik. 2013. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint:1311.2524*, November.

Douwe Kiela and Stephen Clark. 2014. A Systematic Study of Semantic Vector Space Model Parameters. In *Proceedings of EACL 2014, Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*.

Douwe Kiela, Felix Hill, Anna Korhonen, and Stephen Clark. 2014. Improving Multi-Modal Representations Using Image Dispersion: Why Less is Sometimes More. In *Proceedings of ACL 2014*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114.

Angeliki Lazaridou, Elia Bruni, and Marco Baroni. 2014. Is this a wampimuk? cross-modal mapping between distributional semantics and the visual world. In *Proceedings of ACL 2014*.

Geoffrey Leech, Roger Garside, and Michael Bryant. 1994. Claws4: the tagging of the British National Corpus. In *Proceedings of the 15th conference on Computational linguistics-Volume 1*, pages 622–628. Association for Computational Linguistics.

Ben Leong and Rada Mihalcea. 2011. Going Beyond Text: A Hybrid Image-Text Approach for Measuring Word Relatedness. In *Proceedings of Joint International Conference on Natural Language Processing (IJCNLP)*, Chiang Mai, Thailand.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL 2014*.

M. M. Louwerse. 2011. Symbol interdependency in symbolic and embodied cognition. *TopiCS in Cognitive Science*, 3:273—302.

David G. Lowe. 1999. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of International Conference of Learning Representations*, Scottsdale, Arizona, USA.

M. Oquab, L. Bottou, I. Laptev, and J. Sivic. 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

A.S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. 2014. CNN features off-the-shelf: an astounding baseline for recognition. *arXiv preprint:1403.6382*.

Stephen Roller and Sabine Schulte im Walde. 2013. A multimodal LDA model integrating textual, cognitive and visual modalities. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1146–1157, Seattle, Washington, USA, October. Association for Computational Linguistics.

D Sculley. 2010. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM.

Carina Silberer and Mirella Lapata. 2012. Grounded models of semantic representation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1423–1433. Association for Computational Linguistics.

Carina Silberer and Mirella Lapata. 2014. Learning Grounded Meaning Representations with Autoencoders. In *Proceedings of ACL 2014*, Baltimore, MD.

J. Sivic and A. Zisserman. 2003. Video Google: a text retrieval approach to object matching in videos. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, volume 2, pages 1470–1477, Oct.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded Compositional Semantics for Finding and Describing Images with Sentences. *Transactions of the Association for Computational Linguistics (TACL 2014)*.

Nitish Srivastava and Ruslan Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2222–2230.

A. Vedaldi and B. Fulkerson. 2008. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`.

Luis Von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM.

Pengcheng Wu, Steven C.H. Hoi, Hao Xia, Peilin Zhao, Dayong Wang, and Chunyan Miao. 2013. Online multimodal deep similarity learning with application to image retrieval. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 153–162.

Matthew D. Zeiler and Rob Fergus. 2013. Visualizing and understanding convolutional networks. *CoRR*.

Identifying Argumentative Discourse Structures in Persuasive Essays

Christian Stab[†] **and Iryna Gurevych**[†‡]

[†]Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
[‡]Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research
www.ukp.tu-darmstadt.de

Abstract

In this paper, we present a novel approach for identifying argumentative discourse structures in persuasive essays. The structure of argumentation consists of several components (i.e. claims and premises) that are connected with argumentative relations. We consider this task in two consecutive steps. First, we identify the components of arguments using multiclass classification. Second, we classify a pair of argument components as either support or non-support for identifying the structure of argumentative discourse. For both tasks, we evaluate several classifiers and propose novel feature sets including structural, lexical, syntactic and contextual features. In our experiments, we obtain a macro F1-score of 0.726 for identifying argument components and 0.722 for argumentative relations.

1 Introduction

Argumentation is a crucial aspect of writing skills acquisition. The ability of formulating persuasive arguments is not only the foundation for convincing an audience of novel ideas but also plays a major role in general decision making and analyzing different stances. However, current writing support is limited to feedback about spelling, grammar, or stylistic properties and there is currently no system that provides feedback about written argumentation. By integrating argumentation mining in writing environments, students will be able to inspect their texts for plausibility and to improve the quality of their argumentation.

An *argument* consists of several components. It includes a claim that is supported or attacked by at least one premise. The *claim* is the central component of an argument. It is a controversial statement that should not be accepted by the reader without additional support.[1] The *premise* underpins the validity of the claim. It is a reason given by an author for persuading readers of the claim. *Argumentative relations* model the discourse structure of arguments. They indicate which argument components are related and constitute the structure of argumentative discourse. For example, the argument in the following paragraph contains four argument components: one claim (in bold face) and three premises (underlined).

> "(1) ***Museums and art galleries provide a better understanding about arts than Internet***. (2) *In most museums and art galleries, detailed descriptions in terms of the background, history and author are provided.* (3) *Seeing an artwork online is not the same as watching it with our own eyes, as* (4) *the picture online does not show the texture or three-dimensional structure of the art, which is important to study*."

In this example, the premises (2) and (3) support the claim (1) whereas premise (4) is a support for premise (3). Thus, this example includes three argumentative support relations holding between the components (2,1), (3,1) and (4,3) signaling that the source component is a justification of the target component. This illustrates two important properties of argumentative discourse structures. First, argumentative relations are often implicit (not indicated by discourse markers; e.g. the relation holding between (2) and (1)). Indeed, Marcu and Echihabi (2002) found that only 26% of the evidence relations in the RST Discourse Treebank (Carlson et al., 2001) include discourse markers.

[1]We use the term claim synonymously to *conclusion*. In our definition the differentiation between claims and premises does not indicate the validity of the statements but signals which components include the gist of an argument and which are given by the author as justification.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 46–56,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Second, in contrast to Rhetorical Structure Theory (RST) (Mann and Thompson, 1987), argumentative relations also hold between non-adjacent sentences/clauses. For instance, in the corpus compiled by Stab and Gurevych (2014) only 37% of the premises appear adjacent to a claim. Therefore, existing approaches of discourse analysis, e.g. based on RST, do not meet the requirements of argumentative discourse structure identification, since they only consider discourse relations between adjacent sentences/clauses (Peldszus and Stede, 2013). In addition, there are no distinct argumentative relations included in common approaches like RST or the Penn Discourse Treebank (PDTB) (Prasad et al., 2008), since they are focused on identifying general discourse structures (cp. section 2.2).

Most of the existing argumentation mining methods focus solely on the identification of argument components. However, identifying argumentative discourse structures is an important task (Sergeant, 2013) in particular for providing feedback about argumentation. First, argumentative discourse structures are essential for evaluating the quality of an argument, since it is not possible to examine how well a claim is justified without knowing which premises belong to it. Second, methods that recognize if a statement supports a given claim enable the collection of additional evidence from other sources. Third, the structure of argumentation is needed for recommending better arrangements of argument components and meaningful usage of discourse markers. Both foster argument comprehension and recall (Britt and Larson, 2003) and thus increase the argumentation quality. To the best of our knowledge, there is currently only one approach that aims at identifying argumentative discourse structures proposed by Mochales-Palau and Moens (2009). However, it relies on a manually created context-free grammar (CFG) and is tailored to the legal domain, which follows a standardized argumentation style. Therefore, it is likely that it will not achieve acceptable accuracy when applied to more general texts in which discourse markers are missing or even misleadingly used (e.g. student texts).

In this work, we present a novel approach for identifying argumentative discourse structures which includes two consecutive steps. In the first step, we focus on the identification of argument components using a multiclass classification approach. In the second step, we identify argumentative relations by classifying a pair of argument components as either support or non-support. In particular, the contributions of this work are the following: First, we introduce a novel approach for identifying argumentative discourse structures. Contrary to previous approaches, our approach is capable of identifying argumentative discourse structures even if discourse markers are missing or misleadingly used. Second, we present two novel feature sets for identifying argument components as well as argumentative relations. Third, we evaluate several classifiers and feature groups for identifying the best system for both tasks.

2 Related Work

2.1 Argumentation Mining

Previous research on argumentation mining spans several subtasks, including (1) the separation of argumentative from non-argumentative text units (Moens et al., 2007; Florou et al., 2013), (2) the classification of argument components or argumentation schemes (Rooney et al., 2012; Mochales-Palau and Moens, 2009; Teufel, 1999; Feng and Hirst, 2011), and (3) the identification of argumentation structures (Mochales-Palau and Moens, 2009; Wyner et al., 2010).

The separation of argumentative from non-argumentative text units is usually considered as a binary classification task and constitutes one of the first steps in an argumentation mining pipeline. Moens et al. (2007) propose an approach for identifying argumentative sentences in the Araucaria corpus (Reed et al., 2008). The argument annotations in Araucaria are based on a domain-independent argumentation theory proposed by Walton (1996). In their experiments, they obtain the best accuracy (73.75%) using a combination of word pairs, text statistics, verbs, and a list of keywords indicative for argumentative discourse. Florou et al. (2013) report a similar approach. They classify text segments crawled with a focused crawler as either containing an argument or not. Their approach is based on several discourse markers and features extracted from the tense and mood of verbs. They report an F1-score of 0.764 for their best performing system.

One of the first approaches focusing on the identification of argument components is *Argumentative Zoning* proposed by Teufel (1999). The underlying assumption of this work is that argu-

ment components extracted from a scientific article provide a good summary of its content. Each sentence is classified as one of seven rhetorical roles including claim, result or purpose. The approach obtained an F1-score of 0.46[2] using structural, lexical and syntactic features. Rooney et al. (2013) also focus on the identification of argument components but in contrast to the work of Teufel (1999) their scheme is not tailored to a particular genre. In their experiments, they identify claims, premises and non-argumentative text units in the Araucaria corpus and report an overall accuracy of 65%. Feng and Hirst (2011) also use the Araucaria corpus for their experiments but focus on the identification of *argumentation schemes* (Walton, 1996), which are templates for forms of arguments (e.g. argument from example or argument from consequence). Since their approach is based on features extracted from mutual information of claims and premises, it requires that the argument components are reliably identified in advance. In their experiments, they achieve an accuracy between 62.9% and 97.9% depending on the particular scheme and the classification setup.

In contrast to all approaches mentioned above, the work presented in this paper focuses besides the separation of argumentative from non-argumentative text units and the classification of argument components on the extraction of the argumentative discourse structure to identify which components of the argument belong together for achieving a more fine-grained and detailed analysis of argumentation. We are only aware of one approach (Mochales-Palau and Moens, 2009; Wyner et al., 2010) that also focuses on the identification of argumentative discourse structures. However, this approach is based on a manually created CFG that is tailored to documents from the legal domain, which follow a standardized argumentation style. Therefore, it does not accommodate ill-formatted arguments (Wyner et al., 2010), which are likely in argumentative writing support. In addition, the approach relies on discourse markers and is therefore not applicable for identifying implicit argumentative discourse structures.

2.2 Discourse Relations

Identifying argumentative discourse structures is closely related to discourse analysis. As illustrated in the initial example, the identification of argumentative relations postulates the identification of implicit as well as non-adjacent discourse relations. Marcu and Echihabi (2002) present the first approach focused on identifying implicit discourse relations. They exploit several discourse markers (e.g. *'because'* or *'but'*) for collecting large amounts of training data. For their experiments they remove the discourse markers and discover that word pair features are indicative for implicit discourse relations. Depending on the utilized corpus, they obtain accuracies between 64% and 75% for identifying a cause-explanation-evidence relation (the most similar relation of their work compared to argumentative relations).

With the release of the PDTB, the identification of discourse relations gained a lot of interest in the research community. The PDTB includes implicit as well as explicit discourse relations of different types, and there are multiple approaches aiming at automatically identifying implicit relations. Pitler et al. (2009) experiment with polarity tags, verb classes, length of verb phrases, modality, context and lexical features and found that word pairs with non-zero Information Gain yield best results. Lin et al. (2009) show that beside lexical features, production rules collected from parse trees yield good results, whereas Louis et al. (2010) found that features based on named-entities do not perform as well as lexical features. However, current approaches to discourse analysis like the RST or the PDTB are designed to analyze general discourse structures, and thus include a large set of generic discourse relations, whereas only a subset of those relations is relevant for argumentative discourse analysis. For instance, the argumentation scheme proposed by Peldszus and Stede (2013) includes three argumentative relations (support, attack and counter-attack), whereas Stab and Gurevych (2014) propose a scheme including only two relations (support and attack). The difference between argumentative relations and those included in general tagsets like RST and PDTB is best illustrated by the work of Biran and Rambow (2011), which is to the best of our knowledge the only work that focuses on the identification of argumentative relations. They argue that existing definitions of discourse relations are only relevant as a building block for identifying argumentative discourse and that existing approaches do not contain a single relation that corresponds to

[2]Calculated from the precision and recall scores provided for individual rhetorical roles in (Teufel, 1999, p. 225).

a distinct argumentative relation. Therefore, they consider a set of 12 discourse relations from the RST Discourse Treebank (Carlson et al., 2001) as a single argumentative relation in order to identify justifications for a given claim. They first extract a set of lexical indicators for each relation from the RST Discourse Treebank and create a word pair resource using the English Wikipedia. In their experiments, they use the extracted word pairs as features and obtain an F1-score of up to 0.51 using two different corpora. Although the approach considers non-adjacent relations, it is limited to the identification of relations between premises and claims and requires that claims are known in advance. In addition, the combination of several general relations to a single argumentative relation might lead to consistency problems and to noisy corpora (e.g. not each instance of a contrast relation is relevant for argumentative discourse).

3 Data

For our experiments, we use a corpus of persuasive essays compiled by Stab and Gurevych (2014). This corpus contains annotations of argument components at the clause-level as well as argumentative relations. In particular, it includes annotations of *major claims*, *claims* and *premises*, which are connected with argumentative *support* and *attack* relations. Argumentative relations are directed (there is a specified source and target component of each relation) and can hold between a premise and another premise, a premise and a (major-) claim, or a claim and a major claim. Except for the last one, an argumentative relation does not cross paragraph boundaries.

Three raters annotated the corpus with an inter-annotator agreement of $\alpha_U = 0.72$ (Krippendorff, 2004) for argument components and $\alpha = 0.81$ for argumentative relations. In total, the corpus comprises 90 essays including 1,673 sentences. Since it only contains a low number of attack relations, we focus in this work solely on the identification of argument components and argumentative support relations. However, the proposed approach can also be applied to identify attack relations in future work.

4 Identifying Argument Components

We consider the identification of argument components as a multiclass classification task. Each clause in the corpus is either classified as major claim, claim, premise or non-argumentative. So this task includes besides the classification of argument components also the separation of argumentative and non-argumentative text units. We label each sentence that does not contain an argument component as class *'none'*. Since many argument components cover an entire sentence (30%), this is not an exclusive feature of this class. In total, the corpus contains 1,879 instances.

Table 1 shows the class distribution among the instances. The corpus includes 90 major claims (each essay contains exactly one), 429 claims and 1,033 premises. This proportion between claims and premises is common in argumentation since claims are usually supported by several premises for establishing a stable standpoint.

MajorClaim	Claim	Premise	None
90 (4.8%)	429 (22.8%)	1,033 (55%)	327 (17.4%)

Table 1: Class distribution among the instances. The corpus contains 1552 argument components and 327 non-argumentative instances.

For our experiments, we randomly split the data into a 80% training set and a 20% test set with the same class distribution and determine the best performing system using 10-fold cross-validation on the training set only. In our experiments, we use several classifiers (see section 4.2) from the Weka data mining software (Hall et al., 2009). For preprocessing the corpus, we use the Stanford POS-Tagger (Toutanova et al., 2003) and Parser (Klein and Manning, 2003) included in the DKPro Framework (Gurevych et al., 2007). After these steps, we use the DKPro-TC text classification framework (Daxenberger et al., 2014) for extracting the features described in the following section.

4.1 Features

Structural features: We define structural features based on token statistics, the location and punctuations of the argument component and its covering sentence. Since Biran and Rambow (2011) found that premises are longer on the average than other sentences, we add the number of tokens of the argument component and its covering sentence to our feature set. In addition, we define the number of tokens preceding and following an argument component in the covering sentence, the token ratio between covering sentence and argument component, and a Boolean feature that indicates if the

argument component covers all tokens of its covering sentence as token statistics features.

For exploiting the structural properties of persuasive essays, we define a set of location-based features. First, we define four Boolean features that indicate if the argument component is present in the introduction or conclusion of an essay and if it is present in the first or the last sentence of a paragraph. Second, we add the position of the covering sentence in the essay as a numeric feature. Since major claims are always present in the introduction or conclusion of an essay and paragraphs frequently begin or conclude with a claim, we expect that these features are good indicators for classifying (major-) claims.

Further, we define structural features based on the punctuation: the number of punctuation marks of the covering sentence and the argument component, the punctuation marks preceding and following an argument component in its covering sentence and a Boolean feature that indicates if the sentence closes with a question mark.

Lexical features: We define n-grams, verbs, adverbs and modals as lexical features. We consider all n-grams of length 1-3 as a Boolean feature and extract them from the argument component including preceding tokens in the sentence that are not covered by another argument component. So, the n-gram features include discourse markers that indicate certain argument components but which are not included in the actual annotation of argument components.

Verbs and adverbs play an important role for identifying argument components. For instance, certain verbs like *'believe'*, *'think'* or *'agree'* often signal stance expressions which indicate the presence of a major claim and adverbs like *'also'*, *'often'* or *'really'* emphasize the importance of a premise. We model both verbs and adverbs as Boolean features.

Modal verbs like *'should'* and *'could'* are frequently used in argumentative discourse to signal the degree of certainty when expressing a claim. We use the POS tags generated during preprocessing to identify modals and define a Boolean feature which indicates if an argument component contains a modal verb.

Syntactic features: To capture syntactic properties of argument components, we define features extracted from parse trees. We adopt two features proposed by (Mochales-Palau and Moens, 2009): the number of sub-clauses included in the covering sentence and the depth of the parse tree. In addition, we extract *production rules* from the parse tree as proposed by Lin et al. (2009) to capture syntactic characteristics of an argument component. The production rules are collected for each function tag (e.g. VP, NN, S, etc.) in the subtree of an argument component. The feature set includes e.g. rules like $VP \rightarrow VBG, NP$ or $PP \rightarrow IN, NP$. We model each production rule as a Boolean feature and set it to true if it appears in the subtree of an argument component.

Since premises often refer to previous events and claims are usually in present tense, we capture the tense of the main verb of an argument component as proposed by Mochales-Palau and Moens (2009) and define a feature that indicates if an argument component is in the past or present tense.

Indicators: Discourse markers often indicate the components of an argument. For example, claims are frequently introduced with *'therefore'*, *'thus'* or *'consequently'*, whereas premises contain markers like *'because'*, *'reason'* or *'furthermore'*. We collected a list of discourse markers from the Penn Discourse Treebank 2.0 Annotation Manual (Prasad et al., 2007) and removed markers that do not indicate argumentative discourse (e.g. markers which indicate temporal discourse). In total, we collected 55 discourse markers and model each as a Boolean feature set to true if the particular marker precedes the argumentative component.

In addition, we define five Boolean features which denote a reference to the first person in the covering sentence of an argument component: *'I'*, *'me'*, *'my'*, *'mine'*, and *'myself'*. An additional Boolean feature indicates if one of them is present in the covering sentence. We expect that those features are good indicators of the major claim, since it is often introduced with expressions referring to the personal stance of the author.

Contextual features: The context plays a major role for identifying argument components. For instance, a premise can only be classified as such, if there is a corresponding claim. Therefore, we define the following features each extracted from the sentence preceding and following the covering sentence of an argument component: the number of punctuations, the number of tokens, the number of sub-clauses and a Boolean feature indicating the presence of modal verbs.

4.2 Results and Analysis

For identifying the best performing system, we conducted several experiments on the training set using stratified 10-fold cross-validation. We determine the evaluation scores by accumulating the confusion matrices of each fold into one confusion matrix, since it is the less biased method for evaluating cross-validation studies (Forman and Scholz, 2010). In a comparison of several classifiers (Support Vector Machine, Naïve Bayes, C4.5 Decision Tree and Random Forest), we found that each of the classifiers significantly outperforms a majority baseline (McNemar Test (McNemar, 1947) with $p = 0.05$) and that a Support Vector Machine (SVM) achieves the best results using 100 top features ranked by Information Gain.[3] It achieves an accuracy of 77.3% on the test set and outperforms the majority baseline with respect to overall accuracy as well as F1-score (table 2).

	Baseline	**Human**	**SVM**
Accuracy	0.55	0.877	0.773
Macro F1	0.177	0.871	0.726
Macro Precision	0.137	0.864	0.773
Macro Recall	0.25	0.879	0.684
F1 MajorClaim	0	0.916	0.625
F1 Claim	0	0.841	0.538
F1 Premise	0.709	0.911	0.826
F1 None	0	0.812	0.884

Table 2: Results of an SVM for argument component classification on the test set compared to a majority baseline and human performance.

The upper bound for this task constitutes the human performance which we determine by comparing each annotator to the gold standard. Since the boundaries of an argument component in the gold standard can differ from the boundaries identified by a human annotator (the annotation task included the identification of argument component boundaries), we label each argument component of the gold standard with the class of the maximum overlapping annotation of a human annotator for determining the human performance. We obtain a challenging upper bound of 87.7% (accuracy) by averaging the scores of all three annotators on the test set (table 2). So, our system achieves 88.1% of human performance (accuracy).

Feature influence: In subsequent experiments, we evaluate each of the defined feature groups on the entire data set using 10-fold cross-validation to find out which features perform best for identifying argument components. As assumed, structural features perform well for distinguishing claims and premises in persuasive essays. They also yield high results for separating argumentative from non-argumentative text units (table 3).

Feature group	**MajorClaim**	**Claim**	**Premise**	**None**
Structural	0.477	0.419	0.781	0.897
Lexical	0.317	0.401	0.753	0.275
Syntactic	0.094	0.292	0.654	0.427
Indicators	0.286	0.265	0.730	0
Contextual	0	0	0.709	0

Table 3: F1-scores for individual feature groups and classes (SVM with 10-fold cross-validation on the entire data set)

Interestingly, the defined indicators are not useful for separating argumentative from non-argumentative text units though they are helpful for classifying argument components. A reason for this could be that not each occurrence of an indicator distinctly signals argument components, since their sense is often ambiguous (Prasad et al., 2008). For example *'since'* indicates temporal properties as well as justifications, whereas *'because'* also indicates causal links. Syntactic features also contribute to the identification of argument components. They achieve an F1-score of 0.292 for claims and 0.654 for premises and also contribute to the separation of argumentative from non-argumentative text units. Contextual features do not perform well. However, they increase the accuracy by 0.7% in combination with other features. Nevertheless, this difference is not significant ($p = 0.05$).

Error analysis: The system performs well for separating argumentative and non-argumentative text units as well as for identifying premises. However, the identification of claims and major claims yields lower performance. The confusion matrix (table 4) reveals that the most common error is between claims and premises. In total, 193 claims are incorrectly classified as premise. In a manual assessment, we observed that many of these errors occur if the claim is present in the first paragraph sentence and exhibits preceding indicators like *'first(ly)'* or *'second(ly)'* which are also frequently used to enumerate premises. In these cases, the author introduces the claim of the argument as support for the major claim and thus its characteristic is similar to a premise. To prevent

[3] Although the Naïve Bayes classifier achieves lowest accuracy, it exhibits a slightly higher recall compared to SVM.

this type of error, it might help to define features representing the location of indicators or to disambiguate the function of indicators.

Actual \ *Predicted*	MC	Cl	Pr	No
MC	**38**	34	18	0
Cl	19	**210**	193	7
Pr	6	104	**904**	19
No	0	12	23	**292**

Table 4: Confusion matrix (SVM) for argument component classification (MC = Major Claim; Cl = Claim; Pr = Premise; No = None)

We also observed, that some of the misclassified claims cover an entire sentence and don't include indicators. For example, it is even difficult for humans to classify the sentences *'Competition helps in improvement and evolution'* as a claim without knowing the intention of the author. For preventing these errors, it might help to include more sophisticated contextual features.

5 Identifying Argumentative Relations

We consider the identification of argumentative relations as a binary classification task of argument component pairs and classify each pair as either support or non-support. For identifying argumentative relations, all possible combinations of argument components have to be tested. Since this results in a heavily skewed class distribution, we extract all possible combinations of argument components from each paragraph of an essay.[4] So, we omit argumentative relations between claims and major claims which are the only relations in the corpus that cross paragraph boundaries, but obtain a better distribution between true (support) and false (non-support) instances. In total, we obtain 6,330 pairs, of which 15.6% are support and 84.4% are non-support relations (table 5).

Support	Non-support
989 (15.6%)	5341 (84.4%)

Table 5: Class distribution of argument component pairs

Equivalent to the identification of argument components, we randomly split the data in a 80% training and a 20% test set and determine the best performing system using 10-fold cross-validation on the training set. We use the same preprocessing pipeline as described in section 4 and DKPro-TC for extracting the features described below.

5.1 Features

Structural features: We define structural features for each pair based on the source and target components, and on the mutual information of both. Three numeric features are based on token statistics. Two features represent the number of tokens of the source and target components and the third one represents the absolute difference in the number of tokens. Three additional numeric features count the number of punctuation marks of the source and target components as well as the absolute difference between both. We extract both types of features solely from the clause annotated as argument component and do not consider the covering sentence. In addition, we define nine structural features based on the position of both argument components: two of them represent the position of the covering sentences in the essay, four Boolean features indicate if the argument components are present in the first or last sentence of a paragraph, one Boolean feature for representing if the target component occurs before the source component, the sentence distance between the covering sentences, and a Boolean feature which indicates if both argument components are in the same sentence.

Lexical features: We define lexical features based on word pairs, first words and modals. It has been shown in previous work that word pairs are effective for identifying implicit discourse relations (Marcu and Echihabi, 2002). We define each pair of words between the source and target components as a Boolean feature and investigate word pairs containing stop words as well as stop word filtered word pairs.

In addition, we adopt the first word features proposed by Pitler et al. (2009). We extract the first word either from the argument component or from non-annotated tokens preceding the argument component in the covering sentence if present. So, the first word of an argument component is either the first word of the sentence containing the argument component, the first word following a preceding argument component in the same sentence or the first word of the actual argument component if it commences the sentence or directly follows another argument component.

[4] Only 4.6% of 28,434 possible pairs are true instances (support), if all combinations are considered.

So, we ensure that the first word of an argument component includes important discourse markers which are not included in the annotation. We define each first word of the source and target components as a Boolean feature and also add the pairs of first words to our feature set.

Further, we define a Boolean feature for the source as well as for the target component that indicates if they contain a modal verb and a numerical feature that counts the number of common terms of the two argument components.

Syntactic features: For capturing syntactic properties, we extract production rules from the source and target components. Equivalent to the features extracted for the argument component classification (section 4.1), we model each rule as a Boolean feature which is true if the corresponding argument component includes the rule.

Indicators: We use the same list of discourse markers introduced above (section 4.1) as indicator features. For each indicator we define a Boolean feature for the source as well as for the target component of the pair and set it to true if it is present in the argument component or in its preceding tokens.

Predicted type: The *argumentative type* (major claim, claim or premise) of the source and target components is a strong indicator for identifying argumentative relations. For example, there are no argumentative relations from claims to premises. Thus, if the type of the argument component is reliably identified many potential pairs can be excluded. Therefore, we define two features that represent the argumentative type of the source and target components identified in the first experiment.

5.2 Results and Analysis

The comparison of several classifiers reveals that an SVM achieves the best results. In our experiments, all classifiers except the C4.5 Decision Tree significantly outperform a majority baseline which classifies all pairs as non-support ($p = 0.05$). We also conducted several experiments using word pair features only and found in contrast to Pitler et al. (2009) that limiting the number of word pairs decreases the performance. In particular, we compared the top 100, 250, 500, 1000, 2500, 5000 word pairs ranked by Information Gain, non-zero Information Gain word pairs and non-filtered word pairs. The results show that non-filtered word pairs perform best (macro F1-score of 0.68). Our experiments also reveal that filtering stop words containing word pairs decreases the macro F1-score to 0.60. We obtain the best results using an SVM without any feature selection method. Due to the class imbalance, the SVM only slightly outperforms the accuracy of a majority baseline on the test set (table 6). However, the macro F1-score is more appropriate for evaluating the performance if the data is imbalanced since it assigns equal weight to the classes and not to the instances. The SVM achieves a macro F1-score of 0.722 and also outperforms the baseline with respect to the majority class.

	Baseline	**Human**	**SVM**
Accuracy	0.843	0.954	0.863
Macro F1	0.458	0.908	0.722
Macro Precision	0.422	0.937	0.739
Macro Recall	0.5	0.881	0.705
F1 Support	0	0.838	0.519
F1 Non-Support	0.915	0.973	0.92

Table 6: Results of an SVM for classifying argumentative relations on the test set compared to a majority baseline and human performance.

We determined the upper bound constituted by the human performance by comparing the annotations of all three annotators to the gold standard. The scores in table 6 are the average scores of all three annotators. Our system achieves 90.5% of human performance (accuracy).

Feature influence: A comparison of the defined feature groups using 10-fold cross-validation on the entire data set shows that lexical features perform best. They achieve an F1-score of 0.427 for support and 0.911 for non-support pairs (table 7). The syntactic features also perform well followed by the indicators. It turned out that structural features are not effective for identifying argumentative relations though they are the most effective features for identifying argument components (cp. section 4.2). However, when omitted from the entire feature set the performance significantly decreases by 0.018 macro F1-score ($p = 0.05$).

Interestingly, the predicted types from our first experiment are not effective at all. Although the argumentative type of the target component exhibits the highest Information Gain in each fold compared to all other features, the predicted type does not yield a significant difference when combined with all other features ($p = 0.05$). It only improves the macro F1-score by 0.001 when in-

cluded in the entire feature set.

Feature group	Support	Non-Support
Structural	0	0.915
Lexical	0.427	0.911
Syntactic	0.305	0.911
Indicators	0.159	0.916
Predicted types	0	0.915

Table 7: F1-scores for individual feature groups using an SVM and the entire data set

Error analysis: For identifying frequent error patterns, we manually investigated the mistakes of the classifier. Although our system identifies 97.5% of the non-support pairs from claim to premise correctly, there are still some false positives that could be prevented if the argument components had been classified more accurately. For instance, there are 18 non-support relations from claim to another claim, 32 from claim to premise, 5 from major claim to premise and 4 from major claim to claim among the false positives. However, the larger amount of errors is due to not identified support relations (false negatives). We found that some errors might be related to missing contextual information and unresolved coreferences. For instance, it might help to replace *'It'* with *'Exercising'* for classifying the pair *'It helps relieve tension and stress'* → *'Exercising improves self-esteem and confidence'* as support relation or to include contextual information for the premise *'This can have detrimental effects on health'* supporting the claim *'There are some serious problems springing from modern technology'*.

6 Discussion

In our experiments, we have investigated the classification of argument components as well as the identification of argumentative relations for recognizing argumentative discourse structures in persuasive essays. Both tasks are closely related and we assume that sharing mutual information between both tasks might be a promising direction for future research. On the one hand, knowing the type of argument components is a strong indicator for identifying argumentative relations and on the other hand, it is likely that information about the argumentative structure facilitates the identification of argument components. However, our experiments revealed that the current accuracy for identifying argument components is not sufficient for increasing the performance of argumentative relation identification. Nevertheless, we obtain almost human performance when including the types of argument components of the gold standard (macro F1-score >0.85) in our argument relation identification experiment and when including the number of incoming and outgoing support relations for each argument component in our first experiment (macro F1-score >0.9). Therefore, it can be assumed, that if the identification of argument components can be improved, the identification of argumentative relations will achieve better results and vice versa.

The results also show that the distinction between claims and premises is the major challenge for identifying argument components. It turned out that structural features are the most effective ones for this task. However, some of those features are unique to persuasive essays, and it is an open question if there are general structural properties of arguments which can be exploited for separating claims from premises.

Our experiments show that discourse markers yield only low accuracies. Using only our defined indicator features, we obtain an F1-score of 0.265 for identifying claims, whereas Mochales-Palau and Moens (2009) achieve 0.673 for the same task in legal documents using a CFG. This confirms our initial assumption that approaches relying on discourse markers are not applicable for identifying argumentative discourse structures in documents which do not follow a standardized form. In addition, it shows that discourse markers are either frequently missing or misleadingly used in student texts and that there is a need for argumentative writing support systems that assist students in employing discourse markers correctly.

7 Conclusion and Future Work

We presented a novel approach for identifying argumentative discourse structures in persuasive essays. Previous approaches on argument recognition suffer from several limitations: Existing approaches focus either solely on the identification of argument components or rely on manually created rules which are not able to identify implicit argumentative discourse structures. Our approach is the first step towards computational argument analysis in the educational domain and enables the identification of implicit argumentative discourse structures. The presented approach achieves 88.1% of human performance for identi-

fying argument components and 90.5% for identifying argumentative relations.

For future work, we plan to extend our studies to larger corpora, to integrate our classifiers in writing environments, and to investigate their effectiveness for supporting students.

Acknowledgements

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806. We thank Krish Perumal and Piyush Paliwal for their valuable contributions and we thank the anonymous reviewers for their helpful comments.

References

Or Biran and Owen Rambow. 2011. Identifying justifications in written dialogs by classifying text as argumentative. *International Journal of Semantic Computing*, 05(04):363–381.

M. Anne Britt and Aaron A. Larson. 2003. Constructing representations of arguments. *Journal of Memory and Language*, 48(4):794 – 810.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue - Volume 16*, SIGDIAL '01, pages 1–10, Aalborg, Denmark.

Johannes Daxenberger, Oliver Ferschke, Iryna Gurevych, and Torsten Zesch. 2014. DKPro TC: A Java-based framework for supervised learning experiments on textual data. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. System Demonstrations*, pages 61–66, Baltimore, MD, USA.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 987–996, Portland, OR, USA.

Eirini Florou, Stasinos Konstantopoulos, Antonis Koukourikos, and Pythagoras Karampiperis. 2013. Argument extraction for supporting public policy formulation. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 49–54, Sofia, Bulgaria.

George Forman and Martin Scholz. 2010. Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *SIGKDD Explor. Newsl.*, 12(1):49–57.

Iryna Gurevych, Max Mühlhäuser, Christof Mueller, Juergen Steimle, Markus Weimer, and Torsten Zesch. 2007. Darmstadt Knowledge Processing Repository based on UIMA. In *Proceedings of the First Workshop on Unstructured Information Management Architecture at Biannual Conference of the Society for Computational Linguistics and Language Technology, Tuebingen, Germany*.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Sapporo, Japan.

Klaus Krippendorff. 2004. Measuring the Reliability of Qualitative Text Analysis Data. *Quality & Quantity*, 38(6):787–800.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1*, EMNLP '09, pages 343–351, Stroudsburg, PA, USA.

Annie Louis, Aravind Joshi, Rashmi Prasad, and Ani Nenkova. 2010. Using entity features to classify implicit discourse relations. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '10, pages 59–62, Stroudsburg, PA, USA.

William C. Mann and Sandra A. Thompson. 1987. Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, Information Sciences Institute.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 368–375.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Raquel Mochales-Palau and Marie-Francine Moens. 2009. Argumentation mining: The detection, classification and structure of arguments in text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, ICAIL '09, pages 98–107, New York, NY, USA. ACM.

Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. 2007. Automatic detection of arguments in legal texts. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law*, ICAIL '07, pages 225–230, Stanford, California.

Andreas Peldszus and Manfred Stede. 2013. From Argument Diagrams to Argumentation Mining in Texts: A Survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, ACL '09, pages 683–691, Suntec, Singapore. Association for Computational Linguistics.

Rashmi Prasad, Eleni Miltsakaki, Nikhil Dinesh, Alan Lee, Aravind Joshi, Livio Robaldo, and Bonnie L. Webber. 2007. The Penn Discourse Treebank 2.0 annotation manual. Technical report, Institute for Research in Cognitive Science, University of Pennsylvania.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse Treebank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Chris Reed, Raquel Mochales-Palau, Glenn Rowe, and Marie-Francine Moens. 2008. Language resources for studying argument. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*, LREC '08, pages 2613–2618, Marrakech, Morocco.

Niall Rooney, Hui Wang, and Fiona Browne. 2012. Applying kernel methods to argumentation mining. In *Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference*, FLAIRS '12, pages 272–275, Marco Island, FL, USA.

Alan Sergeant. 2013. Automatic argumentation extraction. In *Proceedings of the 10th European Semantic Web Conference*, ESWC '13, pages 656–660, Montpellier, France.

Christian Stab and Iryna Gurevych. 2014. Annotating argument components and relations in persuasive essays. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 1501–1510, Dublin, Ireland, August.

Simone Teufel. 1999. *Argumentative Zoning: Information Extraction from Scientific Text*. Ph.D. thesis, University of Edinburgh.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, NAACL '03, pages 173–180, Edmonton, Canada.

Douglas N Walton. 1996. *Argumentation schemes for presumptive reasoning*. Routledge.

Adam Wyner, Raquel Mochales Palau, Marie-Francine Moens, and David Milward. 2010. Approaches to text mining arguments from legal cases. In *Semantic Processing of Legal Texts*, volume 6036 of *Lecture Notes in Computer Science*, pages 60–79.

Policy Learning for Domain Selection in an Extensible Multi-domain Spoken Dialogue System

Zhuoran Wang
Mathematical & Computer Sciences
Heriot-Watt University
Edinburgh, UK
zhuoran.wang@hw.ac.uk

Hongliang Chen, Guanchun Wang
Hao Tian, Hua Wu†, Haifeng Wang
Baidu Inc., Beijing, P. R. China
SurnameForename@baidu.com
†wu_hua@baidu.com

Abstract

This paper proposes a Markov Decision Process and reinforcement learning based approach for domain selection in a multi-domain Spoken Dialogue System built on a distributed architecture. In the proposed framework, the domain selection problem is treated as sequential planning instead of classification, such that confirmation and clarification interaction mechanisms are supported. In addition, it is shown that by using a model parameter tying trick, the extensibility of the system can be preserved, where dialogue components in new domains can be easily plugged in, without re-training the domain selection policy. The experimental results based on human subjects suggest that the proposed model marginally outperforms a non-trivial baseline.

1 Introduction

Due to growing demand for natural human-machine interaction, over the last decade Spoken Dialogue Systems (SDS) have been increasingly deployed in various commercial applications ranging from traditional call centre automation (e.g. AT&T "Lets Go!" bus information system (Williams et al., 2010)) to mobile personal assistants and knowledge navigators (e.g. Apple's Siri®, Google Now™, Microsoft Cortana, etc.) or voice interaction for smart household appliance control (e.g. Samsung Evolution Kit for Smart TVs). Furthermore, latest progress in open-vocabulary Automatic Speech Recognition (ASR) is pushing SDS from traditional single-domain information systems towards more complex multi-domain speech applications, of which typical examples are those voice assistant mobile applications.

Recent advances in SDS have shown that statistical approaches to dialogue management can result in marginal improvement in both the naturalness and the task success rate for domain-specific dialogues (Lemon and Pietquin, 2012; Young et al., 2013). State-of-the-art statistical SDS treat the dialogue problem as a sequential decision making process, and employ established planning models, such as Markov Decision Processes (MDPs) (Singh et al., 2002) or Partially Observable Markov Decision Processes (POMDPs) (Thomson and Young, 2010; Young et al., 2010; Williams and Young, 2007), in conjunction with reinforcement learning techniques (Jurčíček et al., 2011; Jurčíček et al., 2012; Gašić et al., 2013a) to seek optimal dialogue policies that maximise long-term expected (discounted) rewards and are robust to ASR errors.

However, to the best of our knowledge, most of the existing multi-domain SDS in public use are rule-based (e.g. (Gruber et al., 2012; Mirkovic and Cavedon, 2006)). The application of statistical models in multi-domain dialogue systems is still preliminary. Komatani et al. (2006) and Nakano et al. (2011) utilised a distributed architecture (Lin et al., 1999) to integrate expert dialogue systems in different domains into a unified framework, where a central controller trained as a data-driven classifier selects a domain expert at each turn to address user's query. Alternatively, Hakkani-Tür et al. (2012) adopted the well-known Information State mechanism (Traum and Larsson, 2003) to construct a multi-domain SDS and proposed a discriminative classification model for more accurate state updates. More recently, Gašić et al. (2013b) proposed that by a simple expansion of the kernel function in Gaussian Process (GP) reinforcement learning (Engel et al., 2005; Gašić et al., 2013a), one can adapt pre-trained dialogue policies to handle unseen slots for SDS in extended domains.

In this paper, we use a voice assistant applica-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 57–67,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

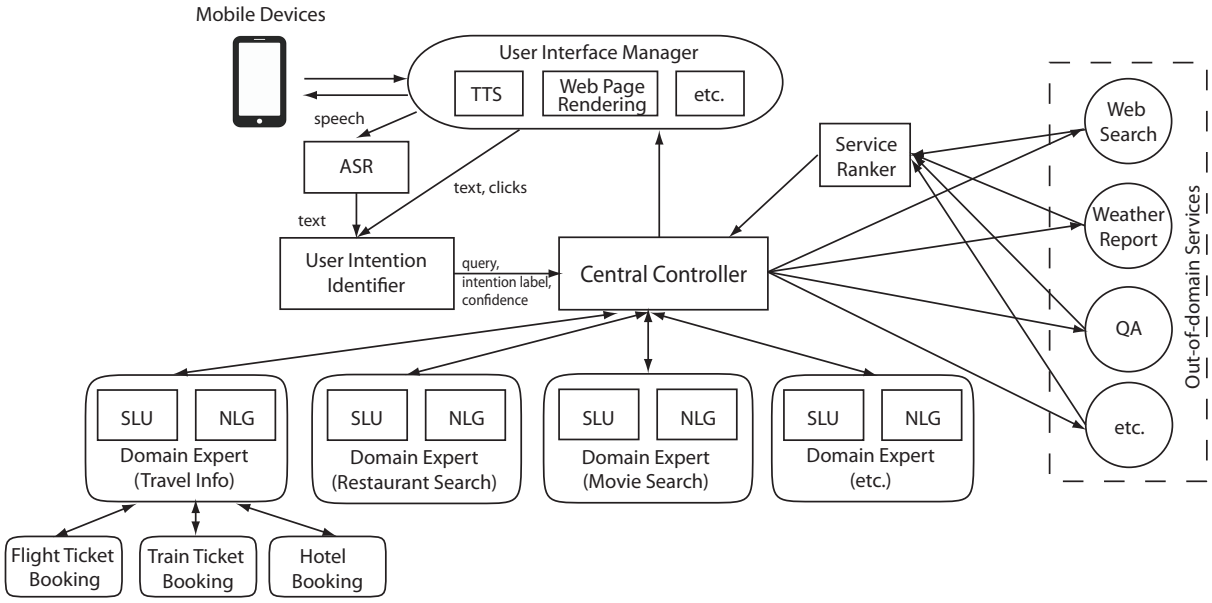


Figure 1: The distributed architecture of the voice assistant system (a simplified illustration).

tion (similar to Apple's Siri but in Chinese language) as an example to demonstrate a novel MDP-based approach for central interaction management in a complex multi-domain dialogue system. The voice assistant employs a distributed architecture similar to (Lin et al., 1999; Komatani et al., 2006; Nakano et al., 2011), and handles mixed interactions of multi-turn dialogues across different domains and single-turn queries powered by a collection of information access services (such as web search, Question Answering (QA), etc.). In our system, the dialogues in each domain are managed by an individual domain expert SDS, and the single-turn services are used to handle those so-called out-of-domain requests. We use featurised representations to summarise the current dialogue states in each domain (see Section 3 for more details), and let the central controller (the MDP model) choose one of the following system actions at each turn: (1) addressing user's query based on a domain expert, (2) treating it as an out-of-domain request, (3) asking user to confirm whether he/she wants to continue a domain expert's dialogue or to switch to out-of-domain services, and (4) clarifying user's intention between two domains. The Gaussian Process Temporal Difference (GPTD) algorithm (Engel et al., 2005; Gašić et al., 2013a) is adopted here for policy optimisation based on human subjects, where a parameter tying trick is applied to preserve the extensibility of the system, such that new domain experts (dialogue systems) can be flexibly plugged in without the need of re-training the central controller.

Comparing to the previous classification-based methods (Komatani et al., 2006; Nakano et al., 2011), the proposed approach not only has the advantage of action selection in consideration of long-term rewards, it can also yield more robust policies that allow clarifications and confirmations to mitigate ASR and Spoken Language Understanding (SLU) errors. Our human evaluation results show that the proposed system with a trained MDP policy achieves significantly better naturalness in domain switching tasks than a non-trivial baseline with a hand-crafted policy.

The remainder of this paper is organised as follows. Section 2 defines the terminology used throughout the paper. Section 3 briefly overviews the distributed architecture of our system. The MDP model and the policy optimisation algorithm are introduced in Section 4 and Section 5, respectively. After this, experimental settings and evaluation results are described in Section 6. Finally, we discuss some possible improvements in Section 7 and conclude ourselves in Section 8.

2 Terminology

A voice assistant application provides a unified speech interface to a collection of individual information access systems. It aims to collect and satisfy user requests in an interactive manner, where

different types of interactions can be involved. Here we focus ourselves on two interaction scenarios, i.e. task-oriented (multi-turn) dialogues and single-turn queries.

According to user intentions, the dialogue interactions in our voice assistant system can further be categorised into different *domains*, of which each is handled by a separate dialogue manager, namely a *domain expert*. Example domains include travel information, restaurant search, etc. In addition, some domains in our system can be further decomposed into sub-domains, e.g. the travel information domain consists of three sub-domains: flight ticket booking, train ticket booking and hotel reservation. We use an integrated domain expert to address queries in all its sub-domains, so that relevant information can be shared across those subdomains to allow intelligent induction in the dialogue flow.

For convenience of future reference, we call those single-turn information access systems *out-of-domain services* or simply *services* for short. The services integrated in our system include web search, semantic search, QA, system command execution, weather report, chat-bot, and many more.

3 System Architecture

The voice assistant system introduced in this paper is built on a distributed architecture (Lin et al., 1999), as shown in Figure 1, where the dialogue flow is processed as follows. Firstly, a user's query (either an ASR utterance or directly typed in text) is passed to a user intention identifier, which labels the raw query with a list of intention hypotheses with confidence scores. Here an intention label could be either a domain name or a service name. After this, the central controller distributes the raw query together with its intention labels and confidence scores to all the domain experts and the service modules, which will attempt to process the query and return their results to the central controller.

The domain experts in the current implementation of our system are all rule-based SDS following the RavenClaw framework proposed in (Bohus and Rudnicky, 2009). When receiving a query, a domain expert will use its own SLU module to parse the utterance or text input and try to update its dialogue state in consideration of both the SLU output and the intention labels. If the dialogue state in the domain expert can be updated given the query, it will return its output, internal session record and a confidence score to the central controller, where the output can be either a natural language utterance realised by its Natural Language Generation (NLG) module or a set of data records obtained from its database (if a database search operation is triggered), or both. If the domain expert cannot update its state using the current query, it will just return an empty result with a low confidence score. Similar procedures apply to those out-of-domain services as well, but there are no session records or confidence scores returned. Finally, given all the returned information, the central controller chooses, according to its policy, the module (either a domain expert or a service) whose results will be provided to the user.

When the central controller decides to pass a domain expert's output to the user, we regard the domain expert as being activated. Also note here, the updated state of a domain expert in a turn will not be physically stored, unless the domain expert is activated in that turn. This is a necessary mechanism to prevent an inactive domain expert being misled by ambiguous queries in other domains.

In addition, we use a well-engineered priority ranker to rank the services based on the numbers of results they returned as well as some prior knowledge about the quality of their data sources. When the central controller decides to show user the results from an out-of-domain service, it will choose the top one from the ranked list.

4 MDP Modelling of the Central Control Process

The main focus of this paper is to seek a policy for robustly switching the control flow among those domain experts and services (the service ranker in practice) during a dialogue, where the user may have multiple or compound goals (e.g. booking a flight ticket, booking a restaurant in the destination city and checking the weather report of the departure or destination city).

In order to make the system robust to ASR errors or ambiguous queries, the central controller should also have basic dialogue abilities for confirmation and clarification purposes. Here we define the confirmation as an action of asking whether a user wants to continue the dialogue in a certain domain. If the system receives a negative response at this point, it will switch to out-of-domain services. On the other hand, the clarification action is de-

fined between domains, in which case, the system will explicitly ask the user to choose between two domain candidates before continuing the dialogue.

Due to the confirmation and clarification mechanisms defined above, the central controller becomes a sequential decision maker that must take the overall smoothness of the dialogue into account. Therefore, we propose an MDP-based approach for learning an optimal central control policy in this section.

The potential state space of our MDP is huge, which in principle consists of the combinations of all possible situations of the domain experts and the out-of-domain services, therefore function approximation techniques must be employed to enable tractable computations. However, when developing such a complex application as the voice assistant here, one also needs to take the extensibility of the system into account, so that new domain experts can be easily integrated into the system without major re-training or re-engineering of the existing components. Essentially, it requires the state featurisation and the central control policy learnt here to be independent of the number of domain experts. In Section 4.3, we show that such a property can be achieved by a parameter tying trick in the definition of the MDP.

4.1 MDP Preliminaries

Let $\mathcal{P}_X$ denote the set of probability distributions over a set X. An MDP is defined as a five tuple $\langle S, A, T, R, \gamma \rangle$, where the components are defined as follows. S and A are the sets of system states and actions, respectively. $T : S \times A \rightarrow \mathcal{P}_S$ is the transition function, and $T(s'|s, a)$ defines the conditional probability of the system transiting from state $s \in S$ to state $s' \in S$ after taking action $a \in A$. $R : S \times A \rightarrow \mathcal{P}_{\mathbb{R}}$ is the reward function with $R(s, a)$ specifying the distribution of the immediate rewards for the system taking action a at state s. In addition, $0 \leq \gamma \leq 1$ is the discount factor on the summed sequence of rewards.

A finite-horizon MDP operates as follows. The system occupies a state s and takes an action a, which then will make it transit to a next state $s' \sim T(\cdot|s, a)$ and receive a reward $r \sim R(s, a)$. This process repeats until a terminal state is reached.

For a given policy $\pi : S \rightarrow A$, the value function V^π is defined to be the expected cumulative reward, as $V^\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{n} \gamma^t r_t|_{s_t, \pi(s_t)}\right]$, where s_0 is the starting state and n is the planning horizon. The aim of policy optimisation is to seek an optimal policy π^* that maximises the value function. If T and R are given, in conjunction with a Q-function, the optimal value V^* can be expressed by recursive equations as $Q(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s'|s, a)V^*(s')$ and $V^*(s) = \max_{a \in A} Q(s, a)$ (here we assume $R(s, a)$ is deterministic), which can be solved by dynamic programming (Bellman, 1957). For problems with unknown T or R, such as dialogue systems, the Q-values are usually estimated via reinforcement learning (Sutton and Barto, 1998).

4.2 Problem Definition

Let D denote the set of the domain experts in our voice assistant system, and s_d be the current dialogue state of domain expert $d \in D$ at a certain timestamp. We also define s_o as an abstract state to describe the current status of those out-of-domain services. Then mathematically we can represent the central control process as an MDP, where its state $\mathbf{s}$ is a joint set of the states of all the domain experts and the services, as $\mathbf{s} = \{s_d\}_{d \in D} \cup \{s_o\}$. Four types of system actions are defined as follows.

- `present(`d`)`: presenting the output of domain expert d to user;
- `present_ood(null)`: presenting the results of the top-ranked out-of-domain service given by the service ranker;
- `confirm(`d`)`: confirming whether user wants to continue with domain expert d (or to switch to out-of-domain services);
- `clarify(`d, d'`)`: asking user to clarify his/her intention between domains d and d'.

For convenience of notations, we use $a(x)$ to denote a system action of our MDP, where $a \in \{\texttt{present}, \texttt{present\_ood}, \texttt{confirm}, \texttt{clarify}\}$, $x \in \{d, \texttt{null}, (d, d')\}_{d,d' \in D, d \neq d'}$, $x = \texttt{null}$ only applies to `present_ood`, and $x = (d, d')$ only applies to `clarify` actions.

4.3 Function Approximation

Function approximation is a commonly used technique to estimate the Q-values when the state space of the MDP is huge. Concretely, in our case, we assume that:

$$Q(\mathbf{s}, a(x)) = f(\phi(\mathbf{s}, a(x)); \theta) \qquad (1)$$

where $\phi : S \times A \rightarrow \mathbb{R}^K$ is a feature function that maps a state-action pair to an K-dimensional feature vector, and $f : \mathbb{R}^K \rightarrow \mathbb{R}$ is a function of $\phi(\mathbf{s}, a(x))$ parameterised by θ. A frequent choice of f is the linear function, as:

$$Q(\mathbf{s}, a(x)) = \theta^\top \phi(\mathbf{s}, a(x)) \tag{2}$$

After this, the policy optimisation problem becomes learning the parameter θ to approximate the Q-values based on example dialogue trajectories.

However, a crucial problem with the standard formulation in Eq. (2) is that the feature function ϕ is defined over the entire state and action spaces. In this case, when a new domain expert is integrated into the system, both the state space and the action space will be changed, therefore one will have to re-define the feature function and consequentially re-train the model. In order to achieve an extensible system, we make some simplification assumptions and decompose the feature function as follows. Firstly, we let:

$$\begin{aligned} \phi(\mathbf{s}, a(x)) &= \phi_a(\mathbf{s}_x) \qquad (3) \\ &= \begin{cases} \phi_{\texttt{pr}}(s_d) & \text{if } a(x) = \texttt{present(}d\texttt{)} \\ \phi_{\texttt{ood}}(s_o) & \text{if } a(x) = \texttt{present\_ood()} \\ \phi_{\texttt{cf}}(s_d) & \text{if } a(x) = \texttt{confirm(}d\texttt{)} \\ \phi_{\texttt{cl}}(s_d, s_{d'}) & \text{if } a(x) = \texttt{clarify(}d\texttt{,}d'\texttt{)} \end{cases} \end{aligned}$$

where the feature function is reduced to only depend on the state of the action's operand, instead of the entire system state. Then, we make those actions $a(x)$ that have a same action type (a) but operate different domain experts (x) share the same parameter, i.e.:

$$Q(\mathbf{s}, a(x)) = \theta_a^\top \phi_a(\mathbf{s}_x) \tag{4}$$

This decomposition and parameter tying trick preserves the extensibility of the system, because both $\theta_a^\top$ and ϕ_a are independent of x, when there is a new domain expert $\tilde{d}$, we can directly substitute its state $s_{\tilde{d}}$ into Eq. (3) and (4) to compute its corresponding Q-values.

4.4 Features

Based on the problem formulation in Eq. (3) and (4), we shall only select high-level summary features to sketch the dialogue state and dialogue history of each domain expert, which must be applicable to all domain experts, regardless of their domain-specific characteristics or implementation differences. Suppose that the dialogue states of the domain experts can be represented as slot-value pairs[1], and for each domain there are required slots and optional slots, where all required slots must be filled before the domain expert can execute a database search operation. The features investigated in the proposed framework are listed in Table 1.

#	Feature	Range
1	the number of unfilled required slots of a domain expert	$\{0, \ldots, M\}$
2	the number of filled required slots of a domain expert	$\{0, \ldots, M\}$
3	the number of filled optional slots of a domain expert	$\{0, \ldots, L\}$
4	whether a domain expert has executed a database search	$\{0, 1\}$
5	the confidence score returned by a domain expert	$[0, 1.2]$
6	the total number of turns that a domain expert has been activated during a dialogue	$\mathbb{Z}^+$
7	e^{-t_a} where t_a denotes the relative turn of a domain expert being last activated, or 0 if not applicable	$[0, 1]$
8	e^{-t_c} where t_c denotes the relative turn of a domain expert being last confirmed, or 0 if not applicable	$[0, 1]$
9	the summed confidence score from the user intention identifier of a query being for out-of-domain services	$[0, 1.2N]$

Table 1: A list of all features used in our model. M and L respectively denote the maximum numbers of required and optional slots for the domain experts. N is the maximum number of hypotheses that the intention identifier can return. $\mathbb{Z}^+$ stands for the non-negative integer set.

Detailed featurisation in Eq. (3) is explained as follows. For $\phi_{\texttt{pr}}$, we choose the first 8 features plus a bias dimension that is always set to

[1]This is a rather general assumption. Informally speaking, for most task-oriented SDS, one can extract a slot-value representation from their dialogue models, of which examples include the RavenClaw architecture (Bohus and Rudnicky, 2009), the Information State dialogue engine (Traum and Larsson, 2003), MDP-SDS (Singh et al., 2002) or POMDP-SDS (Thomson and Young, 2010; Young et al., 2010; Williams and Young, 2007).

-1. Whilst, feature #9 plus a bias is used to define $\phi_{\texttt{ood}}$. All the features are used in $\phi_{\texttt{cf}}$, as to do a confirmation, one needs to consider the joint situation in and out of the domain. Finally, the feature function for a clarification action between two domains d and d' is defined as $\phi_{\texttt{cl}}(s_d, s_{d'}) = \exp\{-|\phi_{\texttt{pr}}(s_d) - \phi_{\texttt{pr}}(s_{d'})|\}$, where we use $|\cdot|$ to denote the element-wise absolute of a vector operand. The intuition here is that the more distinguishable the (featurised) states of two domain experts are, the less we tend to clarify them.

For those domain experts that have multiple sub-domains with different numbers of required and optional slots, the feature extraction procedure only applies to the latest active sub-domain.

In addition, note that, the confidence scores provided by the user intention identifier are only used as features for out-of-domain services. This is because in the current version of our system, the confidence estimation of the intention identifier for domain-dependent dialogue queries is less reliable due to the lack of context information. In contrast, the confidence scores returned by the domain experts will be more informative at this point.

5 Policy Learning with GPTD

In traditional statistical SDS, dialogue policies are usually trained using reinforcement learning based on simulated dialogue trajectories (Schatzmann et al., 2007; Keizer et al., 2010; Thomson and Young, 2010; Young et al., 2010). Although the evaluation of the simulators themselves could be an arguable issue, there are various advantages, e.g. hundreds of thousands of data examples can be easily generated for training and initial policy evaluation purposes, and different reinforcement learning models can be compared without incurring notable extra costs.

However, for more complex multi-domain SDS, particularly a voice assistant application like ours that aims at handling very complicated (ideally open-domain) dialogue scenarios, it would be difficult to develop a proper simulator that can reasonably mimic real human behaviours. Therefore, in this work, we learn the central control policy directly with human subjects, for which the following properties of the learning algorithm are required. Firstly and most importantly, the learner must be sample-efficient as the data collection procedure is costly. Secondly, the algorithm should support batch reinforcement learning. This is because when using function approximation, the learning process may not strictly converge, and the quality of the sequence of generated policies tends to oscillate after a certain number of improving steps at the beginning (Bertsekas and Tsitsiklis, 1996). If online reinforcement learning is used, we will be unable to evaluate the generated policy after each update, and hence will not know which policy to keep for the final evaluation. Therefore, we do a batch policy update and iterate the learning process for a number of batches, such that the data collection phase in a new iteration yields an evaluation of the policy obtained from the previous iteration at the same time.

To fulfill the above two requirements, the Gaussian Process Temporal Difference (GPTD) algorithm (Engel et al., 2005) is a proper choice, due to its sample efficiency (Fard et al., 2011) and batch learning ability (Engel et al., 2005), as well as its previous success in dialogue policy learning with human subjects (Gašić et al., 2013a). Note that, GPTD can also admit recursive (online) computations, but here we focus ourselves on the batch version.

A Gaussian Process (GP) is a generative model of Bayesian inference that can be used for function regression, and has the superiority of obtaining good posterior estimates with just a few observations (Rasmussen and Williams, 2006). GPTD models the Q-function as a zero mean GP which defines correlations in different parts of the featurised state and action spaces through a kernel function κ, as:

$$Q(\mathbf{s}, a(x)) \sim \mathcal{GP}(0, \kappa((\mathbf{s}_x, a), (\mathbf{s}_x, a))) \quad (5)$$

Given a sequence of t state-action pairs $\mathbf{X}_t = [(\mathbf{s}^0, a^0(x^0)), \ldots, (\mathbf{s}^t, a^t(x^t))]$ from a collection of dialogues and their corresponding immediate rewards $\mathbf{r}_t = [r^0, \ldots, r^t]$, the posterior of $Q(\mathbf{s}, a(x))$ for an arbitrary new state-action pair $(\mathbf{s}, a(x))$ can be computed as:

$$\begin{aligned} &Q(\mathbf{s}, a(x))|_{\mathbf{X}_t, \mathbf{r}_t} \\ &\quad \sim \mathcal{N}\left(\bar{Q}(\mathbf{s}, a(x)), \mathrm{cov}\left(\mathbf{s}, a(x)\right)\right) && (6) \\ &\bar{Q}(\mathbf{s}, a(x)) = \mathbf{k}_t(\mathbf{s}_x, a)^\top \mathbf{H}_t^\top \mathbf{G}_t^{-1} \mathbf{r}_t && (7) \\ &\mathrm{cov}\left(\mathbf{s}, a(x)\right) = \kappa((\mathbf{s}_x, a), (\mathbf{s}_x, a)) \\ &\quad - \mathbf{k}_t(\mathbf{s}_x, a)^\top \mathbf{H}_t^\top \mathbf{G}_t^{-1} \mathbf{H}_t \mathbf{k}_t(\mathbf{s}_x, a) && (8) \\ &\mathbf{G}_t = \mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\top + \sigma^2 \mathbf{H}_t \mathbf{H}_t^\top && (9) \end{aligned}$$

$$\mathbf{H}_t = \begin{bmatrix} 1 & -\gamma & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & -\gamma \end{bmatrix} \quad (10)$$

where $\mathbf{K}_t$ is the Gram matrix with elements $\mathbf{K}_t(i,j) = \kappa((\mathbf{s}^i_{x^i}, a^i), (\mathbf{s}^j_{x^j}, a^j))$, $\mathbf{k}_t(\mathbf{s}_x, a) = [\kappa((\mathbf{s}^i_{x^i}, a^i), (\mathbf{s}_x, a))]^t_{i=0}$ is a vector, and σ is a hyperparameter specifying the diagonal covariance values of the zero-mean Gaussian noise. In addition, we use $\mathrm{cov}\,(\mathbf{s}, a(x))$ to denote (for short) the self-covariance $\mathrm{cov}\,(\mathbf{s}, a(x), \mathbf{s}, a(x))$.

In our case, as different feature functions ϕ_a are defined for different action types, the kernel function is defined to be:

$$\kappa((\mathbf{s}_x, a), (\mathbf{s}'_{x'}, a')) = [\![a = a']\!]\kappa_a(\mathbf{s}_x, \mathbf{s}'_{x'}) \quad (11)$$

where $[\![\cdot]\!]$ is an indicator function and κ_a is the kernel function defined corresponding to the feature function ϕ_a.

Given a state, a most straightforward policy is to select the action that corresponds to the maximum mean Q-value estimated by the GP. However, since the objective is to learn the Q-function associated with the optimal policy by interacting directly with users, the policy must exhibit some form of stochastic behaviour in order to explore alternatives during the process of learning. In this work, the strategy employed for the exploration-exploitation trade-off is that, during exploration, actions are chosen according to the variance of the GP estimate for the Q-function, and during exploitation, actions are chosen according to the mean. That is:

$$\pi(\mathbf{s}) = \begin{cases} \arg\max_{a(x)} \bar{Q}(\mathbf{s}, a(x)) : \text{w.p. } 1-\epsilon \\ \arg\max_{a(x)} \mathrm{cov}\,(\mathbf{s}, a(x)) : \text{w.p. } \epsilon \end{cases} \quad (12)$$

where $0 < \epsilon < 1$ is a pre-defined exploration rate, and will be exponentially reduced at each batch iteration during our learning process.

Note that, in practice, not all the actions are valid at every possible state. For example, if a domain expert d has never been activated during a dialogue and can neither process the user's current query, the actions with an operand d will be regarded as invalid at this state. When executing the policy, we only consider those valid actions for a given state.

Score	Interpretation
5	The domain selections are totally correct, and the entire dialogue flow is fluent.
4	The domain selections are totally correct, but the dialogue flow is slightly redundant.
3	There are accidental domain selections errors, or the dialogue flow is perceptually redundant.
2	There are frequent domain selections errors, or the dialogue flow is intolerably redundant.
1	Most domain selections are incorrect, or the dialogue is incompletable.

Table 2: The scoring standard in our experiments.

6 Experimental Results

6.1 Training

We use the batch version of GPTD as described in Section 5 to learn the central control policy with human subjects. There are three domain experts available in our current system, but during the training only two domains are used, which are the travel information domain and the restaurant search domain. We reserve a movie search domain for evaluating the generalisation property of the learnt policy (see Section 6.2). The learning process started from a hand-crafted policy. Then 15 experienced users[2] volunteered to contribute dialogue examples with multiple or compound goals (see Figure 4 for an instance), from whom we collected around 50∼70 dialogues per day for 5 days[3]. After each dialogue, the users were asked to score the system from 5 to 1 according to a scoring standard shown in Table 2. The scores were taken as the (delayed) rewards to train the GPTD model, where we set the rewards for intermediate turns to 0. The working policy was updated daily based on the data obtained up to that day. The data collected on the first day was used for preliminary experiments to choose the hyperparame-

[2]Overall user satisfactions may rely on various aspects of the entire system, e.g. the data source quality of the services, the performance of each domain expert, etc. It will be difficult to make non-experienced users to score the central controller isolatedly.

[3]Not all the users participated the experiments everyday. There were 311 valid dialogues received in total, with an average length of 9 turns.

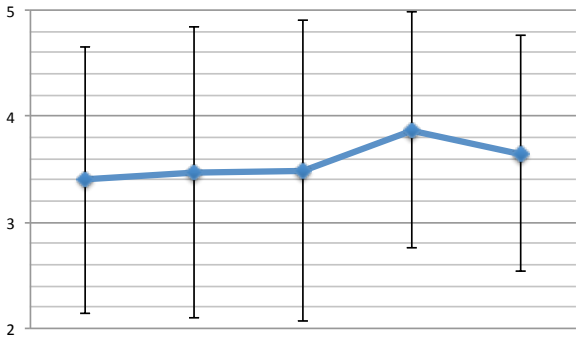


Figure 2: Average scores and standard deviations during policy iteration.

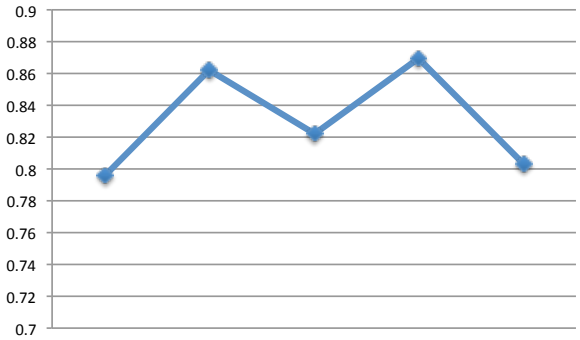


Figure 3: Domain selection accuracies during policy iteration.

ters of the model, such as the kernel function, the kernel parameters (if applicable), and σ and γ in the GPTD model. We initially experimented with linear, polynomial and Gaussian kernels, with different configurations of σ and γ values, as well as kernel parameter values. It was found that the linear kernel in combination with $\sigma = 5$ and $\gamma = 0.99$ works more appropriate than the other settings. This configuration was then fixed for the rest iterations.

The learning process was iterated for 4 days after the first one. On each day, we computed the mean and standard deviation of the user scores as an evaluation of the policy learnt on the previous day. The learning curve is illustrated in Figure 2. Note here, as we were actually executing a stochastic policy according to Eq. (12), to calculate the values in Figure 2 we ignored those dialogues that contain any actions selected due to the exploration. We also manually labelled the correctness of domain selection at every turn of the dialogues. The domain selection accuracies of the obtained policy sequence are shown in Figure 3, where similarly, those exploration actions as well as the clarification and confirmation actions were excluded from the calculations. Although the domain selection accuracy is not the target that our learning algorithm aims to optimise, it reflects the quality of the learnt policies from a different angle of view.

Scenario	Policy		p-value
	Baseline	GPTD	
(i)	4.5±0.8	4.2±0.8	0.387
(ii)	3.4±0.9	4.2±0.8	0.018
(iii)	4.1±1.0	4.3±1.0	0.0821
(iv)	3.9±1.1	4.5±0.8	0.0440

Table 3: Paired comparison experiments between the system with a trained GPTD policy and the rule-based baseline.

It can be found in Figure 2 that the best policy was obtained in the third iteration, and after that the policy quality oscillated. The same finding is indicated in Figure 3 as well, when we use the domain selection accuracy as the evaluation metric. Therefore, we kept the policy corresponding to the peak point of the learning curve for the comparison experiments below.

6.2 Comparison Experiments

We conducted paired comparison experiments in four scenarios to compare between the system with the GPTD-learnt central control policy and a non-trivial baseline. The baseline is a publicly deployed version of the voice assistant application. The central control policy of the baseline system is handcrafted, which has a separate list of semantic matching rules for each domain to enable domain switching.

The first two scenarios aim to test the performance of the two systems on (i) switching between a domain expert and out-of-domain services, and (ii) switching between two domain experts, where only the two training domains (travel information and restaurant search) were considered. Scenarios (iii) and (iv) are similar to scenarios (i) and (ii) respectively, but at this time, the users were required to carry out the tests surrounding the movie search domain (which is addressed by a new domain expert not used in the training phase). There were 13 users who participated this experiment. In each scenario, every user was required to test the two systems with an identical goal and similar queries. After each test, the users were asked to

score the two systems separately according to the scoring standard in Table 2.

The average scores received by the two systems are shown in Table 3, where we also compute the statistical significance (the p-values) of the results based on paired t-tests. It can be found that the learnt policy works significantly better than the rule-based policy in scenarios (ii) and (iv), but in scenarios (i) and (iii) the differences between two systems are statistically insignificant. Moreover, the learnt policy preserves the extensibility of the entire system as expected, of which strong evidences are given by the results in scenarios (iii) and (iv).

6.3 Policy Analysis

To better understand the policy learnt by the GPTD model, we look into the obtained weight vectors, as shown in Table 4. It can be found that confidence score (#5) is an informative feature for all the system actions, while the relative turn of a domain being last activated (#7) is an additional strong evidence for a confirmation decision. In addition, the similarity between the dialogue completion status (#1 & #2) of two ambiguous domain experts and the relative turns of them being last confirmed (#8) tend to be extra dominating features for clarification decisions, besides the closeness of the confidence scores returned by the two domain experts.

A less noticeable but important phenomenon is observed for feature #6, i.e. the total number of active turns of a domain expert during a dialogue. Concretely, feature #6 has a small negative effect on presentation actions but a small positive contribution to confirmation actions. Such weights could correspond to the discount factor's penalty to long dialogues in the value function. However, it implies an unexpected effect in extreme cases, which we explain in detail as follows. Although the absolute weights for feature #6 are tiny for both presentation and confirmation actions, the feature value will grow linearly during a dialogue. Therefore, when a dialogue in a certain domain last rather long, there tend to be very frequent confirmations. A possible solution to this problem could be either ignoring feature #6 or twisting it to some nonlinear function, such that its value stops increasing at a certain threshold point. In addition, to cover sufficient amount of those "extreme" examples in the training phase could also be an alternative solution, as our current training set contains very few examples that exhibit extraordinary long domain persistence.

#	Feature Weights			
	present	confirm	clarify	
1	0.09	0.02	0.60	present_ood
2	0.20	0.29	0.53	
3	0.18	0.29	0.16	
4	-0.10	0.16	0.25	
5	0.75	0.57	0.54	
6	-0.02	0.11	0.13	
7	0.25	1.19	0.36	
8	-0.22	-0.19	0.69	
9	–	0.20	–	0.47
Bias	-1.79	–	–	-2.42

Table 4: Feature weights learnt by GPTD. See Table 1 for the meanings of the features.

7 Further Discussions

The proposed approach is a rather general framework to learn extensible central control policies for multi-domain SDS based on distributed architectures. It does not rely on any internal representations of those individual domain experts, as long as a unified featurisation of their dialogue states can be achieved.

However, from the entire system point of view, the current implementation is still preliminary. Particularly, the confirmation and clarification mechanisms are isolated, for which the surface realisations sometimes may sound stiff. This phenomenon explains one of the reasons that make the proposed system slightly less preferred by the users than the baseline in scenario (i), when the interaction flows are relative simple. A possible improvement here could be associating the confirmation and clarification actions in the central controller to the error handling mechanisms within each domain expert, and letting domain experts generate their own utterances on receiving a confirmation/clarification request from the central controller.

Online reinforcement learning with real user cases will be another undoubted direction of further improvement of our system. The key challenge here is to automatically estimate user's satisfactions, which will be transformed to the rewards for the reinforcement learners. Strong feedbacks such as clicks or actual order placements can be

collected. But to regress user's true satisfaction, other environment features must also be taken into account. Practical solutions are still an open issue at this stage, and are left to our future work.

8 Conclusion

In this paper, we introduce an MDP framework for learning domain selection policies in a complex multi-domain SDS. Standard problem formulation is modified with tied model parameters, so that the entire system is extensible and new domain experts can be easily integrated without re-training the policy. This expectation is confirmed by empirical experiments with human subjects, where the proposed system marginally beats a non-trivial baseline and demonstrates proper extensibility. Several possible improvements are discussed, which will be the central arc of our future research.

Acknowledgements

The research in this paper is supported by China's 973 Programme no. 2014CB340505. The first author is partially funded by the EC FP7 programme under grant agreement no. 287615 (PARLANCE) and a SICSA PECE grant. The authors would also like to thank Qiaoqiao She, Duo Cai and the HCI-APP group at Baidu for volunteering to participate in the human subject experiments.

USER/ASR: 去青岛的机票 (Flight ticket to Qingdao.)

SYSTEM: 好的，到青岛的飞机票。请问您什么时候离开？
自哪儿出发啊？
(Okay, flight ticket to Qingdao. When are you departing, and where are you departing from?)
[Travel Info. Domain Expert]

USER/ASR: 明天出发 (I' m leaving tomorrow.)

SYSTEM: 好的，明天到青岛的飞机票。您预计由哪儿出发啊？
(Okay, flight ticket to Qingdao, leaving tomorrow. Where are you departing from?)
[Travel Info. Domain Expert]

USER/ASR: 北京 (Beijing)

SYSTEM: [Results Displayed  - Travel Info. Domain Expert]

USER: 青岛有好吃的四川菜吗
(Are there any good Sichuan cuisines in Qingdao?)
ASR: 听到有好吃的酸的川菜吗
(ungrammatical utterance, partially with "delicious" and "Sichuan food".)

SYSTEM: 您要继续订机票吗？
(Do you want to continue with flight ticket booking?)
[Confirmation - Central Controller]

USER: 不是 (No)

SYSTEM: [Web Search Service: "听到有好吃的酸的川菜吗" ]

USER/ASR: 明天去青岛 (I' m going to Qingdao tomorrow.)

SYSTEM: [Results Displayed  - Travel Info. Domain Expert]

USER/ASR: 青岛机场附近的餐厅 (Restaurants near Qingdao Airport)

SYSTEM: 为您找到以下餐厅
(I found the following restaurants for you.)
[Results Displayed - Restaurant Search Domain Expert]

USER/ASR: 青岛明天会下雨吗? (Will it rain in Qingdao tomorrow?)

SYSTEM: 青岛明天阵雨，温度16℃至20℃，东风5-6级
(Qingdao is rainy tomorrow. The temperature is 16 to 20 degree centigrade with moderate east wind.)
[Results Displayed  - Weather Report Service]

Figure 4: An example dialogue containing multiple user goals. The icons stand for graphical user interface based nonverbal outputs.

References

Richard Ernest Bellman. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Dimitri P. Bertsekas and John N. Tsitsiklis. 1996. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

Dan Bohus and Alexander I. Rudnicky. 2009. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech and Language*, 23(3):332–361.

Yaakov Engel, Shie Mannor, and Ron Meir. 2005. Reinforcement learning with Gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 201–208.

Mahdi Milani Fard, Joelle Pineau, and Csaba Szepesvári. 2011. PAC-Bayesian policy evaluation for reinforcement learning. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 195–202.

Milica Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013a. Online policy optimisation of Bayesian spoken dialogue systems via human interaction. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8367–8371.

Milica Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013b. POMDP-based dialogue manager adaptation to extended domains. In *Proceedings of the 14th annual SIGdial Meeting on Discourse and Dialogue*, pages 214–222.

Thomas Robert Gruber, Adam John Cheyer, Dag

Kittlaus, Didier Rene Guzzoni, Christopher Dean Brigham, Richard Donald Giuli, Marcello Bastea-Forte, and Harry Joseph Saddler. 2012. Intelligent automated assistant. United States Patent No. US 20120245944 A1.

Dilek Z. Hakkani-Tür, Gokhan Tür, Larry P. Heck, Ashley Fidler, and Asli Çelikyilmaz. 2012. A discriminative classification-based approach to information state updates for a multi-domain dialog system. In *Proceedings of the 13th Annual Conference of the International Speech Communication Association (INTERSPEECH)*.

Filip Jurčíček, Blaise Thomson, and Steve Young. 2011. Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as POMDPs. *ACM Transactions on Speech and Language Processing*, 7(3):6:1–6:25.

Filip Jurčíček, Blaise Thomson, and Steve Young. 2012. Reinforcement learning for parameter estimation in statistical spoken dialogue systems. *Computer Speech & Language*, 26(3):168–192.

Simon Keizer, Milica Gašić, Filip Jurčíček, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. Parameter estimation for agenda-based user simulation. In *Proceedings of the 11th annual SIGdial Meeting on Discourse and Dialogue*, pages 116–123.

Kazunori Komatani, Naoyuki Kanda, Mikio Nakano, Kazuhiro Nakadai, Hiroshi Tsujino, Tetsuya Ogata, and Hiroshi G. Okuno. 2006. Multi-domain spoken dialogue system with extensibility and robustness against speech recognition errors. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, pages 9–17.

Oliver Lemon and Olivier Pietquin, editors. 2012. *Data-Driven Methods for Adaptive Spoken Dialogue Systems: Computational Learning for Conversational Interfaces*. Springer.

Bor-shen Lin, Hsin-min Wang, and Lin-Shan Lee. 1999. A distributed architecture for cooperative spoken dialogue agents with coherent dialogue state and history. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Danilo Mirkovic and Lawrence Cavedon. 2006. Dialogue management using scripts. United States Patent No. US 20060271351 A1.

Mikio Nakano, Shun Sato, Kazunori Komatani, Kyoko Matsuyama, Kotaro Funakoshi, and Hiroshi G. Okuno. 2011. A two-stage domain selection framework for extensible multi-domain spoken dialogue systems. In *Proceedings of the 12th annual SIGdial Meeting on Discourse and Dialogue*, pages 18–29.

Carl Edward Rasmussen and Christopher K. I. Williams, editors. 2006. *Gaussian Processes for Machine Learning*. MIT Press.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16(1):105–133.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech and Language*, 24(4):562–588.

David R. Traum and Staffan Larsson. 2003. The Information State approach to dialogue management. In Jan van Kuppevelt and Ronnie W. Smith, editors, *Current and New Directions in Discourse and Dialogue*, pages 325–353. Springer.

Jason D. Williams and Steve Young. 2007. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422.

Jason D. Williams, Iker Arizmendi, and Alistair Conkie. 2010. Demonstration of AT&T "Let's Go": A production-grade statistical spoken dialog system. In *Proceedings of the 3rd IEEE Workshop on Spoken Language Technology (SLT)*.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The Hidden Information State model: a practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language*, 24(2):150–174.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D. Williams. 2013. POMDP-based statistical spoken dialogue systems: a review. *Proceedings of the IEEE*, PP(99):1–20.

A Constituent-Based Approach to Argument Labeling with Joint Inference in Discourse Parsing

Fang Kong[1*] **Hwee Tou Ng**[2] **Guodong Zhou**[1]
[1] School of Computer Science and Technology, Soochow University, China
[2] Department of Computer Science, National University of Singapore
kongfang@suda.edu.cn nght@comp.nus.edu.sg gdzhou@suda.edu.cn

Abstract

Discourse parsing is a challenging task and plays a critical role in discourse analysis. In this paper, we focus on labeling full argument spans of discourse connectives in the Penn Discourse Treebank (PDTB). Previous studies cast this task as a linear tagging or subtree extraction problem. In this paper, we propose a novel constituent-based approach to argument labeling, which integrates the advantages of both linear tagging and subtree extraction. In particular, the proposed approach unifies intra- and inter-sentence cases by treating the immediately preceding sentence as a special constituent. Besides, a joint inference mechanism is introduced to incorporate global information across arguments into our constituent-based approach via integer linear programming. Evaluation on PDTB shows significant performance improvements of our constituent-based approach over the best state-of-the-art system. It also shows the effectiveness of our joint inference mechanism in modeling global information across arguments.

1 Introduction

Discourse parsing determines the internal structure of a text and identifies the discourse relations between its text units. It has attracted increasing attention in recent years due to its importance in text understanding, especially since the release of the Penn Discourse Treebank (PDTB) corpus (Prasad et al., 2008), which adds a layer of discourse annotations on top of the Penn Treebank (PTB) corpus (Marcus et al., 1993). As the largest available discourse corpus, the PDTB corpus has become the defacto benchmark in recent studies on discourse parsing.

Compared to connective identification and discourse relation classification in discourse parsing, the task of labeling full argument spans of discourse connectives is much harder and thus more challenging. For connective identification, Lin et al. (2014) achieved the performance of 95.76% and 93.62% in F-measure using gold-standard and automatic parse trees, respectively. For discourse relation classification, Lin et al. (2014) achieved the performance of 86.77% in F-measure on classifying discourse relations into 16 level 2 types. However, for argument labeling, Lin et al. (2014) only achieved the performance of 53.85% in F-measure using gold-standard parse trees and connectives, much lower than the inter-annotation agreement of 90.20% (Miltsakaki et al., 2004).

In this paper, we focus on argument labeling in the PDTB corpus. In particular, we propose a *novel* constituent-based approach to argument labeling which views constituents as candidate arguments. Besides, our approach unifies intra- and inter-sentence cases by treating the immediately preceding sentence as a special constituent. Finally, a joint inference mechanism is introduced to incorporate global information across arguments via integer linear programming. Evaluation on the PDTB corpus shows the effectiveness of our approach.

The rest of this paper is organized as follows. Section 2 briefly introduces the PDTB corpus. Related work on argument labeling is reviewed in Section 3. In Section 4, we describe our constituent-based approach to argument labeling. In Section 5, we present our joint inference mechanism via integer linear programming (ILP). Section 6 gives the experimental results and analysis. Finally, we conclude in Section 7.

[*] The research reported in this paper was carried out while Fang Kong was a research fellow at the National University of Singapore.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 68–77,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

2 Penn Discourse Treebank

As the first large-scale annotated corpus that follows the lexically grounded, predicate-argument approach in D-LTAG (Lexicalized Tree Adjoining Grammar for Discourse) (Webber, 2004), the PDTB regards a connective as the predicate of a discourse relation which takes exactly two text spans as its arguments. In particular, the text span that the connective is syntactically attached to is called Arg2, and the other is called Arg1.

Although discourse relations can be either explicitly or implicitly expressed in PDTB, this paper focuses only on explicit discourse relations that are explicitly signaled by discourse connectives. Example (1) shows an explicit discourse relation from the article wsj_2314 with connective so underlined, Arg1 span *italicized*, and Arg2 span **bolded**.

(1) *But its competitors have much broader business interests* **and** so **are better cushioned against price swings .**

Note that a connective and its arguments can appear in any relative order, and an argument can be arbitrarily far away from its corresponding connective. Although the position of Arg2 is fixed once the connective is located, Arg1 can occur in the same sentence as the connective (SS), in a sentence preceding that of the connective (PS), or in a sentence following that of the connective (FS), with proportions of 60.9%, 39.1%, and less than 0.1% respectively for explicit relations in the PDTB corpus (Prasad et al., 2008). Besides, out of all PS cases where Arg1 occurs in some preceding sentence, 79.9% of them are the exact immediately preceding sentence. As such, in this paper, we only consider the current sentence containing the connective and its immediately preceding sentence as the text span where Arg1 occurs, similar to what was done in (Lin et al., 2014).

3 Related Work

For argument labeling in discourse parsing on the PDTB corpus, the related work can be classified into two categories: locating parts of arguments, and labeling full argument spans.

As a representative on locating parts of arguments, Wellner and Pustejovsky (2007) proposed several machine learning approaches to identify the head words of the two arguments for discourse connectives. Following this work, Elwell and Baldridge (2008) combined general and connective specific rankers to improve the performance of labeling the head words of the two arguments. Prasad et al. (2010) proposed a set of heuristics to locate the position of the Arg1 sentences for inter-sentence cases. The limitation of locating parts of arguments, such as the positions and head words, is that it is only a partial solution to argument labeling in discourse parsing.

In comparison, labeling full argument spans can provide a complete solution to argument labeling in discourse parsing and has thus attracted increasing attention recently, adopting either a subtree extraction approach (Dinesh et al. (2005), Lin et al. (2014)) or a linear tagging approach (Ghosh et al. (2011)).

As a representative subtree extraction approach, Dinesh et al. (2005) proposed an automatic tree subtraction algorithm to locate argument spans for intra-sentential subordinating connectives. However, only dealing with intra-sentential subordinating connectives is not sufficient since they constitute only 40.93% of all cases. Instead, Lin et al. (2014) proposed a two-step approach. First, an argument position identifier was employed to locate the position of Arg1. For the PS case, it directly selects the immediately preceding sentence as Arg1. For other cases, an argument node identifier was employed to locate the Arg1- and Arg2-nodes. Next, a tree subtraction algorithm was used to extract the arguments. However, as pointed out in Dinesh et al. (2005), it is not necessarily the case that a connective, Arg1, or Arg2 is dominated by a single node in the parse tree (that is, it can be dominated by a set of nodes). Figure 1 shows the gold-standard parse tree corresponding to Example (1). It shows that Arg1 includes three nodes: [$_{CC}$ But], [$_{NP}$ its competitors], [$_{VP}$ have much broader business interests], and Arg2 includes two nodes: [$_{CC}$ and], [$_{VP}$ are better cushioned against price swings]. Therefore, such an argument node identifier has inherent shortcomings in labeling arguments. Besides, the errors propagated from the upstream argument position classifier may adversely affect the performance of the downstream argument node identifier.

As a representative linear tagging approach, Ghosh et al. (2011) cast argument labeling as a linear tagging task using conditional random fields. Ghosh et al. (2012) further improved the perfor-

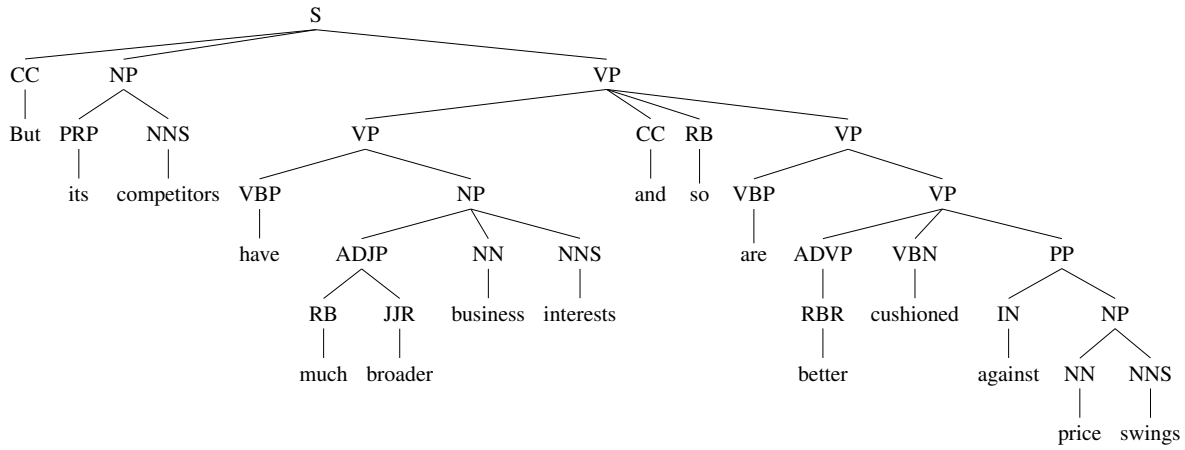


Figure 1: The gold-standard parse tree corresponding to Example (1)

mance with integration of the n-best results.

While the subtree extraction approach locates argument spans based on the nodes of a parse tree and is thus capable of using rich syntactic information, the linear tagging approach works on the tokens in a sentence and is thus capable of capturing local sequential dependency between tokens. In this paper, we take advantage of both subtree extraction and linear tagging approaches by proposing a novel constituent-based approach. Furthermore, intra- and inter-sentence cases are unified by treating the immediately preceding sentence as a special constituent. Finally, a joint inference mechanism is proposed to add global information across arguments.

4 A Constituent-Based Approach to Argument Labeling

Our constituent-based approach works by first casting the constituents extracted from a parse tree as argument candidates, then determining the role of every constituent as part of Arg1, Arg2, or NULL, and finally, merging all the constituents for Arg1 and Arg2 to obtain the Arg1 and Arg2 text spans respectively. Obviously, the key to the success of our constituent-based approach is constituent-based argument classification, which determines the role of every constituent argument candidate.

As stated above, the PDTB views a connective as the predicate of a discourse relation. Similar to semantic role labeling (SRL), for a given connective, the majority of the constituents in a parse tree may not be its arguments (Xue and Palmer, 2004). This indicates that negative instances (constituents marked NULL) may overwhelm positive instances. To address this problem, we use a simple algorithm to prune out these constituents which are clearly not arguments to the connective in question.

4.1 Pruning

The pruning algorithm works recursively in preprocessing, starting from the target connective node, i.e. the lowest node dominating the connective. First, all the siblings of the connective node are collected as candidates, then we move on to the parent of the connective node and collect its siblings, and so on until we reach the root of the parse tree. In addition, if the target connective node does not cover the connective exactly, the children of the target connective node are also collected.

For the example shown in Figure 1, we can locate the target connective node [$_{RB}$ so] and return five constituents — [$_{VP}$ have much broader business interests], [$_{CC}$ and], [$_{VP}$ are better cushioned against price swings], [$_{CC}$ But], and [$_{NP}$ its competitors] — as argument candidates.

It is not surprising that the pruning algorithm works better on gold parse trees than automatic parse trees. Using gold parse trees, our pruning algorithm can recall 89.56% and 92.98% (489 out of 546 Arg1s, 808 out of 869 Arg2s in the test data) of the Arg1 and Arg2 spans respectively and prune out 81.96% (16284 out of 19869) of the nodes in the parse trees. In comparison, when automatic parse trees (based on the Charniak parser (Charniak, 2000)) are used, our pruning algorithm can recall 80.59% and 89.87% of the Arg1 and Arg2 spans respectively and prune out 81.70% (16190

Feature	Description	Example
CON-Str	The string of the given connective (case-sensitive)	so
CON-LStr	The lowercase string of the given connective	so
CON-Cat	The syntactic category of the given connective: subordinating, coordinating, or discourse adverbial	Subordinating
CON-iLSib	Number of left siblings of the connective	2
CON-iRSib	Number of right siblings of the connective	1
NT-Ctx	The context of the constituent. We use POS combination of the constituent, its parent, left sibling and right sibling to represent the context. When there is no parent or siblings, it is marked NULL.	VP-VP-NULL-CC
CON-NT-Path	The path from the parent node of the connective to the node of the constituent	$RB \uparrow VP \downarrow VP$
CON-NT-Position	The position of the constituent relative to the connective: left, right, or previous	left
CON-NT-Path-iLsib	The path from the parent node of the connective to the node of the constituent and whether the number of left siblings of the connective is greater than one	$RB \uparrow VP \downarrow VP$:>1

Table 1: Features employed in argument classification.

out of 19816) of the nodes in the parse trees.

4.2 Argument Classification

In this paper, a multi-category classifier is employed to determine the role of an argument candidate (i.e., Arg1, Arg2, or NULL). Table 1 lists the features employed in argument classification, which reflect the properties of the connective and the candidate constituent, and the relationship between them. The third column of Table 1 shows the features corresponding to Figure 1, considering [$_{RB}$ so] as the given connective and [$_{VP}$ have much broader business interests] as the constituent in question.

Similar to Lin et al. (2014), we obtained the syntactic category of the connectives from the list provided in Knott (1996). However, different from Lin et al. (2014), only the siblings of the root path nodes (i.e., the nodes occurring in the path of the connective to root) are collected as the candidate constituents in the pruning stage, and the value of the relative position can be left or right, indicating that the constituent is located on the left- or right-hand of the root path respectively. Besides, we view the root of the previous sentence as a special candidate constituent. For example, the value of the feature CON-NT-Position is *previous* when the current constituent is the root of the previous sentence. Finally, we use the part-of-speech (POS) combination of the constituent itself, its parent node, left sibling node and right sibling node to represent the context of the candidate constituent. Intuitively, this information can help determine the role of the constituent.

For the example shown in Figure 1, we first employ the pruning algorithm to get the candidate constituents, and then employ our argument classifier to determine the role for every candidate. For example, if the five candidates are labeled as Arg1, Arg2, Arg2, Arg1, and Arg1, respectively, we merge all the Arg1 constituents to obtain the Arg1 text span (i.e., *But its competitors have much broader business interests*). Similarly, we merge the two Arg2 constituents to obtain the Arg2 text span (i.e., **and are better cushioned against price swings**).

5 Joint Inference via Integer Linear Programming

In the above approach, decisions are always made for each candidate independently, ignoring global information across candidates in the final output. For example, although an argument span can be split into multiple discontinuous segments (e.g., the Arg2 span of Example (1) contains two discontinuous segments, **and**, **are better cushioned against price swings**), the number of discontinuous segments is always limited. Statistics on the PDTB corpus shows that the number of discontin-

uous segments for both Arg1 and Arg2 is generally ($>= 99\%$) at most 2. For Example (1), from left to right, we can obtain the list of constituent candidates: [$_{CC}$ But], [$_{NP}$ its competitors], [$_{VP}$ have much broader business interests], [$_{CC}$ and], [$_{VP}$ are better cushioned against price swings]. If our argument classifier wrongly determines the roles as Arg1, Arg2, Arg1, Arg2, and Arg1 respectively, we can find that the achieved Arg1 span contains three discontinuous segments. Such errors may be corrected from a global perspective.

In this paper, a joint inference mechanism is introduced to incorporate various kinds of knowledge to resolve the inconsistencies in argument classification to ensure global legitimate predictions. In particular, the joint inference mechanism is formalized as a constrained optimization problem, represented as an integer linear programming (ILP) task. It takes as input the argument classifiers' confidence scores for each constituent candidate along with a list of constraints, and outputs the optimal solution that maximizes the objective function incorporating the confidence scores, subject to the constraints that encode various kinds of knowledge.

In this section, we meet the requirement of ILP with focus on the definition of variables, the objective function, and the problem-specific constraints, along with ILP-based joint inference integrating multiple systems.

5.1 Definition of Variables

Given an input sentence, the task of argument labeling is to determine what labels should be assigned to which constituents corresponding to which connective. It is therefore natural that encoding the output space of argument labeling requires various kinds of information about the connectives, the argument candidates corresponding to a connective, and their argument labels.

Given an input sentence s, we define following variables:

(1) P: the set of connectives in a sentence.

(2) $p \in P$: a connective in P.

(3) $C(p)$: the set of argument candidates corresponding to connective p. (i.e., the parse tree nodes obtained in the pruning stage).

(4) $c \in C(p)$: an argument candidate.

(5) L: the set of argument labels {Arg1, Arg2, NULL }.

(6) $l \in L$: an argument label in L.

In addition, we define the integer variables as follows:

$$Z^l_{c,p} \in \{0, 1\} \tag{1}$$

If $Z^l_{c,p} = 1$, the argument candidate c, which corresponds to connective p, should be assigned the label l. Otherwise, the argument candidate c is not assigned this label.

5.2 The Objective Function

The objective of joint inference is to find the best arguments for all the connectives in one sentence. For every connective, the pruning algorithm is first employed to determine the set of corresponding argument candidates. Then, the argument classifier is used to assign a label to every candidate. For an individual labeling $Z^l_{c,p}$, we measure the quality based on the confidence scores, $f_{l,c,p}$, returned by the argument classifier. Thus, the objective function can be defined as

$$\max \sum_{l,c,p} f_{l,c,p} Z^l_{c,p} \tag{2}$$

5.3 Constraints

As the key to the success of ILP-based joint inference, the following constraints are employed:

Constraint 1: The arguments corresponding to a connective cannot overlap with the connective. Let $c_1, c_2..., c_k$ be the argument candidates that correspond to the same connective and overlap with the connective in a sentence.[1] Then this constraint ensures that none of them will be assigned as Arg1 or Arg2.

$$\sum_{i=1}^{k} Z^{NULL}_{c_i,p} = k \tag{3}$$

Constraint 2: There are no overlapping or embedding arguments. Let $c_1, c_2..., c_k$ be the argument candidates that correspond to the same connective and cover the same word in a sentence.[2]

[1] Only when the target connective node does not cover the connective exactly and our pruning algorithm collects all the children of the target connective node as part of constituent candidates, such overlap can be introduced.

[2] This constraint only works in system combination of Section 5.4, where additional phantom candidates may introduce such overlap.

Then this constraint ensures that at most one of the constituents can be assigned as Arg1 or Arg2. That is, at least $k-1$ constituents should be assigned the special label NULL.

$$\sum_{i=1}^{k} Z_{c_i,p}^{NULL} \geq k-1 \quad (4)$$

Constraint 3: For a connective, there is at least one constituent candidate assigned as Arg2.

$$\sum_{c} Z_{c,p}^{Arg2} \geq 1 \quad (5)$$

Constraint 4: Since we view the previous complete sentence as a special Arg1 constituent candidate, denoted as m, there is at least one candidate assigned as Arg1 for every connective.

$$\sum_{c} Z_{c,p}^{Arg1} + Z_{m,p}^{Arg1} \geq 1 \quad (6)$$

Constraint 5: The number of discontinuous constituents assigned as Arg1 or Arg2 should be at most 2. That is, if argument candidates $c_1, c_2..., c_k$ corresponding to the same connective are discontinuous, this constraint ensures that at most two of the constituents can be assigned the same label Arg1 or Arg2.

$$\sum_{i=1}^{k} Z_{c_i,p}^{Arg1} \leq 2, and \sum_{i=1}^{k} Z_{c_i,p}^{Arg2} \leq 2 \quad (7)$$

5.4 System Combination

Previous work shows that the performance of argument labeling heavily depends on the quality of the syntactic parser. It is natural that combining different argument labeling systems on different parse trees can potentially improve the overall performance of argument labeling.

To explore this potential, we build two argument labeling systems — one using the Berkeley parser (Petrov et al., 2006) and the other the Charniak parser (Charniak, 2000). Previous studies show that these two syntactic parsers tend to produce different parse trees for the same sentence (Zhang et al., 2009). For example, our preliminary experiment shows that applying the pruning algorithm on the output of the Charniak parser produces a list of candidates with recall of 80.59% and 89.87% for Arg1 and Arg2 respectively, while achieving recall of 78.6% and 91.1% for Arg1 and Arg2 respectively on the output of the Berkeley parser. It also shows that combining these two candidate lists significantly improves recall to 85.7% and 93.0% for Arg1 and Arg2, respectively.

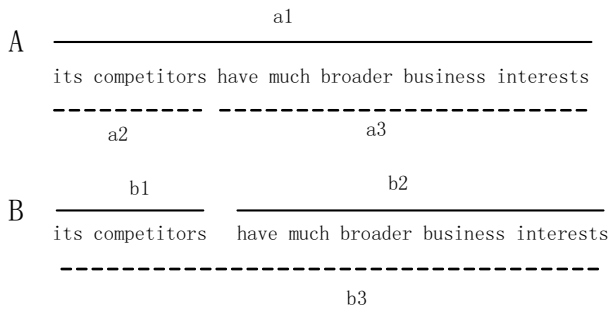


Figure 2: An example on unifying different candidates.

In subsection 5.2, we only consider the confidence scores returned by an argument classifier. Here, we proceed to combine the probabilities produced by two argument classifiers. There are two remaining problems to resolve:

- How do we unify the two candidate lists? In principle, constituents spanning the same sequence of words should be viewed as the same candidate. That is, for different candidates, we can unify them by adding phantom candidates. This is similar to the approach proposed by Punyakanok et al. (2008) for the semantic role labeling task. For example, Figure 2 shows the candidate lists generated by our pruning algorithm based on two different parse trees given the segment "its competitors have much broader business interests". Dashed lines are used for phantom candidates and solid lines for true candidates. Here, system A produces one candidate a1, with two phantom candidates a2 and a3 added. Analogously, phantom candidate b3 is added to the candidate list output by System B. In this way, we can get the unified candidate list: "its competitors have much broader business interests", "its competitors", "have much broader business interests".

- How do we compute the confidence score for every decision? For every candidate in the unified list, we first determine whether it is a true candidate based on the specific parse tree. Then, for a true candidate, we extract the features from the corresponding parse

tree. On this basis, we can determine the confidence score using our argument classifier. For a phantom candidate, we set the same prior distribution as the confidence score. In particular, the probability of the "NULL" class is set to 0.55, following (Punyakanok et al., 2008), and the probabilities of Arg1 and Arg2 are set to their occurrence frequencies in the training data. For the example shown in Figure 2, since System A returns "its competitors have much broader business interests" as a true candidate, we can obtain its confidence score using our argument classifier. For the two phantom candidates — "its competitors" and "have much broader business interests" — we use the prior distributions directly. This applies to the candidates for System B. Finally, we simply average the estimated probabilities to determine the final probability estimate for every candidate in the unified list.

6 Experiments

In this section, we systematically evaluate our constituent-based approach with a joint inference mechanism to argument labeling on the PDTB corpus.

6.1 Experimental settings

All our classifiers are trained using the OpenNLP maximum entropy package[3] with the default parameters (i.e. without smoothing and with 100 iterations). As the PDTB corpus is aligned with the PTB corpus, the gold parse trees and sentence boundaries are obtained from PTB. Under the automatic setting, the NIST sentence segmenter[4] and the Charniak parser[5] are used to segment and parse the sentences, respectively. lp_solve[6] is used for our joint inference.

This paper focuses on automatically labeling the full argument spans of discourse connectives. For a fair comparison with start-of-the-art systems, we use the NUS PDTB-style end-to-end discourse parser[7] to perform other subtasks of discourse parsing except argument labeling, which includes connective identification, non-explicit discourse relation identification and classification.

Finally, we evaluate our system on two aspects: (1) the dependence on the parse trees (GS/Auto, using gold standard or automatic parse trees and sentence boundaries); and (2) the impact of errors propagated from previous components (noEP/EP, using gold annotation or automatic results from previous components). In combination, we have four different settings: GS+noEP, GS+EP, Auto+noEP and Auto+EP. Same as Lin et al. (2014), we report exact match results under these four settings. Here, exact match means two spans match identically, except beginning or ending punctuation symbols.

6.2 Experimental results

We first evaluate the effectiveness of our constituent-based approach by comparing our system with the state-of-the-art systems, ignoring the joint inference mechanism. Then, the contribution of the joint inference mechanism to our constituent-based approach, and finally the contribution of our argument labeling system to the end-to-end discourse parser are presented.

Effectiveness of our constituent-based approach

By comparing with two state-of-the-art argument labeling approaches, we determine the effectiveness of our constituent-based approach.

Comparison with the linear tagging approach

As a representative linear tagging approach, Ghosh et al. (2011; 2012; 2012) only reported the exact match results for Arg1 and Arg2 using the evaluation script for chunking evaluation[8] under GS+noEP setting with Section 02–22 of the PDTB corpus for training, Section 23–24 for testing, and Section 00–01 for development. It is also worth mentioning that an argument span can contain multiple discontinuous segments (i.e., chunks), so chunking evaluation only shows the exact match of every argument segment but not the exact match of every argument span. In order to fairly compare our system with theirs, we evaluate our system using both the exact metric and the chunking evaluation. Table 2 compares the results of our system without joint inference and the results reported by Ghosh et al. (2012) on the same data split. We can find that our system performs much bet-

[3] http://maxent.sourceforge.net/

[4] http://duc.nist.gov/duc2004/software/duc2003.breakSent.tar.gz

[5] ftp://ftp.cs.brown.edu/pub/nlparser/

[6] http://lpsolve.sourceforge.net/

[7] http://wing.comp.nus.edu.sg/ linzihen/parser/

[8] http://www.cnts.ua.ac.be/conll2000/chunking/conlleval.txt

ter than Ghosh's on both Arg1 and Arg2, even on much stricter metrics.

Systems	Arg1	Arg2
ours using exact match	65.68	84.50
ours using chunking evaluation	67.48	88.08
reported by Ghosh et al. (2012)	59.39	79.48

Table 2: Performance (F1) comparison of our argument labeling approach with the linear tagging approach as adopted in Ghosh et al. (2012)

Comparison with the subtree extracting approach

For a fair comparison, we also conduct our experiments on the same data split of Lin et al. (2014) with Section 02 to 21 for training, Section 22 for development, and Section 23 for testing. Table 3 compares our labeling system without joint inference with Lin et al. (2014), a representative subtree extracting approach. From the results, we find that the performance of our argument labeling system significantly improves under all settings. This is because Lin et al. (2014) considered all the internal nodes of the parse trees, whereas the pruning algorithm in our approach can effectively filter out those unlikely constituents when determining Arg1 and Arg2.

	Setting	Arg1	Arg2	Arg1&2
ours	GS+noEP	62.84	84.07	55.69
	GS+EP	61.46	81.30	54.31
	Auto+EP	56.04	76.53	48.89
Lin's	GS+noEP	59.15	82.23	53.85
	GS+EP	57.64	79.80	52.29
	Auto+EP	47.68	70.27	40.37

Table 3: Performance (F1) comparison of our argument labeling approach with the subtree extraction approach as adopted in Lin et al. (2014)

As justified above, by integrating the advantages of both linear tagging and subtree extraction, our constituent-based approach can capture both rich syntactic information from parse trees and local sequential dependency between tokens. The results show that our constituent-based approach indeed significantly improves the performance of argument labeling, compared to both linear tagging and subtree extracting approaches.

Contribution of Joint Inference

Same as Lin et al. (2014), we conduct our experiments using Section 02 to 21 for training, Section 22 for development, and Section 23 for testing. Table 4 lists the performance of our argument labeling system without and with ILP inference under four different settings, while Table 5 reports the contribution of system combination. It shows the following:

- On the performance comparison of Arg1 and Arg2, the performance on Arg2 is much better than that on Arg1 with the performance gap up to 8% under different settings. This is due to the fact that the relationship between Arg2 and the connective is much closer. This result is also consistent with previous studies on argument labeling.

- On the impact of error propagation from connective identification, the errors propagated from connective identification reduce the performance of argument labeling by less than 2% in both Arg1 and Arg2 F-measure under different settings.

- On the impact of parse trees, using automatic parse trees reduces the performance of argument labeling by about 5.5% in both Arg1 and Arg2 F-measure under different settings. In comparison with the impact of error propagation, parse trees have much more impact on argument labeling.

- On the impact of joint inference, it improves the performance of argument labeling, especially on automatic parse trees by about 2%.[9]

- On the impact of system combination, the performance is improved by about 1.5%.

	Setting	Arg1	Arg2	Arg1&2
without Joint Inference	GS+noEP	62.84	84.07	55.69
	GS+EP	61.46	81.30	54.31
	Auto+noEP	57.75	79.85	50.27
	Auto+EP	56.04	76.53	48.89
with Joint Inference	GS+noEP	65.76	83.86	58.18
	GS+EP	63.96	81.19	56.37
	Auto+noEP	60.24	79.74	52.55
	Auto+EP	58.10	76.53	50.73

Table 4: Performance (F1) of our argument labeling approach.

Contribution to the end-to-end discourse parser

[9]Unless otherwise specified, all the improvements in this paper are significant with $p < 0.001$.

Systems	Setting	Arg1	Arg2	Arg1&2
Charniak	noEP	60.24	79.74	52.55
	EP	58.10	76.53	50.73
Berkeley	noEP	60.78	80.07	52.98
	EP	58.80	77.21	51.43
Combined	noEP	61.97	80.61	54.50
	EP	59.72	77.55	52.52

Table 5: Contribution of System Combination in Joint Inference.

Lastly, we focus on the contribution of our argument labeling approach to the overall performance of the end-to-end discourse parser. This is done by replacing the argument labeling model of the NUS PDTB-style end-to-end discourse parser with our argument labeling model. Table 6 shows the results using gold parse trees and automatic parse trees, respectively.[10] From the results, we find that using gold parse trees, our argument labeling approach significantly improves the performance of the end-to-end system by about 1.8% in F-measure, while using automatic parse trees, the improvement significantly enlarges to 6.7% in F-measure.

Setting	New d-parser	Lin et al.'s (2014)
GS	34.80	33.00
Auto	27.39	20.64

Table 6: Performance (F1) of the end-to-end discourse parser.

7 Conclusion

In this paper, we focus on the problem of automatically labeling the full argument spans of discourse connectives. In particular, we propose a constituent-based approach to integrate the advantages of both subtree extraction and linear tagging approaches. Moreover, our proposed approach integrates inter- and intra-sentence argument labeling by viewing the immediately preceding sentence as a special constituent. Finally, a joint inference mechanism is introduced to incorporate global information across arguments into our constituent-based approach via integer linear programming.

[10]Further analysis found that the error propagated from sentence segmentation can reduce the performance of the end-to-end discourse parser. Retraining the NIST sentence segmenter using Section 02 to 21 of the PDTB corpus, the original NUS PDTB-style end-to-end discourse parser can achieve the performance of 25.25% in F-measure, while the new version (i.e. replace the argument labeling model with our argument labeling model) can achieve the performance of 30.06% in F-measure.

Acknowledgments

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office. This research is also partially supported by Key project 61333018 and 61331011 under the National Natural Science Foundation of China.

References

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2005. Attribution and the (non-)alignment of syntactic and discourse arguments of connectives. In *Proceedings of the Workshop on Frontiers in Corpus Annotation II: Pie in the Sky*, pages 29–36.

Robert Elwell and Jason Baldridge. 2008. Discourse connective argument identification with connective specific rankers. In *Second IEEE International Conference on Semantic Computing*, pages 198–205.

Sucheta Ghosh, Richard Johansson, Giuseppe Riccardi, and Sara Tonelli. 2011. Shallow discourse parsing with conditional random fields. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 1071–1079.

Sucheta Ghosh, Giuseppe Riccardi, and Richard Johansson. 2012. Global features for shallow discourse parsing. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 150–159.

Sucheta Ghosh. 2012. *End-to-End Discourse Parsing using Cascaded Structured Prediction*. Ph.D. thesis, University of Trento.

Alistair Knott. 1996. *A Data-Driven Methodology for Motivating a Set of Coherence Relations*. Ph.D. thesis, University of Edinburgh.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A PDTB-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2):151–184.

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

Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2004. The Penn Discourse Treebank. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, pages 2237–2240.

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

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the LREC 2008 Conference*, pages 2961–2968.

Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2010. Exploiting scope for shallow discourse parsing. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The important of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Bonnie Webber. 2004. D-LTAG: extending lexicalized TAG to discourse. *Cognitive Science*, 28(5):751–779.

Ben Wellner and James Pustejovsky. 2007. Automatically identifying the arguments of discourse connectives. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 92–101.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing*, pages 88–94.

Hui Zhang, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. K-best combination of syntactic parsers. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1552–1560.

Strongly Incremental Repair Detection

Julian Hough[1,2]
[1]Dialogue Systems Group
Faculty of Linguistics
and Literature
Bielefeld University
julian.hough@uni-bielefeld.de

Matthew Purver[2]
[2]Cognitive Science Research Group
School of Electronic Engineering
and Computer Science
Queen Mary University of London
m.purver@qmul.ac.uk

Abstract

We present STIR (STrongly Incremental Repair detection), a system that detects speech repairs and edit terms on transcripts incrementally with minimal latency. STIR uses information-theoretic measures from n-gram models as its principal decision features in a pipeline of classifiers detecting the different stages of repairs. Results on the Switchboard disfluency tagged corpus show utterance-final accuracy on a par with state-of-the-art incremental repair detection methods, but with better incremental accuracy, faster time-to-detection and less computational overhead. We evaluate its performance using incremental metrics and propose new repair processing evaluation standards.

1 Introduction

Self-repairs in spontaneous speech are annotated according to a well established three-phase structure from (Shriberg, 1994) onwards, and as described in Meteer et al. (1995)'s Switchboard corpus annotation handbook:

$$\text{John} \underbrace{[\ \text{likes}}_{\text{reparandum}} + \underbrace{\{\text{uh}\}}_{\text{interregnum}} \underbrace{\text{loves}\ ]}_{\text{repair}} \text{Mary} \quad (1)$$

From a dialogue systems perspective, detecting repairs and assigning them the appropriate structure is vital for robust natural language understanding (NLU) in interactive systems. Downgrading the commitment of *reparandum* phases and assigning appropriate *interregnum* and *repair* phases permits computation of the user's intended meaning.

Furthermore, the recent focus on *incremental* dialogue systems (see e.g. (Rieser and Schlangen, 2011)) means that repair detection should operate without unnecessary processing overhead, and function efficiently within an incremental framework. However, such left-to-right operability on its own is not sufficient: in line with the principle of strong incremental interpretation (Milward, 1991), a repair detector should give *the best results possible as early as possible*. With one exception (Zwarts et al., 2010), there has been no focus on evaluating or improving the *incremental performance* of repair detection.

In this paper we present STIR (Strongly Incremental Repair detection), a system which addresses the challenges of incremental accuracy, computational complexity and latency in self-repair detection, by making local decisions based on relatively simple measures of fluency and similarity. Section 2 reviews state-of-the-art methods; Section 3 summarizes the challenges and explains our general approach; Section 4 explains STIR in detail; Section 5 explains our experimental set-up and novel evaluation metrics; Section 6 presents and discusses our results and Section 7 concludes.

2 Previous work

Qian and Liu (2013) achieve the state of the art in Switchboard corpus self-repair detection, with an F-score for detecting reparandum words of 0.841 using a three-step weighted Max-Margin Markov network approach. Similarly, Georgila (2009) uses Integer Linear Programming post-processing of a CRF to achieve F-scores over 0.8 for reparandum start and repair start detection. However neither approach can operate incrementally.

Recently, there has been increased interest in left-to-right repair detection: Rasooli and Tetreault (2014) and Honnibal and Johnson (2014) present dependency parsing systems with reparandum detection which perform similarly, the latter equalling Qian and Liu (2013)'s F-score at 0.841. However, while operating left-to-right, these systems are not designed or evaluated for their *incremental* performance. The use of beam search over

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 78–89,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

different repair hypotheses in (Honnibal and Johnson, 2014) is likely to lead to unstable repair label sequences, and they report repair hypothesis 'jitter'. Both of these systems use a non-monotonic dependency parsing approach that immediately removes the reparandum from the linguistic analysis of the utterance in terms of its dependency structure and repair-reparandum correspondence, which from a downstream NLU module's perspective is undesirable. Heeman and Allen (1999) and Miller and Schuler (2008) present earlier left-to-right operational detectors which are less accurate and again give no indication of the incremental performance of their systems. While Heeman and Allen (1999) rely on repair structure template detection coupled with a multi-knowledge-source language model, the rarity of the tail of repair structures is likely to be the reason for lower performance: Hough and Purver (2013) show that only 39% of repair alignment structures appear at least twice in Switchboard, supported by the 29% reported by Heeman and Allen (1999) on the smaller TRAINS corpus. Miller and Schuler (2008)'s encoding of repairs into a grammar also causes sparsity in training: repair is a general processing strategy not restricted to certain lexical items or POS tag sequences.

The model we consider most suitable for incremental dialogue systems so far is Zwarts et al. (2010)'s incremental version of Johnson and Charniak (2004)'s noisy channel repair detector, as it incrementally applies structural repair analyses (rather than just identifying reparanda) and is evaluated for its incremental properties. Following (Johnson and Charniak, 2004), their system uses an n-gram language model trained on roughly 100K utterances of reparandum-excised ('cleaned') Switchboard data. Its channel model is a statistically-trained S-TAG parser whose grammar has simple reparandum-repair alignment rule categories for its non-terminals (copy, delete, insert, substitute) and words for its terminals. The parser hypothesises all possible repair structures for the string consumed so far in a chart, before pruning the unlikely ones. It performs equally well to the non-incremental model by the end of each utterance (F-score = 0.778), and can make detections early via the addition of a speculative next-word repair completion category to their S-TAG non-terminals. In terms of incremental performance, they report the novel evaluation metric of *time-to-detection* for correctly identified repairs, achieving an average of 7.5 words from the start of the reparandum and 4.6 from the start of the repair phase. They also introduce *delayed accuracy*, a word-by-word evaluation against gold-standard disfluency tags up to the word before the current word being consumed (in their terms, the *prefix boundary*), giving a measure of the stability of the repair hypotheses. They report an F-score of 0.578 at one word back from the current prefix boundary, increasing word-by-word until 6 words back where it reaches 0.770. These results are the point-of-departure for our work.

3 Challenges and Approach

In this section we summarize the challenges for incremental repair detection: computational complexity, repair hypothesis stability, latency of detection and repair structure identification. In 3.1 we explain how we address these.

Computational complexity Approaches to detecting repair structures often use chart storage (Zwarts et al., 2010; Johnson and Charniak, 2004; Heeman and Allen, 1999), which poses a computational overhead: if considering all possible boundary points for a repair structure's 3 phases beginning on any word, for prefixes of length n the number of hypotheses can grow in the order $O(n^4)$. Exploring a subset of this space is necessary for assigning entire repair structures as in (1) above, rather than just detecting reparanda: the (Johnson and Charniak, 2004; Zwarts et al., 2010) noisy-channel detector is the only system that applies such structures but the potential runtime complexity in decoding these with their S-TAG repair parser is $O(n^5)$. In their approach, complexity is mitigated by imposing a maximum repair length (12 words), and also by using beam search with re-ranking (Lease et al., 2006; Zwarts and Johnson, 2011). If we wish to include full decoding of the repair's structure (as argued by Hough and Purver (2013) as necessary for full interpretation) whilst taking a strictly incremental and time-critical perspective, reducing this complexity by minimizing the size of this search space is crucial.

Stability of repair hypotheses and latency Using a beam search of n-best hypotheses on a word-by-word basis can cause 'jitter' in the detector's output. While utterance-final accuracy is desired,

for a truly incremental system good intermediate results are equally important. Zwarts et al. (2010)'s time-to-detection results show their system is only certain about a detection after processing the entire repair. This may be due to the string alignment-inspired S-TAG that matches repair and reparanda: a 'rough copy' dependency only becomes likely once the entire repair has been consumed. The latency of 4.6 words to detection and a relatively slow rise to utterance-final accuracy up to 6 words back is undesirable given repairs have a mean reparandum length of ≈1.5 words (Hough and Purver, 2013; Shriberg and Stolcke, 1998).

Structural identification Classifying repairs has been ignored in repair processing, despite the presence of distinct categories (e.g. repeats, substitutions, deletes) with different pragmatic effects (Hough and Purver, 2013).[1] This is perhaps due to lack of clarity in definition: even for human annotators, verbatim repeats withstanding, agreement is often poor (Hough and Purver, 2013; Shriberg, 1994). Assigning and evaluating repair (not just reparandum) structures will allow repair interpretation in future; however, work to date evaluates only reparandum detection.

3.1 Our approach

To address the above, we propose an alternative to (Johnson and Charniak, 2004; Zwarts et al., 2010)'s noisy channel model. While the model elegantly captures intuitions about parallelism in repairs and modelling fluency, it relies on string-matching, motivated in a similar way to automatic spelling correction (Brill and Moore, 2000): it assumes a speaker chooses to utter fluent utterance X according to some prior distribution $P(X)$, but a noisy channel causes them instead to utter a noisy Y according to channel model $P(Y \mid X)$. Estimating $P(Y \mid X)$ directly from observed data is difficult due to sparsity of repair instances, so a transducer is trained on the rough copy alignments between reparandum and repair. This approach succeeds because repetition and simple substitution repairs are very common; but repair as a psychological process is not driven by string alignment, and deletes, restarts and rarer substitution forms are not captured. Furthermore, the noisy channel model assumes an inherently utterance-global process for generating (and therefore finding) an underlying 'clean' string — much as similar spelling correction models are word-global — we instead take a very local perspective here.

In accordance with psycholinguistic evidence (Brennan and Schober, 2001), we assume characteristics of the repair onset allow hearers to detect it very quickly and solve the *continuation problem* (Levelt, 1983) of integrating the repair into their linguistic context immediately, before processing or even hearing the end of the repair phase. While repair onsets may take the form of interregna, this is not a reliable signal, occurring in only ≈15% of repairs (Hough and Purver, 2013; Heeman and Allen, 1999). Our repair onset detection is therefore driven by departures from fluency, via information-theoretic features derived incrementally from a language model in line with recent psycholinguistic accounts of incremental parsing – see (Keller, 2004; Jaeger and Tily, 2011).

Considering the time-linear way a repair is processed and the fact speakers are exponentially less likely to trace one word further back in repair as utterance length increases (Shriberg and Stolcke, 1998), backwards search seems to be the most efficient reparandum extent detection method.[2] Features determining the detection of the reparandum extent in the backwards search can also be information-theoretic: entropy measures of distributional parallelism can characterize not only rough copy dependencies, but distributionally similar or dissimilar correspondences between sequences. Finally, when detecting the repair end and structure, distributional information allows computation of the similarity between reparandum and repair. We argue a local-detection-with-backtracking approach is more cognitively plausible than string-based left-to-right repair labelling, and using this insight should allow an improvement in incremental accuracy, stability and time-to-detection over string-alignment driven approaches in repair detection.

4 STIR: Strongly Incremental Repair detection

Our system, STIR (Strongly Incremental Repair detection), therefore takes a local incremental ap-

[1] Though see (Germesin et al., 2008) for one approach, albeit using idiosyncratic repair categories.

[2] We acknowledge a purely position-based model for reparandum extent detection under-estimates prepositions, which speakers favour as the retrace start and over-estimates verbs, which speakers tend to avoid retracing back to, preferring to begin the utterance again, as (Healey et al., 2011)'s experiments also demonstrate.

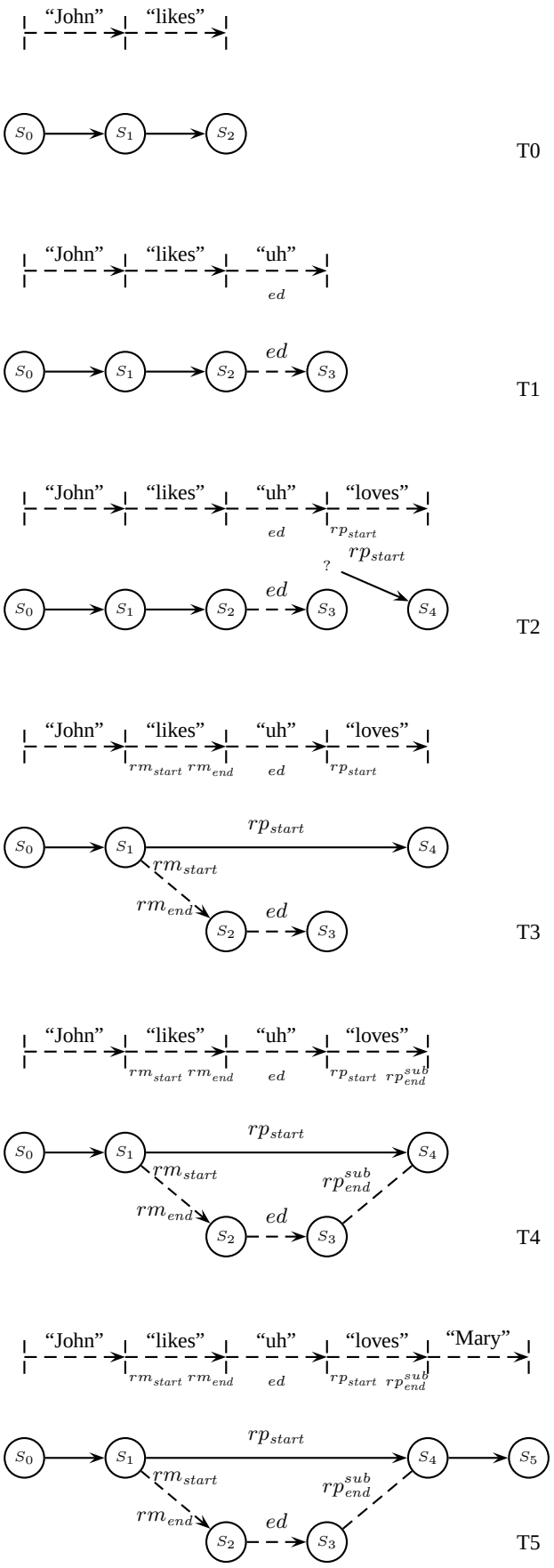


Figure 1: Strongly Incremental Repair Detection

proach to detecting repairs and isolated edit terms, assigning words the structures in (2). We include interregnum recognition in the process, due to the inclusion of interregnum vocabulary within edit term vocabulary (Ginzburg, 2012; Hough and Purver, 2013), a useful feature for repair detection (Lease et al., 2006; Qian and Liu, 2013).

$$\begin{cases} ...[rm_{start}...rm_{end} + \{ed\}rp_{start}...rp_{end}]... \\ ...\{ed\}... \end{cases} \quad (2)$$

Rather than detecting the repair structure in its left-to-right string order as above, STIR functions as in Figure 1: first detecting edit terms (possibly interregna) at step T1; then detecting repair onsets rp_{start} at T2; if one is found, backwards searching to find rm_{start} at T3; then finally finding the repair end rp_{end} at T4. Step T1 relies mainly on lexical probabilities from an edit term language model; T2 exploits features of divergence from a fluent language model; T3 uses fluency of hypothesised repairs; and T4 the similarity between distributions after reparandum and repair. However, each stage integrates these basic insights via multiple related features in a statistical classifier.

4.1 Enriched incremental language models

We derive the basic information-theoretic features required using n-gram language models, as they have a long history of information theoretic analysis (Shannon, 1948) and provide reproducible results without forcing commitment to one particular grammar formalism. Following recent work on modelling grammaticality judgements (Clark et al., 2013), we implement several modifications to standard language models to develop our basic measures of fluency and uncertainty.

For our main fluent language models we train a trigram model with Kneser-Ney smoothing (Kneser and Ney, 1995) on the words and POS tags of the standard Switchboard training data (all files with conversation numbers beginning sw2*,sw3* in the Penn Treebank III release), consisting of ≈100K utterances, ≈600K words. We follow (Johnson and Charniak, 2004) by cleaning the data of disfluencies (i.e. edit terms and reparanda), to approximate a 'fluent' language model. We call these probabilities p^{lex}_{kn}, p^{pos}_{kn} below.[3]

[3]We suppress the ^{pos} and ^{lex} superscripts below where we refer to measures from either model.

We then derive *surprisal* as our principal default lexical uncertainty measurement s (equation 3) in both models; and, following (Clark et al., 2013), the (unigram) Weighted Mean Log trigram probability (WML, eq. 4)– the trigram logprob of the sequence divided by the inverse summed logprob of the component unigrams (apart from the first two words in the sequence, which serve as the first trigram history). As here we use a local approach we restrict the WML measures to single trigrams (weighted by the inverse logprob of the final word). While use of standard n-gram probability conflates syntactic with lexical probability, WML gives us an approximation to *incremental syntactic probability* by factoring out lexical frequency.

$$s(w_{i-2} \dots w_i) = -\log_2 p_{kn}(w_i \mid w_{i-2}, w_{i-1}) \quad (3)$$

$$WML(w_0 \dots w_n) = \frac{\sum_{i=2}^{i=n} \log_2 p_{kn}(w_i \mid w_{i-2}, w_{i-1})}{-\sum_{j=2}^{n} \log_2 p_{kn}(w_j)} \quad (4)$$

Distributional measures To approximate uncertainty, we also derive the entropy $H(w \,|\, c)$ of the possible word continuations w given a context c, from $p(w_i \,|\, c)$ for all words w_i in the vocabulary – see (5). Calculating distributions over the entire lexicon incrementally is costly, so we approximate this by constraining the calculation to words which are observed at least once in context c in training, $w_c = \{w | count(c, w) \geq 1\}$, assuming a uniform distribution over the unseen suffixes by using the appropriate smoothing constant, and subtracting the latter from the former – see eq. (6).

Manual inspection showed this approximation to be very close, and the trie structure of our n-gram models allows efficient calculation. We also make use of the Zipfian distribution of n-grams in corpora by storing entropy values for the 20% most common trigram contexts observed in training, leaving entropy values of rare or unseen contexts to be computed at decoding time with little search cost due to their small or empty w_c sets.

$$H(w \mid c) = -\sum_{w \in Vocab} p_{kn}(w \mid c) \log_2 p_{kn}(w \mid c) \quad (5)$$

$$\begin{aligned} H(w \mid c) \approx & \left[-\sum_{w \in w_c} p_{kn}(w \mid c) \log_2 p_{kn}(w \mid c) \right] \\ & - [n \times \lambda \log_2 \lambda] \\ & \text{where } n = |Vocab| - |w_c| \\ & \text{and } \lambda = \frac{1 - \sum_{w \in w_c} p_{kn}(w \mid c)}{n} \end{aligned} \quad (6)$$

Given entropy estimates, we can also similarly approximate the Kullback-Leibler (KL) divergence (relative entropy) between distributions in two different contexts c_1 and c_2, i.e. $\theta(w|c_1)$ and $\theta(w|c_2)$, by pair-wise computing $p(w|c_1) \log_2(\frac{p(w|c_1)}{p(w|c_2)})$ only for words $w \in w_{c_1} \cap w_{c_2}$, then approximating unseen values by assuming uniform distributions. Using p_{kn} smoothed estimates rather than raw maximum likelihood estimations avoids infinite KL divergence values. Again, we found this approximation sufficiently close to the real values for our purposes. All such probability and distribution values are stored in incrementally constructed directed acyclic graph (DAG) structures (see Figure 1), exploiting the Markov assumption of n-gram models to allow efficient calculation by avoiding re-computation.

4.2 Individual classifiers

This section details the features used by the 4 individual classifiers. To investigate the utility of the features used in each classifier we obtain values on the standard Switchboard heldout data (PTB III files sw4[5-9]*: 6.4K utterances, 49K words).

4.2.1 Edit term detection

In the first component, we utilise the well-known observation that edit terms have a distinctive vocabulary (Ginzburg, 2012), training a bigram model on a corpus of all edit words annotated in Switchboard's training data. The classifier simply uses the surprisal s^{lex} from this edit word model, and the trigram surprisal s^{lex} from the standard fluent model of Section 4.1. At the current position w_n, one, both or none of words w_n and w_{n-1} are classified as edits. We found this simple approach effective and stable, although some delayed decisions occur in cases where s^{lex} and WML^{lex} are high in both models before the end of the edit, e.g. "I like" $\rightarrow$ "I $\{like\}$ want...". Words classified as ed are removed from the incremental processing graph (indicated by the dotted line transition in Figure 1) and the stack updated if repair hypotheses are cancelled due to a delayed edit hypothesis of w_{n-1}.

4.2.2 Repair start detection

Repair onset detection is arguably the most crucial component: the greater its accuracy, the better the input for downstream components and the lesser the overhead of filtering false positives required.

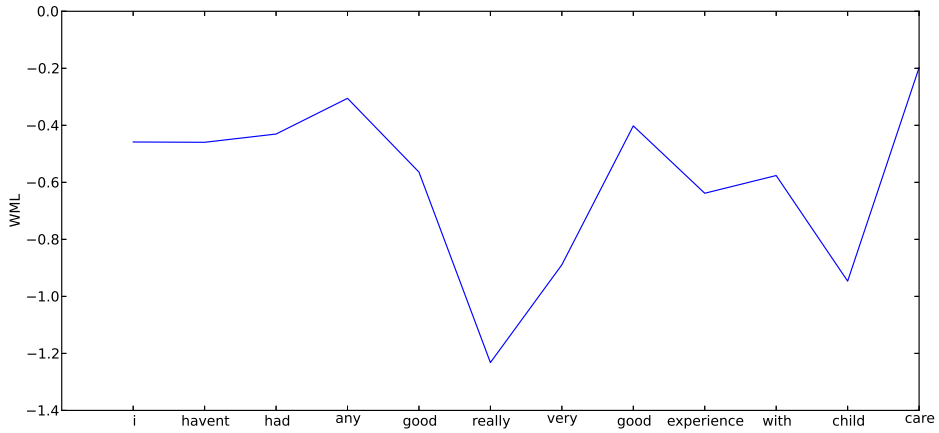


Figure 2: WML^{lex} values for trigrams for a repaired utterance exhibiting the drop at the repair onset

We use Section 4.1's information-theoretic features s, WML, H for words and POS, and introduce 5 additional information-theoretic features: ΔWML is the difference between the WML values at w_{n-1} and w_n; ΔH is the difference in entropy between w_{n-1} and w_n; $InformationGain$ is the difference between expected entropy at w_{n-1} and observed s at w_n, a measure that factors out the effect of naturally high entropy contexts; $BestEntropyReduce$ is the best reduction in entropy possible by an early rough hypothesis of reparandum onsets within 3 words; and $BestWMLBoost$ similarly speculates on the best improvement of WML possible by positing rm_{start} positions up to 3 words back. We also include simple alignment features: binary features which indicate if the word w_{i-x} is identical to the current word w_i for $x \in \{1, 2, 3\}$. With 6 alignment features, 16 N-gram features and a single logical feature $edit$ which indicates the presence of an edit word at position w_{i-1}, rp_{start} detection uses 23 features– see Table 1.

We hypothesised repair onsets rp^{start} would have significantly lower p^{lex} (lower lexical-syntactic probability) and WML^{lex} (lower syntactic probability) than other fluent trigrams. This was the case in the Switchboard heldout data for both measures, with the biggest difference obtained for WML^{lex} (non-repair-onsets: -0.736 (sd=0.359); repair onsets: -1.457 (sd=0.359)). In the POS model, entropy of continuation H^{pos} was the strongest feature (non-repair-onsets: 3.141 (sd=0.769); repair onsets: 3.444 (sd=0.899)). The trigram WML^{lex} measure for the repaired utterance "I haven't had any [good + really very good] experience with child care" can be seen in Figure 2. The steep drop at the repair onset shows the usefulness of WML features for fluency measures.

To compare n-gram measures against other local features, we ranked the features by Information Gain using 10-fold cross validation over the Switchboard heldout data– see Table 1. The language model features are far more discriminative than the alignment features, showing the potential of a general information-theoretic approach.

4.2.3 Reparandum start detection

In detecting rm_{start} positions given a hypothesised rp_{start} (stage T3 in Figure 1), we use the noisy channel intuition that removing the reparandum (from rm_{start} to rp_{start}) increases fluency of the utterance, expressed here as $WMLboost$ as described above. When using gold standard input we found this was the case on the heldout data, with a mean $WMLboost$ of 0.223 (sd=0.267) for reparandum onsets and -0.058 (sd=0.224) for other words in the 6-word history- the negative boost for non-reparandum words captures the intuition that backtracking from those points would make the utterance less grammatical, and conversely the boost afforded by the correct rm_{start} detection helps solve the continuation problem for the listener (and our detector).

Parallelism in the onsets of rp_{start} and rm_{start} can also help solve the continuation problem, and in fact the KL divergence between $\theta^{pos}(w \mid rm_{start}, rm_{start-1})$ and $\theta^{pos}(w \mid rp_{start}, rp_{start-1})$ is the second most useful feature with average merit 0.429 (+- 0.010) in cross-

validation. The highest ranked feature is ΔWML (0.437 (+- 0.003)) which here encodes the drop in the $WMLboost$ from one backtracked position to the next. In ranking the 32 features we use, again information-theoretic ones are higher ranked than the logical features.

average merit	average rank	attribute
0.139 (+- 0.002)	1 (+- 0.00)	H^{pos}
0.131 (+- 0.001)	2 (+- 0.00)	WML^{pos}
0.126 (+- 0.001)	3.4 (+- 0.66)	WML^{lex}
0.125 (+- 0.003)	4 (+- 1.10)	s^{pos}
0.122 (+- 0.001)	5.9 (+- 0.94)	$w_{i-1} = w_i$
0.122 (+- 0.001)	5.9 (+- 0.70)	BestWMLBoost^{lex}
0.122 (+- 0.002)	5.9 (+- 1.22)	InformationGain^{pos}
0.119 (+- 0.001)	7.9 (+- 0.30)	BestWMLBoost^{pos}
0.098 (+- 0.002)	9 (+- 0.00)	H^{lex}
0.08 (+- 0.001)	10.4 (+- 0.49)	ΔWML^{pos}
0.08 (+- 0.003)	10.6 (+- 0.49)	ΔH^{pos}
0.072 (+- 0.001)	12 (+- 0.00)	$POS_{i-1} = POS_i$
0.066 (+- 0.003)	13.1 (+- 0.30)	s^{lex}
0.059 (+- 0.000)	14.2 (+- 0.40)	ΔWML^{lex}
0.058 (+- 0.005)	14.7 (+- 0.64)	BestEntropyReduce^{pos}
0.049 (+- 0.001)	16.3 (+- 0.46)	InformationGain^{lex}
0.047 (+- 0.004)	16.7 (+- 0.46)	BestEntropyReduce^{lex}
0.035 (+- 0.004)	18 (+- 0.00)	ΔH^{lex}
0.024 (+- 0.000)	19 (+- 0.00)	$w_{i-2} = w_i$
0.013 (+- 0.000)	20 (+- 0.00)	$POS_{i-2} = POS_i$
0.01 (+- 0.000)	21 (+- 0.00)	$w_{i-3} = w_i$
0.009 (+- 0.000)	22 (+- 0.00)	edit
0.006 (+- 0.000)	23 (+- 0.00)	$POS_{i-3} = POS_i$

Table 1: Feature ranker (Information Gain) for rp_{start} detection- 10-fold x-validation on Switchboard heldout data.

4.2.4 Repair end detection and structure classification

For rp_{end} detection, using the notion of parallelism, we hypothesise an effect of divergence between θ^{lex} at the reparandum-final word rm_{end} and the repair-final word rp_{end}: for repetition repairs, KL divergence will trivially be 0; for substitutions, it will be higher; for deletes, even higher. Upon inspection of our feature ranking this KL measure ranked 5th out of 23 features (merit= 0.258 (+- 0.002)).

We introduce another feature encoding parallelism $ReparandumRepairDifference$: the difference in probability between an utterance cleaned of the reparandum and the utterance with its repair phase substituting its reparandum. In both the POS (merit=0.366 (+- 0.003)) and word (merit=0.352 (+- 0.002)) LMs, this was the most discriminative feature.

4.3 Classifier pipeline

STIR effects a pipeline of classifiers as in Figure 3, where the ed classifier only permits non ed words to be passed on to rp_{start} classification and for rp_{end} classification of the active repair hypotheses, maintained in a stack. The rp_{start} classifier passes positive repair hypotheses to the rm_{start} classifier, which backwards searches up to 7 words back in the utterance. If a rm_{start} is classified, the output is passed on for rp_{end} classification at the end of the pipeline, and if not rejected this is pushed onto the repair stack. Repair hypotheses are are popped off when the string is 7 words beyond its rp_{start} position. Putting limits on the stack's storage space is a way of controlling for processing overhead and complexity. Embedded repairs whose rm_{start} coincide with another's rp_{start} are easily dealt with as they are added to the stack as separate hypotheses.[4]

Classifiers Classifiers are implemented using Random Forests (Breiman, 2001) and we use different error functions for each stage using MetaCost (Domingos, 1999). The flexibility afforded by implementing adjustable error functions in a pipelined incremental processor allows control of the trade-off of immediate accuracy against runtime and stability of the sequence classification.

Processing complexity This pipeline avoids an exhaustive search all repair hypotheses. If we limit the search to within the $\langle rm_{start}, rp_{start} \rangle$ possibilities, this number of repairs grows approximately in the triangular number series– i.e. $\frac{n(n+1)}{2}$, a nested loop over previous words as n gets incremented – which in terms of a complexity class is a quadratic $O(n^2)$. If we allow more than one $\langle rm_{start}, rp_{start} \rangle$ hypothesis per word, the complexity goes up to $O(n^3)$, however in the tests that we describe below, we are able to achieve good detection results without permitting this extra search space. Under our assumption that reparandum onset detection is only triggered after repair onset detection, and repair extent detection is dependent on positive reparandum onset detection, a pipeline with accurate components will allow us to limit processing to a small subset of this search space.

[4]We constrain the problem not to include embedded deletes which may share their rp_{start} word with another repair – these are in practice very rare.

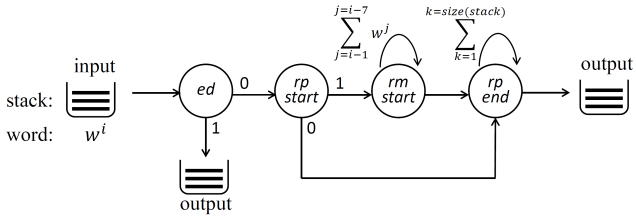


Figure 3: Classifier pipeline

5 Experimental set-up

We train STIR on the Switchboard data described above, and test it on the standard Switchboard test data (PTB III files 4[0-1]*). In order to avoid overfitting of classifiers to the basic language models, we use a cross-fold training approach: we divide the corpus into 10 folds and use language models trained on 9 folds to obtain feature values for the 10th fold, repeating for all 10. Classifiers are then trained as standard on the resulting feature-annotated corpus. This resulted in better feature utility for n-grams and better F-score results for detection in all components in the order of 5-6%.[5]

Training the classifiers Each Random Forest classifier was limited to 20 trees of maximum depth 4 nodes, putting a ceiling on decoding time. In making the classifiers cost-sensitive, MetaCost resamples the data in accordance with the cost functions: we found using 10 iterations over a resample of 25% of the training data gave the most effective trade-off between training time and accuracy.[6] We use 8 different cost functions in rp_{start} with differing costs for false negatives and positives of the form below, where R is a repair element word and F is a fluent onset:

$$\begin{array}{c} \\ R^{gold} \\ F^{gold} \end{array} \begin{array}{c} \begin{array}{cc} R^{hyp} & F^{hyp} \end{array} \\ \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix} \end{array}$$

We adopt a similar technique in rm_{start} using 5 different cost functions and in rp_{end} using 8 different settings, which when combined gives a total of 320 different cost function configurations. We hypothesise that higher recall permitted in the pipeline's first components would result in better overall accuracy as these hypotheses become refined, though at the cost of the stability of the hypotheses of the sequence and extra downstream processing in pruning false positives.

We also experiment with the number of repair hypotheses permitted per word, using limits of 1-best and 2-best hypotheses. We expect that allowing 2 hypotheses to be explored per rp_{start} should allow greater final accuracy, but with the trade-off of greater decoding and training complexity, and possible incremental instability.

As we wish to explore the incrementality versus final accuracy trade-off that STIR can achieve we now describe the evaluation metrics we employ.

5.1 Incremental evaluation metrics

Following (Baumann et al., 2011) we divide our evaluation metrics into *similarity metrics* (measures of equality with or similarity to a gold standard), *timing metrics* (measures of the timing of relevant phenomena detected from the gold standard) and *diachronic metrics* (evolution of incremental hypotheses over time).

Similarity metrics For direct comparison to previous approaches we use the standard measure of overall accuracy, the F-score over reparandum words, which we abbreviate $\mathbf{F_{rm}}$ (see 7):

$$\begin{aligned} \text{precision} &= \frac{rm^{correct}}{rm^{hyp}} \\ \text{recall} &= \frac{rm^{correct}}{rm^{gold}} \\ \text{F}_{rm} &= 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \end{aligned} \tag{7}$$

We are also interested in repair structural classification, we also measure F-score over *all* repair components (rm words, ed words as interregna and rp words), a metric we abbreviate $\mathbf{F_s}$. This is not measured in standard repair detection on Switchboard. To investigate incremental accuracy we evaluate the *delayed accuracy* (**DA**) introduced by (Zwarts et al., 2010), as described in section 2 against the utterance-final gold standard disfluency annotations, and use the mean of the 6 word F-scores.

[5] Zwarts and Johnson (2011) take a similar approach on Switchboard data to train a re-ranker of repair analyses.

[6] As (Domingos, 1999) demonstrated, there are only relatively small accuracy gains when using more than this, with training time increasing in the order of the re-sample size.

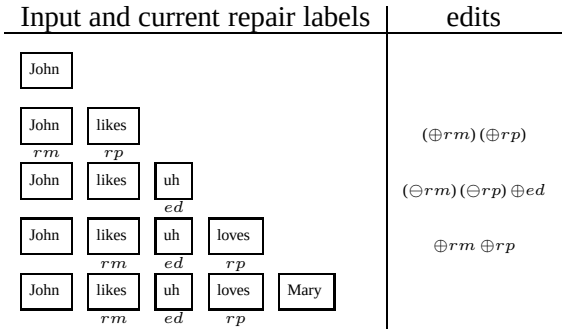
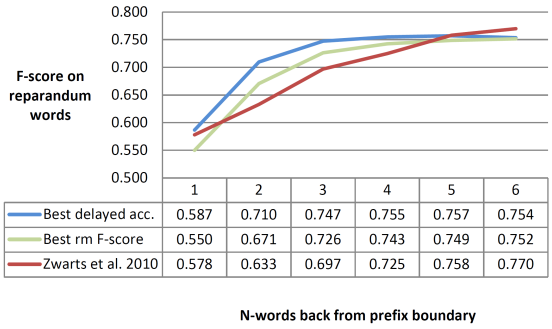


Figure 4: Edit Overhead- 4 unnecessary edits



	1	2	3	4	5	6
Best delayed acc.	0.587	0.710	0.747	0.755	0.757	0.754
Best rm F-score	0.550	0.671	0.726	0.743	0.749	0.752
Zwarts et al. 2010	0.578	0.633	0.697	0.725	0.758	0.770

Figure 6: Delayed Accuracy Curves

Timing and resource metrics Again for comparative purposes we use Zwarts et al's *time-to-detection* metrics, that is the two average distances (in numbers of words) consumed before first detection of gold standard repairs, one from rm_{start}, $\mathbf{TD_{rm}}$ and one from rp_{start}, $\mathbf{TD_{rp}}$. In our 1-best detection system, before evaluation we know a priori TD_{rp} will be 1 token, and TD_{rm} will be 1 more than the average length of $rm_{start} - rp_{start}$ repair spans correctly detected. However when we introduce a beam where multiple rm_{start}s are possible per rp_{start} with the most likely hypothesis committed as the current output, the latency may begin to increase: the initially most probable hypothesis may not be the correct one. In addition to output timing metrics, we account for intrinsic processing complexity with the metric *processing overhead* (**PO**), which is the number of classifications made by all components per word of input.

Diachronic metrics To measure stability of repair hypotheses over time we use (Baumann et al., 2011)'s *edit overhead* (**EO**) metric. EO measures the proportion of edits (add, revoke, substitute) applied to a processor's output structure that are unnecessary. STIR's output is the repair label sequence shown in Figure 1, however rather than evaluating its EO against the current gold standard labels, we use a new mark-up we term the *incremental repair gold standard*: this does not penalise lack of detection of a reparandum word rm as a bad edit until the corresponding rp_{start} of that rm has been consumed. While F_{rm}, F_s and DA evaluate against what Baumann et al. (2011) call the *current gold standard*, the incremental gold standard reflects the repair processing approach we set out in 3. An example of a repaired utterance with an EO of 44% ($\frac{4}{9}$) can be seen in Figure 4: of the 9 edits (7 repair annotations and 2 correct fluent words), 4 are unnecessary (bracketed). Note the final $\oplus rm$ is not counted as a bad edit for the reasons just given.

6 Results and Discussion

We evaluate on the Switchboard test data; Table 2 shows results of the best performing settings for each of the metrics described above, together with the setting achieving the highest total score (**TS**)– the average % achieved of the best performing system's result in each metric.[7] The settings found to achieve the highest $\text{F}_{\mathbf{rm}}$ (the metric standardly used in disfluency detection), and that found to achieve the highest TS for each stage in the pipeline are shown in Figure 5.

Our experiments showed that different system settings perform better in different metrics, and no individual setting achieved the best result in all of them. Our best utterance-final F_{rm} reaches 0.779, marginally though not significantly exceeding (Zwarts et al., 2010)'s measure and STIR achieves 0.736 on the previously unevaluated F_s. The setting with the best DA improves on (Zwarts et al., 2010)'s result significantly in terms of mean values (0.718 vs. 0.694), and also in terms of the steepness of the curves (Figure 6). The fastest average time to detection is 1 word for TD_{rp} and 2.6 words for TD_{rm} (Table 3), improving dramatically on the noisy channel model's 4.6 and 7.5 words.

Incrementality versus accuracy trade-off We aimed to investigate how well a system could do in terms of achieving both good final accuracy and incremental performance, and while the best F_{rm} setting had a large PO and relatively slow DA increase, we find STIR can find a good trade-off set-

[7] We do not include time-to-detection scores in TS as it did not vary enough between settings to be significant, however there was a difference in this measure between the 1-best stack condition and the 2-best stack condition – see below.

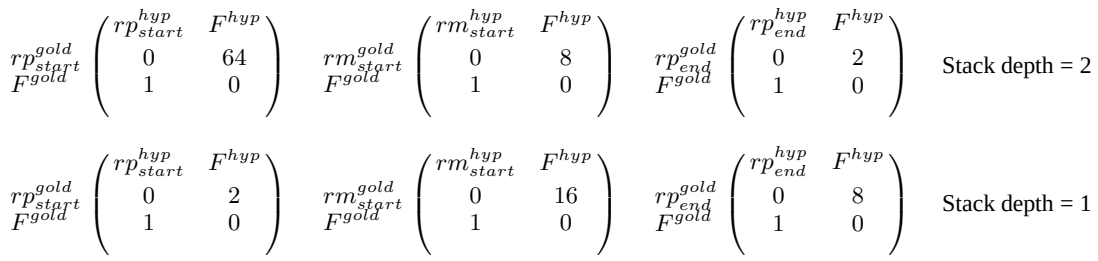


Figure 5: The cost function settings for the MetaCost classifiers for each component, for the best F_{rm} setting (top row) and best total score (TS) setting (bottom row)

	F_{rm}	F_s	DA	EO	PO
Best Final rm F-score (F_{rm})	**0.779**	0.735	0.698	3.946	1.733
Best Final repair structure F-score (F_s)	0.772	**0.736**	0.707	4.477	1.659
Best Delayed Accuracy of rm (DA)	0.767	0.721	**0.718**	1.483	1.689
Best (lowest) Edit Overhead (EO)	0.718	0.674	0.675	**0.864**	1.230
Best (lowest) Processing Overhead (PO)	0.716	0.671	0.673	0.875	**1.229**
Best Total Score (mean % of best scores) (TS)	*0.754*	*0.708*	*0.711*	*0.931*	*1.255*

Table 2: Comparison of the best performing system settings using different measures

	F_{rm}	F_s	DA	EO	PO	TD_{rp}	TD_{rm}
1-best rm_{start}	0.745	0.707	0.699	3.780	1.650	1.0	2.6
2-best rm_{start}	0.758	0.721	0.701	4.319	1.665	1.1	2.7

Table 3: Comparison of performance of systems with different stack capacities

ting: the highest TS scoring setting achieves an F_{rm} of 0.754 whilst also exhibiting a very good DA (0.711) – over 98% of the best recorded score – and low PO and EO rates – over 96% of the best recorded scores. See the bottom row of Table 2. As can be seen in Figure 5, the cost functions for these winning settings are different in nature. The best non-incremental F_{rm} measure setting requires high recall for the rest of the pipeline to work on, using the highest cost, 64, for false negative rp_{start} words and the highest stack depth of 2 (similar to a wider beam); but the best overall TS scoring system uses a less permissive setting to increase incremental performance.

We make a preliminary investigation into the effect of increasing the stack capacity by comparing stacks with 1-best rm_{start} hypotheses per rp_{start} and 2-best stacks. The average differences between the two conditions is shown in Table 3. Moving to the 2-stack condition results in gain in overall accuracy in F_{rm} and F_s, but at the cost of EO and also time-to-detection scores TD_{rm} and TD_{rp}. The extent to which the stack can be increased without increasing jitter, latency and complexity will be investigated in future work.

7 Conclusion

We have presented STIR, an incremental repair detector that can be used to experiment with incremental performance and accuracy trade-offs. In future work we plan to include probabilistic and distributional features from a top-down incremental parser e.g. Roark et al. (2009), and use STIR's distributional features to classify repair type.

Acknowledgements

We thank the three anonymous EMNLP reviewers for their helpful comments. Hough is supported by the DUEL project, financially supported by the Agence Nationale de la Research (grant number ANR-13-FRAL-0001) and the Deutsche Forschungsgemainschaft. Much of the work was carried out with support from an EPSRC DTA scholarship at Queen Mary University of London. Purver is partly supported by ConCreTe: the project ConCreTe acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET grant number 611733.

References

T. Baumann, O. Buß, and D. Schlangen. 2011. Evaluation and optimisation of incremental processors. *Dialogue & Discourse*, 2(1):113–141.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

S.E. Brennan and M.F. Schober. 2001. How listeners compensate for disfluencies in spontaneous speech. *Journal of Memory and Language*, 44(2):274–296.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293. Association for Computational Linguistics.

Alexander Clark, Gianluca Giorgolo, and Shalom Lappin. 2013. Statistical representation of grammaticality judgements: the limits of n-gram models. In *Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*, pages 28–36, Sofia, Bulgaria, August. Association for Computational Linguistics.

Pedro Domingos. 1999. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164. ACM.

Kallirroi Georgila. 2009. Using integer linear programming for detecting speech disfluencies. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 109–112. Association for Computational Linguistics.

Sebastian Germesin, Tilman Becker, and Peter Poller. 2008. Hybrid multi-step disfluency detection. In *Machine Learning for Multimodal Interaction*, pages 185–195. Springer.

Jonathan Ginzburg. 2012. *The Interactive Stance: Meaning for Conversation*. Oxford University Press.

P. G. T. Healey, Arash Eshghi, Christine Howes, and Matthew Purver. 2011. Making a contribution: Processing clarification requests in dialogue. In *Proceedings of the 21st Annual Meeting of the Society for Text and Discourse*, Poitiers, July.

Peter Heeman and James Allen. 1999. Speech repairs, intonational phrases, and discourse markers: modeling speakers' utterances in spoken dialogue. *Computational Linguistics*, 25(4):527–571.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association of Computational Linugistics (TACL)*, 2:131–142.

Julian Hough and Matthew Purver. 2013. Modelling expectation in the self-repair processing of annotat-, um, listeners. In *Proceedings of the 17th SemDial Workshop on the Semantics and Pragmatics of Dialogue (DialDam)*, pages 92–101, Amsterdam, December.

T Florian Jaeger and Harry Tily. 2011. On language utility: Processing complexity and communicative efficiency. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(3):323–335.

Mark Johnson and Eugene Charniak. 2004. A TAG-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 33–39, Barcelona. Association for Computational Linguistics.

Frank Keller. 2004. The entropy rate principle as a predictor of processing effort: An evaluation against eye-tracking data. In *EMNLP*, pages 317–324.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Matthew Lease, Mark Johnson, and Eugene Charniak. 2006. Recognizing disfluencies in conversational speech. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(5):1566–1573.

W.J.M. Levelt. 1983. Monitoring and self-repair in speech. *Cognition*, 14(1):41–104.

M. Meteer, A. Taylor, R. MacIntyre, and R. Iyer. 1995. Disfluency annotation stylebook for the switchboard corpus. ms. Technical report, Department of Computer and Information Science, University of Pennsylvania.

Tim Miller and William Schuler. 2008. A syntactic time-series model for parsing fluent and disfluent speech. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 569–576. Association for Computational Linguistics.

David Milward. 1991. *Axiomatic Grammar, Non-Constituent Coordination and Incremental Interpretation*. Ph.D. thesis, University of Cambridge.

Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In *Proceedings of NAACL-HLT*, pages 820–825.

Mohammad Sadegh Rasooli and Joel Tetreault. 2014. Non-monotonic parsing of fluent umm I mean disfluent sentences. *EACL 2014*, pages 48–53.

Hannes Rieser and David Schlangen. 2011. Introduction to the special issue on incremental processing in dialogue. *Dialogue & Discourse*, 2(1):1–10.

Brian Roark, Asaf Bachrach, Carlos Cardenas, and Christophe Pallier. 2009. Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 324–333. Association for Computational Linguistics.

Claude E. Shannon. 1948. A mathematical theory of communication. technical journal. *AT & T Bell Labs*.

Elizabeth Shriberg and Andreas Stolcke. 1998. How far do speakers back up in repairs? A quantitative model. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2183–2186.

Elizabeth Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. Ph.D. thesis, University of California, Berkeley.

Simon Zwarts and Mark Johnson. 2011. The impact of language models and loss functions on repair disfluency detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 703–711, Stroudsburg, PA, USA. Association for Computational Linguistics.

Simon Zwarts, Mark Johnson, and Robert Dale. 2010. Detecting speech repairs incrementally using a noisy channel approach. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1371–1378, Stroudsburg, PA, USA. Association for Computational Linguistics.

Semi-Supervised Chinese Word Segmentation Using Partial-Label Learning With Conditional Random Fields

Fan Yang
Nuance Communications Inc.
fan.yang@nuance.com

Paul Vozila
Nuance Communications Inc.
paul.vozila@nuance.com

Abstract

There is rich knowledge encoded in online web data. For example, punctuation and entity tags in Wikipedia data define some word boundaries in a sentence. In this paper we adopt partial-label learning with conditional random fields to make use of this valuable knowledge for semi-supervised Chinese word segmentation. The basic idea of partial-label learning is to optimize a cost function that marginalizes the probability mass in the constrained space that encodes this knowledge. By integrating some domain adaptation techniques, such as EasyAdapt, our result reaches an F-measure of 95.98% on the CTB-6 corpus, a significant improvement from both the supervised baseline and a previous proposed approach, namely *constrained decode*.

1 Introduction

A general approach for supervised Chinese word segmentation is to formulate it as a character sequence labeling problem, to label each character with its location in a word. For example, Xue (2003) proposes a four-label scheme based on some linguistic intuitions: 'B' for the beginning character of a word, 'I' for the internal characters, 'E' for the ending character, and 'S' for single-character word. Thus the word sequence "洽谈会很 成功" can be turned into a character sequence with labels as 洽\B 谈\I 会\E 很\S 成\B 功\E. A machine learning algorithm for sequence labeling, such as conditional random fields (CRF) (Lafferty et al., 2001), can be applied to the labelled training data to learn a model.

Labelled data for supervised learning of Chinese word segmentation, however, is usually expensive and tends to be of a limited amount. Researchers are thus interested in semi-supervised learning, which is to make use of unlabelled data to further improve the performance of supervised learning. There is a large amount of unlabelled data available, for example, the Gigaword corpus in the LDC catalog or the Chinese Wikipedia on the web.

Faced with the large amount of unlabelled data, an intuitive idea is to use self-training or EM, by first training a baseline model (from the supervised data) and then iteratively decoding the unlabelled data and updating the baseline model. Jiao et al. (2006) and Mann and McCallum (2007) further propose to minimize the entropy of the predicted label distribution on unlabeled data and use it as a regularization term in CRF (i.e. *entropy regularization*). Beyond these ideas, Liang (2005) and Sun and Xu (2011) experiment with deriving a large set of statistical features such as mutual information and accessor variety from unlabelled data, and add them to supervised discriminative training. Zeng et al. (2013b) experiment with graph propagation to extract information from unlabelled data to regularize the CRF training. Yang and Vozila (2013), Zhang et al. (2013), and Zeng et al. (2013a) experiment with co-training for semi-supervised Chinese word segmentation. All these approaches only leverage the distribution of the unlabelled data, yet do not make use of the knowledge that the unlabelled data might have integrated in.

There could be valuable information encoded within the unlabelled data that researchers can take advantage of. For example, punctuation creates natural word boundaries (Li and Sun, 2009): the character before a comma can only be labelled as either 'S' or 'E', while the character after a comma can only be labelled as 'S' or 'B'. Furthermore, entity tags (HTML tags or Wikipedia tags) on the web, such as emphasis and cross reference, also provide rich information for word segmentation: they might define a word or at least

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 90–98,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

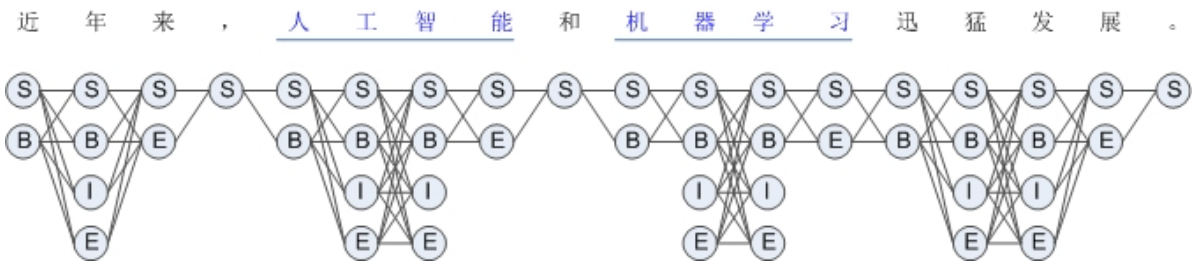


Figure 1: Sausage constraint (partial labels) from natural annotations and punctuation

give word boundary information similar to punctuation. Jiang et al. (2013) refer to such structural information on the web as natural annotations, and propose that they encode knowledge for NLP. For Chinese word segmentation, natural annotations and punctuation create a sausage[1] constraint for the possible labels, as illustrated in Figure 1. In the sentence "近年来，人工智能和机器学习迅猛发展。", the first character 近 can only be labelled with 'S' or 'B'; and the characters 来 before the comma and 展 before the Chinese period can only be labelled as 'S' or 'E'. "人工智能" and "机器学习" are two Wikipedia entities, and so they define the word boundaries before the first character and after the last character of the entities as well. The single character 和 between these two entities has only one label 'S'. This sausage constraint thus encodes rich information for word segmentation.

To make use of the knowledge encoded in the sausage constraint, Jiang et al. (2013) adopt a constrained decode approach. They first train a baseline model with labelled data, and then run constrained decode on the unlabelled data by binding the search space with the sausage; and so the decoded labels are consistent with the sausage constraint. The unlabelled data, together with the labels from constrained decode, are then selectively added to the labelled data for training the final model. This approach, using constrained decode as a middle step, provides an indirect way of leaning the knowledge. However, the middle step, constrained decode, has the risk of reinforcing the errors in the baseline model: the decoded labels added to the training data for building the final model might contain errors introduced from the baseline model. The knowledge encoded in the data carrying the information from punctuation and natural annotations is thus polluted by the errorful re-decoded labels.

A sentence where each character has exactly one label is fully-labelled; and a sentence where each character receives all possible labels is zero-labelled. A sentence with sausage-constrained labels can be viewed as partially-labelled. These partial labels carry valuable information that researchers would like to learn in a model, yet the normal CRF training typically uses fully-labelled sentences. Recently, Täckström et al. (2013) propose an approach to train a CRF model directly from partial labels. The basic idea is to marginalize the probability mass of the constrained sausage in the cost function. The normal CRF training using fully-labelled sentences is a special case where the sausage constraint is a linear line; while on the other hand a zero-labelled sentence, where the sausage constraint is the full lattice, makes no contribution in the learning since the sum of probabilities is deemed to be one. This new approach, without the need of using constrained re-decoding as a middle step, provides a direct means to learn the knowledge in the partial labels.

In this research we explore using the partial-label learning for semi-supervised Chinese word segmentation. We use the CTB-6 corpus as the labelled training, development and test data, and use the Chinese Wikipedia as the unlabelled data. We first train a baseline model with labelled data only, and then selectively add Wikipedia data with partial labels to build a second model. Because the Wikipedia data is out of domain and has distribution bias, we also experiment with two domain adaptation techniques: model interpolation and EasyAdapt (Daumé III, 2007). Our result reaches an F-measure of 95.98%, an absolute improvement of 0.72% over the very strong base-

[1] Also referred to as confusion network.

line (corresponding to 15.19% relative error reduction), and 0.33% over the constrained decode approach (corresponding to 7.59% relative error reduction). We conduct a detailed error analysis, illustrating how partial-label learning excels constrained decode in learning the knowledge encoded in the Wikipedia data. As a note, our result also out-performs (Wang et al., 2011) and (Sun and Xu, 2011).

2 Partial-Label Learning with CRF

In this section, we review in more detail the partial-label learning algorithm with CRF proposed by (Täckström et al., 2013). CRF is an exponential model that expresses the conditional probability of the labels given a sequence, as Equation 1, where y denotes the labels, x denotes the sequence, $\Phi(x, y)$ denotes the feature functions, and θ is the parameter vector. $Z(x) = \sum_y \exp(\theta^T \Phi(x, y))$ is the normalization term.

$$p_\theta(y|x) = \frac{\exp(\theta^T \Phi(x, y))}{Z(x)} \quad (1)$$

In full-label training, where each item in the sequence is labelled with exactly one tag, maximum likelihood is typically used as the optimization target. We simply sum up the log-likelihood of the n labelled sequences in the training set, as shown in Equation 2.

$$\begin{aligned} L(\theta) &= \sum_{i=1}^{n} log\ p_\theta(y|x) \\ &= \sum_{i=1}^{n} (\theta^T \Phi(x_i, y_i) - log\ Z(x_i)) \end{aligned} \quad (2)$$

The gradient is calculated as Equation 3, in which the first term $\frac{1}{n}\sum_{i=1}^{n}\Phi_j$ is the empirical expectation of feature function Φ_j, and the second term $E[\Phi_j]$ is the model expectation. Typically a forward-backward process is adopted for calculating the latter.

$$\frac{\partial}{\partial\theta_j} L(\theta) = \frac{1}{n}\sum_{i=1}^{n}\Phi_j - E[\Phi_j] \quad (3)$$

In partial-label training, each item in the sequence receives multiple labels, and so for each sequence we have a sausage constraint, denoted as $\hat{Y}(x, \tilde{y})$. The marginal probability of the sausage is defined as Equation 4.

$$p_\theta(\hat{Y}(x, \tilde{y})|x) = \sum_{y \in \hat{Y}(x,\tilde{y})} p_\theta(y|x) \quad (4)$$

The optimization target thus is to maximize the probability mass of the sausage, as shown in Equation 5.

$$L(\theta) = \sum_{i=1}^{n} log p_\theta(\hat{Y}(x_i, \tilde{y_i})|x_i) \quad (5)$$

A gradient-based approach such as L-BFGS (Liu and Nocedal, 1989) can be employed to optimize Equation 5. The gradient is calculated as Equation 6, where $E_{\hat{Y}(x,\tilde{y})}[\Phi_j]$ is the empirical expectation of feature function Φ_j constrained by the sausage, and $E[\Phi_j]$ is the same model expectation as in standard CRF. $E_{\hat{Y}(x,\tilde{y})}[\Phi_j]$ can be calculated via a forward-backward process in the constrained sausage.

$$\frac{\partial}{\partial\theta_j} L(\theta) = E_{\hat{Y}(x,\tilde{y})}[\Phi_j] - E[\Phi_j] \quad (6)$$

For fully-labelled sentences, $E_{\hat{Y}(x,\tilde{y})}[\Phi_j] = \frac{1}{n}\sum_{i=1}^{n}\Phi_j$, and so the standard CRF is actually a special case of the partial-label learning.

3 Experiment setup

In this section we describe the basic setup for our experiments of semi-supervised Chinese word segmentation.

3.1 Data

We use the CTB-6 corpus as the labelled data. We follow the official CTB-6 guideline in splitting the corpus into a training set, a development set, and a test set. The training set has 23420 sentences; the development set has 2079 sentences; and the test set has 2796 sentences. These are fully-labelled data.

For unlabelled data we use the Chinese Wikipedia. The Wikipedia data is quite noisy and asks for a lot of cleaning. We first filter out references and lists etc., and sentences with obviously bad segmentations, for example, where every character is separated by a space. We also remove sentences that contain mostly English words. We then convert all characters into full-width. We also convert traditional Chinese characters into simplified characters using the tool

mediawiki-zhconverter[2]. We then randomly select 7737 sentences and reserve them as the test set.

To create the partial labels in the Wikipedia data, we use the information from cross-reference, emphasis, and punctuation. In our pilot study we found that it's beneficial to force a cross-reference or emphasis entity as a word when the item has 2 or 3 characters. That is, if an entity in the Wikipedia has three characters it receives the labels of "BIE"; and if it has two characters it is labelled as "BE".[3]

3.2 Supervised baseline model

We create the baseline supervised model by using an order-1 linear CRF with L2 regularization, to label a character sequence with the four candidate labels "BIES". We use the tool wapiti (Lavergne et al., 2010).

Following Sun et al. (2009), Sun (2010), and Low et al. (2005), we extract two types of features: character-level features and word-level features. Given a character c_0 in the character sequence ...$c_{-2}c_{-1}c_0c_1c_2$...:

Character-level features :

- Character unigrams: c_{-2}, c_{-1}, c_0, c_1, c_2
- Character bigrams: $c_{-2}c_{-1}$, $c_{-1}c_{-0}$, c_0c_1, c_1c_2
- Consecutive character equivalence: $?c_{-2} = c_{-1}$, $?c_{-1} = c_{-0}$, $?c_0 = c_1$, $?c_1 = c_2$
- Separated character equivalence: $?c_{-3} = c_{-1}$, $?c_{-2} = c_0$, $?c_{-1} = c_1$, $?c_0 = c_2$, $?c_1 = c_3$
- Whether the current character is a punctuation: $?Punct(c_0)$
- Character sequence pattern: $T(C_{-2})T(C_{-1})T(C_0)T(C_1)T(C_2)$. We classify all characters into four types. Type one has three characters '年' (year) '月' (month) '日' (date). Type two includes number characters. Type three includes English characters. All others are Type four characters. Thus "去年三月S" would generate the character sequence pattern "41213".

Word-level features :

- The identity of the string $c[s:i]$ ($i-6 < s < i$), if it matches a word from the list of word unigrams; multiple features could be generated.
- The identity of the string $c[i:e]$ ($i < e < i+6$), if it matches a word; multiple features could be generated.
- The identity of the bi-gram $c[s:i-1]c[i:e]$ ($i-6 < s, e < i+6$), if it matches a word bigram; multiple features could be generated.
- The identity of the bi-gram $c[s:i]c[i+1:e]$ ($i-6 < s, e < i+6$), if it matches a word bigram; multiple features could be generated.
- Idiom. We use the idiom list from (Sun and Xu, 2011). If the current character c_0 and its surrounding context compose an idiom, we generate a feature for c_0 of its position in the idiom. For example, if $c_{-1}c_0c_1c_2$ is an idiom, we generate feature "Idiom-2" for c_0.

The above features together with label bigrams are fed to wapiti for training. The supervised baseline model is created with the CTB-6 corpus without the use of Wikipedia data.

3.3 Partial-label learning

The overall process of applying partial-label learning to Wikipedia data is shown in Algorithm 1. Following (Jiang et al., 2013), we first train the supervised baseline model, and use it to estimate the potential contribution for each sentence in the Wikipedia training data. We label the sentence with the baseline model, and then compare the labels with the constrained sausage. For each character, a consistent label is defined as an element in the constrained labels. For example, if the constrained labels for a character are "SB", the label 'S' or 'B' is consistent but 'I' or 'E' is not. The number of inconsistent labels for each sentence is then used as its potential contribution to the partial-label learning: higher number indicates that the partial-labels for the sentence contain more knowledge that the baseline system does not integrate, and so have higher potential contribution. The Wikipedia training sentences are then ranked by their potential contribution, and the top

[2]https://github.com/tszming/mediawiki-zhconverter

[3]Another possibility is to label it as "SS" but we find that it's very rare the case.

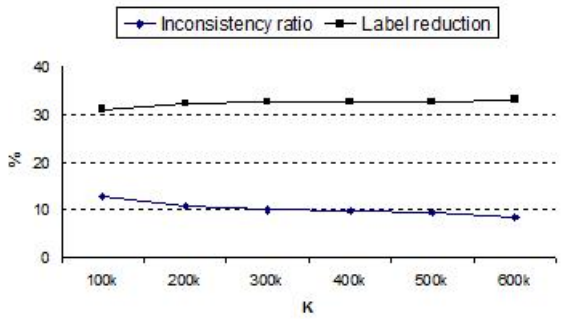


Figure 2: Encoded knowledge: inconsistency ratio and label reduction

K sentences together with their partial labels are then added to the CTB-6 training data to build a new model, using partial-label learning.[4] In our experiments, we try six data points with K = 100k, 200k, 300k, 400k, 500k, 600k. Figure 2 gives a rough idea of the knowledge encoded in Wikipedia for these data points with inconsistency ratio and label reduction. Inconsistency ratio is the percentage of characters that have inconsistent labels; and label reduction is the percentage of the labels reduced in the full lattice.

We modify wapiti to implement the partial-label learning as described in Section 2. Same as baseline, L2 regularization is adopted.

Algorithm 1 Partial-label learning

1. Train supervised baseline model M_0
2. For each sentence x in Wiki-Train:
3. $y \leftarrow$ Decode(x, M_0)
4. diff $\leftarrow$ Inconsistent(y, $\hat{Y}(x, \tilde{y})$)
5. if diff > 0:
6. C $\leftarrow$ C $\cup$ ($\hat{Y}(x, \tilde{y})$, diff)
7. Sort(C, diff, reverse)
8. Train model M^{pl} with CTB-6 and top K sentences in C using partial-label learning

3.4 Constrained decode

Jiang et al. (2013) implement the constrained decode algorithm with perceptron. However, CRF is generally believed to out-perform perceptron, yet the comparison of CRF vs perceptron is out of the scope of this paper. Thus for fair comparison, we re-implement the constrained decode algorithm with CRF.

Algorithm 2 shows the constrained decode implementation. We first train the baseline model with the CTB-6 data. We then use this baseline model to run normal decode and constrained decode for each sentence in the Wikipedia training set. If the normal decode and constrained decode have different labels, we add the constrained decode together with the number of different labels to the filtered Wikipedia training corpus. The filtered Wikipedia training corpus is then sorted using the number of different labels, and the top K sentences with constrained decoded labels are then added to the CTB-6 training data for building a new model using normal CRF.

Algorithm 2 Constrained decode

1. Train supervised baseline model M_0
2. For each sentence x in Wiki-Train:
3. $y \leftarrow$ Decode(x, M_0)
4. $\bar{y} \leftarrow$ ConstrainedDecode(x, M_0)
5. diff $\leftarrow$ Difference(y, $\bar{y}$)
6. if diff > 0:
7. C $\leftarrow$ C $\cup$ ($\bar{y}$, diff)
8. Sort(C, diff, reverse)
9. Train model M^{cd} with CTB-6 and top K sentences in C using normal CRF

4 Evaluation on Wikipedia test set

In order to determine how well the models learn the encoded knowledge (i.e. partial labels) from the Wikipedia data, we first evaluate the models against the Wikipedia test set. The Wikipedia test set, however, is only partially-labelled. Thus the metric we use here is *consistent label accuracy*, similar to how we rank the sentences in Section 3.3, defined as whether a predicted label for a character is an element in the constrained labels. Because partial labels are only sparsely distributed in the test data, a lot of characters receive all four labels in the constrained sausage. Evaluating against characters with all four labels do not really represent the models' difference as it is deemed to be consistent. Thus beyond evaluating against all characters in the Wikipedia test set (referred to as *Full* measurement), we also evaluate against characters that are only constrained with less than four labels (referred to as *Label* measurement). The *Label* measurement focuses on en-

[4] Knowledge is sparsely distributed in the Wikipedia data. Using the Wikipedia data without the CTB-6 data for partial-label learning does not necessarily guarantee convergence. Also the CTB-6 training data helps to learn that certain label transitions, such as "B B" or "E E", are not legal.

coded knowledge in the test set and so can better represent the model's capability of learning from the partial labels.

Results are shown in Figure 3 with the *Full* measurement and in Figure 4 with the *Label* measurement. The x axes are the size of Wikipedia training data, as explained in Section 3.3. As can be seen, both constrained decode and partial-label learning perform much better than the baseline supervised model that is trained from CTB-6 data only, indicating that both of them are learning the encoded knowledge from the Wikipedia training data. Also we see the trend that the performance improves with more data in training, also suggesting the learning of encoded knowledge. Most importantly, we see that partial-label learning consistently out-performs constrained decode in all data points. With the *Label* measurement, partial-label learning gives 1.7% or higher absolute improvement over constrained decode across all data points. At the data point of 600k, constrained decode gives an accuracy of 97.14%, while partial-label learning gives 98.93% (baseline model gives 87.08%). The relative gain (from learning the knowledge) of partial-label learning over constrained decode is thus 18% $((98.93 - 97.14)/(97.14 - 87.08))$. These results suggest that partial-label learning is more effective in learning the encoded knowledge in the Wikipedia data than constrained decode.

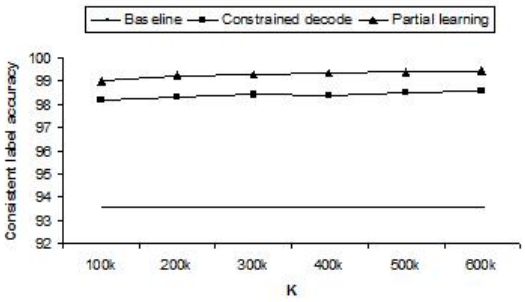


Figure 3: Wiki label evaluation results: Full

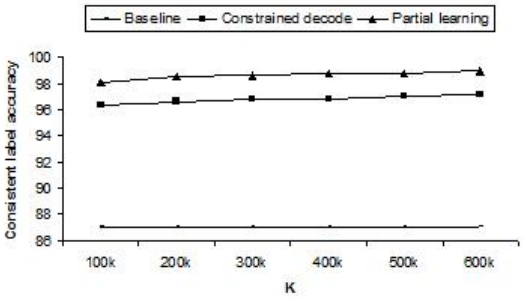


Figure 4: Wiki label evaluation results: Label

5 CTB evaluation

5.1 Model adaptation

Our ultimate goal, however, is to determine whether we can leverage the encoded knowledge in the Wikipedia data to improve the word segmentation in CTB-6. We run our models against the CTB-6 test set, with results shown in Figure 5. Because we have fully-labelled sentences in the CTB-6 data, we adopt the F-measure as our evaluation metric here. The baseline model achieves 95.26% in F-measure, providing a state-of-the-art supervised performance. Constrained decode is able to improve on this already very strong baseline performance, and we see the nice trend of higher performance with more unlabeled data for training, indicating that constrained decode is making use of the encoded knowledge in the Wikipedia data to help CTB-6 segmentation.

When we look at the partial-label model, however, the results tell a totally different story.

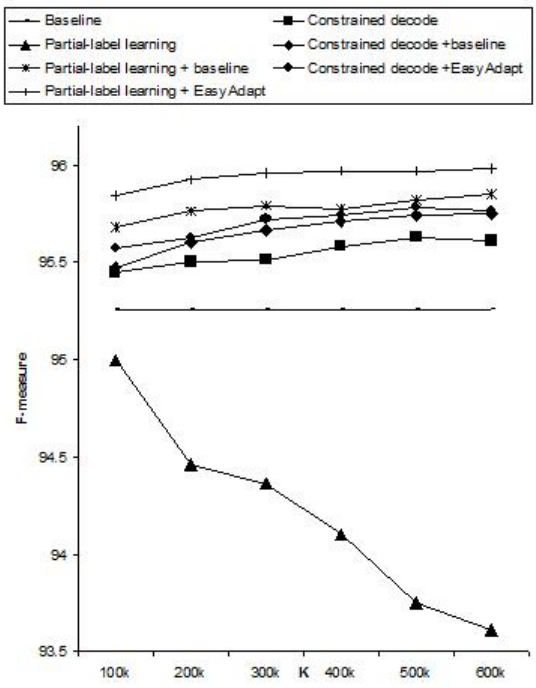


Figure 5: CTB evaluation results

First, it actually performs worse than the baseline model, and the more data added to training, the worse the performance is. In the previous section we show that partial-label learning is more effective in learning the encoded knowledge in Wikipedia data than constrained decode. So, what goes wrong? We hypothesize that there is an out-of-domain distribution bias in the partial labels, and so the more data we add, the worse the in-domain performance is. Constrained decode actually helps to smooth out the out-of-domain distribution bias by using the re-decoded labels with the in-domain supervised baseline model. For example, both the baseline model and constrained decode correctly give the segmentation "提供/了/运输/和/**给排水**/之/便", while partial-label learning gives incorrect segmentation "提供/了/运输/和/**给/排水**/之/便". Looking at the Wikipedia training data, 排水 is tagged as an entity 13 times; and 给排水, although occurs 13 times in the data, is never tagged as an entity. Partial-label learning, which focuses on the tagged entities, thus overrules the segmentation of 给排水. Constrained decode, on the other hand, by using the correctly re-decoded labels from the baseline model, observes enough evidence to correctly segment 给排水 as a word.

To smooth out the out-of-domain distribution bias, we experiment with two approaches: model interpolation and EasyAdapt (Daumé III, 2007).

5.1.1 Model interpolation

We linearly interpolate the model of partial-label learning M^{pl} with the baseline model M_0 to create the final model M_+^{pl}, as shown in Equation 7. The interpolation weight is optimized via a grid search between 0.0 and 1.0 with a step of 0.1, tuned on the CTB-6 development set. Again we modify wapiti so that it takes two models and an interpolation weight as input. For each model it creates a search lattice with posteriors, and then linearly combines the two lattices using the interpolation weight to create the final search space for decoding. As shown in Figure 5, model M_+^{pl} consistently out-performs constrained decode in all data points. We also see the trend of better performance with more training data.

$$M_+^{pl} = \lambda * M_0 + (1 - \lambda) * M^{pl} \qquad (7)$$

5.1.2 EasyAdapt

EasyAdapt is a straightforward technique but has been shown effective in many domain adaptation tasks (Daumé III, 2007). We train the model M_{ea}^{pl} with feature augmentation. For each out-of-domain training instance $< x_o, y_o >$, where x_o is the input features and y_o is the (partial) labels, we copy the features and file them as an additional feature set, and so the training instance becomes $< x_o, x_o, y_o >$. The in-domain training data remains the same. Consistent with (Daumé III, 2007), EasyAdapt gives us the best performance, as show in Figure 5. Furthermore, unlike in (Jiang et al., 2013) where they find a plateau, our results show no harm adding more training data for partial-label learning when integrated with domain adaptation, although the performance seems to saturate after 400k sentences.

Finally, we search for the parameter setting of best performance on the CTB-6 development set, which is to use EasyAdapt with $K = 600k$ sentences of Wikipedia data. With this setting, the performance on the CTB-6 test set is 95.98% in F-measure. This is 0.72% absolute improvement over supervised baseline (corresponding to 15.19% relative error reduction), and 0.33% absolute improvement over constrained decode (corresponding to 7.59% relative error reduction); the differences are both statistically significant ($p < 0.001$).[5] As a note, this result out-performs (Sun and Xu, 2011) (95.44%) and (Wang et al., 2011) (95.79%), and the differences are also statistically significant ($p < 0.001$).

5.2 Analysis with examples

To better understand why partial-label learning is more effective in learning the encoded knowledge, we look at cases where M_0 and M^{cd} have the incorrect segmentation while M^{pl} (and its domain adaptation variance M_+^{pl} and M_{ea}^{pl}) have the correct segmentation. We find that the majority is due to the error in re-decoded labels outside of encoded knowledge. For example, M_0 and M^{cd} give the segmentation "地震/为/**里/氏**/6.9/ 级", yet the correct segmentation given by partial-label learning is "地震/为/**里 氏**/6.9/ 级". Looking at the Wikipedia training data, there are 38 tagged entities of 里氏, but there are another 190 mentions of

[5] Statistical significance is evaluated with z-test using the standard deviation of $\sqrt{F * (1 - F)/N}$, where F is the F-measure and N is the number of words.

里氏 that are not tagged as an entity. Thus for constrained decode it sees 38 cases of "里\B 氏\E" and 190 cases of "里\S 氏\S" in the Wikipedia training data. The former comes from the encoded knowledge while the latter comes from re-decoded labels by the baseline model. The much bigger number of incorrect labels from the baseline re-decoding badly pollute the encoded knowledge. This example illustrates that constrained decode reinforces the errors from the baseline. On the other hand, the training materials for partial-label learning are purely the encoded knowledge, which is not impacted by the baseline model error. In this example, partial-label learning focuses only on the 38 cases of "里\B 氏\E" and so is able to learn that 里氏 is a word.

As a final remark, we want to make a point that, although the re-decoded labels serve to smooth out the distribution bias, the Wikipedia data is indeed not the ideal data set for such a purpose, because it itself is out of domain. The performance tends to degrade when we apply the baseline model to re-decode the out-of-domain Wikipedia data. The errorful re-decoded labels, when being used to train the model M^{cd}, could lead to further errors. For example, the baseline model M_0 is able to give the correct segmentation "电脑/**元器件**" in the CTB-6 test set. However, when it is applied to the Wikipedia data for constrained decode, for the seven occurrences of 元器件, three of which are correctly labelled as "元\B 器\I 件\E", but the other four have incorrect labels. The final model M^{cd} trained from these labels then gives incorrect segmentation "两/市/生产/的/电脑/**元/器件**/大量/销往/世界/各地" in the CTB-6 test set. On the other hand, model interpolation or EasyAdapt with partial-label learning, focusing only on the encoded knowledge and not being impacted by the errorful re-decoded labels, performs correctly in this case. For a more fair comparison between partial-label learning and constrained decode, we have also plotted the results of model interpolation and EasyAdapt for constrained decode in Figure 5. As can be seen, they improve on constrained decode a bit but still fall behind the correspondent domain adaptation approach of partial-label learning.

6 Conclusion and future work

There is rich information encoded in online web data. For example, punctuation and entity tags define some word boundaries. In this paper we show the effectiveness of partial-label learning in digesting the encoded knowledge from Wikipedia data for the task of Chinese word segmentation. Unlike approaches such as constrained decode that use the errorful re-decoded labels, partial-label learning provides a direct means to learn the encoded knowledge. By integrating some domain adaptation techniques such as EasyAdapt, we achieve an F-measure of 95.98% in the CTB-6 corpus, a significant improvement from both the supervised baseline and constrained decode. Our results also beat (Wang et al., 2011) and (Sun and Xu, 2011).

In this research we employ a sausage constraint to encode the knowledge for Chinese word segmentation. However, a sausage constraint does not reflect the legal label sequence. For example, in Figure 1 the links between label 'B' and label 'S', between 'S' and 'E', and between 'E' and 'I' are illegal, and can confuse the machine learning. In our current work we solve this issue by adding some fully-labelled data into training. Instead we can easily extend our work to use a lattice constraint by removing the illegal transitions from the sausage. The partial-label learning stands the same, by executing the forward-backward process in the constrained lattice. In future work we will examine partial-label learning with this more enforced lattice constraint in depth.

Acknowledgments

The authors would like to thank Wenbin Jiang, Xiaodong Zeng, and Weiwei Sun for helpful discussions, and the anonymous reviewers for insightful comments.

References

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Annual meetingassociation for computational linguistics*, pages 256–263. Association for Computational Linguistics.

Wenbin Jiang, Meng Sun, Yajuan Lv, Yating Yang, and Qun Liu. 2013. Discriminative learning with natural annotations: Word segmentation as a case study. In *Proceedings of The 51st Annual Meeting of the Association for Computational Linguistics*, pages 761–769.

Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Com-*

putational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, ACL-44, pages 209–216.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 504–513. Association for Computational Linguistics, July.

Zhongguo Li and Maosong Sun. 2009. Punctuation as implicit annotations for Chinese word segmentation. *Computational Linguistics*, 35:505–512.

Percy Liang. 2005. Semi-supervised learning for natural language. Master's thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, May.

D. C. Liu and J. Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(3):503–528, December.

Jin Kiat Low, Hwee Tou Ng, and Wenyuan Guo. 2005. A maximum entropy approach to chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 161–164, San Francisco, CA, USA.

Gideon S. Mann and Andrew McCallum. 2007. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, NAACL-Short '07, pages 109–112.

Weiwei Sun and Jia Xu. 2011. Enhancing chinese word segmentation using unlabeled data. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 970–979, Edinburgh, Scotland, UK., July.

Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2009. A discriminative latent variable Chinese segmenter with hybrid word/character information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 56–64.

Weiwei Sun. 2010. Word-based and character-based word segmentation models: comparison and combination. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1211–1219.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving chinese word segmentation and POS tagging with semi-supervised methods using large auto-analyzed data. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 309–317.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, pages 29–48.

Fan Yang and Paul Vozila. 2013. An empirical study of semi-supervised Chinese word segmentation using co-training. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1191–1200, Seattle, Washington, USA, October. Association for Computational Linguistics.

Xiaodong Zeng, Derek F. Wong, Lidia S. Chao, and Isabel Trancoso. 2013a. Co-regularizing character-based and word-based models for semi-supervised chinese word segmentation. In *Proceedings of The 51st Annual Meeting of the Association for Computational Linguistics*, pages 171–176.

Xiaodong Zeng, Derek F. Wong, Lidia S. Chao, and Isabel Trancoso. 2013b. Graph-based semi-supervised model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of The 51st Annual Meeting of the Association for Computational Linguistics*, pages 770–779.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013. Exploring representations from unlabeled data with co-training for Chinese word segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 311–321, Seattle, Washington, USA, October. Association for Computational Linguistics.

Accurate Word Segmentation and POS Tagging for Japanese Microblogs: Corpus Annotation and Joint Modeling with Lexical Normalization

Nobuhiro Kaji[*†] and **Masaru Kitsuregawa**[†‡]
[*]National Institute of Information and Communications Technology
[†]Institute of Industrial Science, The University of Tokyo
[‡]National Institute of Informatics
{kaji, kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract

Microblogs have recently received widespread interest from NLP researchers. However, current tools for Japanese word segmentation and POS tagging still perform poorly on microblog texts. We developed an annotated corpus and proposed a joint model for overcoming this situation. Our annotated corpus of microblog texts enables not only training of accurate statistical models but also quantitative evaluation of their performance. Our joint model with lexical normalization handles the orthographic diversity of microblog texts. We conducted an experiment to demonstrate that the corpus and model substantially contribute to boosting accuracy.

1 Introduction

Microblogs, such as Twitter[1] and Weibo[2], have recently become an important target of NLP technology. Since microblogs offer an instant way of posting textual messages, they have been given increasing attention as valuable sources for such actions as mining opinions (Jiang et al., 2011) and detecting sudden events such as earthquake (Sakaki et al., 2010).

However, many studies have reported that current NLP tools do not perform well on microblog texts (Foster et al., 2011; Gimpel et al., 2011). In the case of Japanese text processing, the most serious problem is poor accuracy of word segmentation and POS tagging. Since these two tasks are positioned as the fundamental step in the text processing pipeline, their accuracy is vital for all downstream applications.

[1]https://twitter.com
[2]https://www.weibo.com

1.1 Development of annotated corpus

The main obstacle that makes word segmentation and POS tagging in the microblog domain challenging is the lack of annotated corpora. Because current annotated corpora are from other domains, such as news articles, it is difficult to train models that perform well on microblog texts. Moreover, system performance cannot be evaluated quantitatively.

We remedied this situation by developing an annotated corpus of Japanese microblogs. We collected 1831 sentences from Twitter and manually annotated these sentences with word boundaries, POS tags, and normalized forms of words (*c.f.*, Section 1.2).

We, for the first time, present a comprehensive empirical study of Japanese word segmentation and POS tagging on microblog texts by using this corpus. Specifically, we investigated how well current models trained on existing corpora perform in the microblog domain. We also explored performance gains achieved by using our corpus for training, and by jointly performing lexical normalization (*c.f.*, Section 1.2).

1.2 Joint modeling with lexical normalization

Orthographic diversity in microblog texts causes a problem when training a statistical model for word segmentation and POS tagging. Microblog texts frequently contain informal words that are spelled in a non-standard manner, *e.g.*, "*oredi* (*already*)", "*b4* (*before*)", and "*talkin* (*talking*)" (Han and Baldwin, 2011). Such words, hereafter referred to as *ill-spelled words*, are so productive that they considerably increase the vocabulary size. This makes training of statistical models difficult.

We address this problem by jointly conducting lexical normalization. Although a wide variety of ill-spelled words are used in microblog texts, many can be normalized into *well-spelled equivalents*, which conform to standard rules of spelling.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 99–109,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

A joint model with lexical normalization is able to handle orthographic diversity by exploiting information obtainable from the well-spelled equivalents.

The proposed joint model was empirically evaluated on the microblog corpus we developed. Our experiment demonstrated that the proposed model can perform word segmentation and POS tagging substantially better than current state-of-the-art models.

1.3 Summary

Contributions of this paper are the following:

- We developed a microblog corpus that enables not only training of accurate models but also quantitative evaluation for word segmentation and POS tagging in the microblog domain.[3]
- We propose a joint model with lexical normalization for better handling of orthographic diversity in microblog texts. In particular, we present a new method of training the joint model using a partially annotated corpus (*c.f.*, Section 7.4).
- We, for the first time, present a comprehensive empirical study of word segmentation and POS tagging for microblogs. The experimental results demonstrated that both the microblog corpus and joint model greatly contributes to training accurate models for word segmentation and POS tagging.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 discusses the task of lexical normalization and introduces terminology. Section 4 presents our microblog corpus and results of our corpus analysis. Section 5 presents an overview of our joint model with lexical normalization, and Sections 6 and 7 provide details of the model. Section 8 presents experimental results and discussions, and Section 9 presents concluding remarks.

2 Related Work

Researchers have recently developed various microblog corpora annotated with rich linguistic information. Gimpel et al. (2011) and Foster et al. (2011) annotated English microblog posts with POS tags. Han and Baldwin (2011) released a microblog corpus annotated with normalized forms of words. A Chinese microblog corpus annotated with word boundaries was developed for SIGHAN bakeoff (Duan et al., 2012). However, there are no microblog corpora annotated with word boundaries, POS tags, and normalized sentences.

There has been a surge of interest in lexical normalization with the advent of microblogs (Han and Baldwin, 2011; Liu et al., 2012; Han et al., 2012; Wang and Ng, 2013; Zhang et al., 2013; Ling et al., 2013; Yang and Eisenstein, 2013; Wang et al., 2013). However, these studies did not address enhancing word segmentation.

Wang et al. (2013) proposed a method of joint ill-spelled word recognition and word segmentation. With their method, informal spellings are merely recognized and not normalized. Therefore, they did not investigate how to exploit the information obtainable from well-spelled equivalents to increase word segmentation accuracy.

Some studies also explored integrating the lexical normalization process into word segmentation and POS tagging (Ikeda et al., 2009; Sasano et al., 2013). A strength of our joint model is that it uses rich character-level and word-level features used in state-of-the-art models of joint word segmentation and POS tagging (Kudo et al., 2004; Neubig et al., 2011; Kaji and Kitsuregawa, 2013). Thanks to these features, our model performed much better than Sasano et al.'s system, which is the only publicly available system that jointly conducts lexical normalization, in the experiments (see Section 8). Another advantage is that our model can be trained on a partially annotated corpus. Furthermore, we present a comprehensive evaluation in terms of precision and recall on our microblog corpus. Such an evaluation has not been conducted in previous work due to the lack of annotated corpora.[4]

3 Lexical Normalization Task

This section explains the task of lexical normalization addressed in this paper. Since lexical normalization is a relatively new research topic, there are no precise definitions of a lexical normalization task that are widely accepted by researchers.

[3] Please contact the first author for this corpus.

[4] Very recently, Saito et al. (2014) conducted similar empirical evaluation on microblog corpus. However, they used biased dataset, in which every sentence includes at least one ill-spelled words.

Table 1: Examples of our target ill-spelled words and their well-spelled equivalents. Phonemes are shown between slashes. English translations are provided in parentheses.

Ill-spelled word	Well-spelled equivalent
すげえ /sugee/	すごい /sugoi/ (great)
戻ろ /modoro/	戻ろう /modorou/ (going to return)
うまいいいい /umaiiii/	うまい /umai/ (yummy)

Therefore, it is important to clarify our task setting before discussing our joint model.

3.1 Target ill-spelled words

Many studies on lexical normalization have pointed out that phonological factors are deeply involved in the process of deriving ill-spelled words. Xia et al. (2006) investigated a Chinese chat corpus and reported that 99.2% of the ill-spelled words were derived by phonetic mapping from well-spelled equivalents. Wang and Ng (2013) analyzed 200 Chinese messages from Weibo and 200 English SMS messages from the NUS SMS corpus (How and Kan, 2005). Their analysis revealed that most ill-spelled words were derived from well-spelled equivalents based on pronunciation similarity.

On top of these investigations, we focused on ill-spelled words that are derived by phonological mapping from well-spelled words by assuming that such ill-spelled words are dominant in Japanese microblogs as well. We also assume that these ill-spelled words can be normalized into well-spelled equivalents on a word-to-word basis, as assumed in a previous study (Han and Baldwin, 2011). The validity of these two assumptions is empirically assessed in Section 4.

Table 1 lists examples of our target ill-spelled words, their well-spelled equivalents, and their phonemes. The ill-spelled word in the first row is formed by changing the continuous two vowels from /oi/ to /ee/. This type of change in pronunciation is often observed in Japanese spoken language. The second row presents contractions. The last vowel character "う" /u/ of the well-spelled word is dropped. The third row illustrates word lengthening. The ill-spelled word is derived by repeating the vowel character "い" /i/.

3.2 Terminology

We now introduce the terminology that will be used throughout the remainder of this paper. The term *word surface form* (or *surface form* for short) is used to refer to the word form observed in an actual text, while *word normal form* (or *normal form*) refers to the normalized word form. Note that surface forms of well-spelled words are always identical to their normal forms.

It is possible that the word surface form and normal form have distinct POS tags, although they are identical in most cases. Take the ill-spelled word "戻ろ" /modoro/ as an example (the second row of Table 1). According to the JUMAN POS tag set,[5] POS of its surface form is CONTRACTED VERB, while that of its normal form is VERB.[6] To handle such a case, we strictly distinguish between these two POS tags by referring to them as *surface POS tags* and *normal POS tags*, respectively.

Given these terms, the tasks addressed in this paper can be stated as follows. Word segmentation is a task of segmenting a sentence into a sequence of word surface forms, and POS tagging is a task of providing surface POS tags. The task of joint lexical normalization, word segmentation, and POS tagging is to map a sentence into a sequence of quadruplets: word surface form, surface POS tag, normal form, and normal POS tag.

4 Microblog Corpus

This section introduces our microblog corpus. We first explain the process of developing the corpus then present the results of our agreement study and corpus analysis.

4.1 Data collection and annotation

The corpus was developed by manually annotating text messages posted to Twitter.

The posts to be annotated were collected as follows. 171,386 Japanese posts were collected using the Twitter API[7] on December 6, 2013. Among these, 1000 posts were randomly selected then manually split into sentences. As a result, we obtained 1831 sentences as a source of the corpus.

Two human participants annotated the 1831 sentences with surface forms and surface POS tags. Since much effort has already been done to annotate corpora with this information, the annotation process here follows the guidelines used to

[5] http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN

[6] In this paper, we use simplified POS tags for explanation purposes. Remind that these tags are different from the original ones defined in JUMAN POS tag set.

[7] https://stream.twitter.com/1.1/statuses/sample.json

develop such corpora in previous studies (Kurohashi and Nagao, 1998; Hashimoto et al., 2011).

The two participants also annotated ill-spelled words with their normal forms and normal POS tags. Although this paper targets only informal phonological variations (*c.f.*, Section 3), other types of ill-spelled words were also annotated to investigate their frequency distribution in microblog texts. Specifically, besides informal phonological variations, spelling errors and Twitter-specific abbreviations were annotated. As a result, 833 ill-spelled words were identified (Table 2). They were all annotated with normal forms and normal POS tags.

4.2 Agreement study

We investigated the inter-annotator agreement to check the reliability of the annotation. During the annotation process, the two participants collaboratively annotated around 90% of the sentences (specifically, 1647 sentences) with normal forms and normal POS tags, and elaborated an annotation guideline through discussion. They then independently annotated the remaining 184 sentences (1431 words), which were used for the agreement study. Our annotation guideline is shown in the supplementary material.

We first explored the extent to which the two participants agreed in distinguishing between well-spelled words and ill-spelled words. For this task, we observed Cohen's kappa of 0.96 (almost perfect agreement). This results show that it is easy for humans to distinguish between these two types of words.

Next, we investigated whether the two participants could give ill-spelled words with the same normal forms and normal POS tags. For this purpose, we regarded the normal forms and normal POS tags annotated by one participant as goldstandards and calculated precision and recall achieved by the other participant. We observed moderate agreement between the two participants: 70% (56/80) precision and 73% (56/76) recall. We manually analyzed the conflicted examples and found that there were more than one acceptable normal form in many of these cases. Therefore, we would like to note that the precision and recall reported above are rather pessimistic estimations.

4.3 Analysis

We conducted corpus analysis to confirm the feasibility of our approach.

Table 2: Frequency distribution over three types of ill-spelled words in corpus.

Type	Frequency
Informal phonological variation	804 (92.9%)
Spelling error	27 (3.1%)
Twitter-specific abbreviation	34 (3.9%)
Total	865 (100%)

Table 2 illustrates that phonological variations constitute a vast majority of ill-spelled words in Japanese microblog texts. In addition, analysis of the 804 phonological variations showed that 793 of them can be normalized into single words. These represent the validity of the two assumptions we made in Section 3.1.

We then investigated whether lexical normalization can decrease the number of out-of-vocabulary words. For the 793 ill-spelled words, we counted how many of their surface forms and normal forms were not registered in the JUMAN dictionary.[8] The result suggests that 411 (51.8%) and 74 (9.3%) are not registered in the dictionary. This indicates the effectiveness of lexical normalization for decreasing out-of-vocabulary words.

5 Overview of Joint Model

This section gives an overview of our joint model with lexical normalization for accurate word segmentation and POS tagging.

5.1 Lattice-based approach

A lattice-based approach has been commonly adopted to perform joint word segmentation and POS tagging (Jiang et al., 2008; Kudo et al., 2004; Kaji and Kitsuregawa, 2013). In this approach, an input sentence is transformed into a word lattice in which the edges are labeled with surface POS tags (Figure 1). Given such a lattice, word segmentation and POS tagging can be performed at the same time by traversing the lattice. A discriminative model is typically used for the traversal.

An advantage of this approach is that, while the lattice can represent an exponentially large number of candidate analyses, it can be quickly traversed using dynamic programming (Kudo et al., 2004; Kaji and Kitsuregawa, 2013) or beam search (Jiang et al., 2008). In addition, a discriminative model allows the use of rich word-level features to find the correct analysis.

[8] http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN

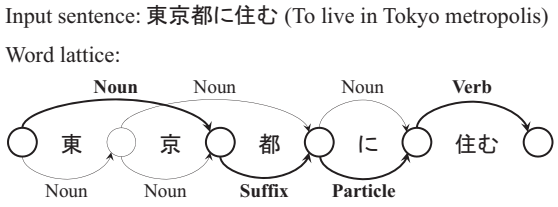


Figure 1: Example lattice (Kudo et al., 2004; Kaji and Kitsuregawa, 2013). Circle and arrow represent node and edge, respectively. Bold edges represent correct analysis.

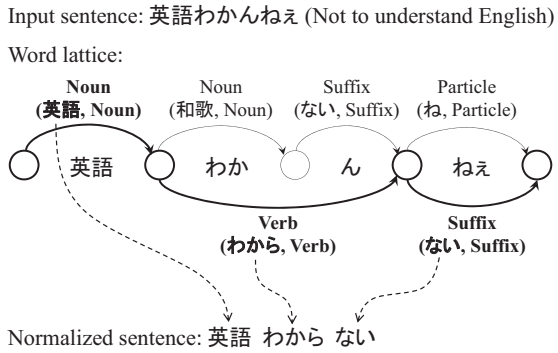


Figure 2: Lattice used to perform joint task. Normal forms and normal POS tags are shown in parentheses. As indicated by dotted arrows, normalized sentence can be obtained by concatenating normal forms associated with edges in correct analysis.

We propose extending the lattice-based approach to jointly perform lexical normalization, word segmentation, and POS tagging. We transform an input sentence into a word lattice in which the edges are labeled with not only surface POS tags but normal forms and normal POS tags (Figure 2). By traversing such a lattice, the three tasks can be performed at the same time. This approach can not only exploit rich information obtainable from word normal forms, but also achieve efficiency similar to the original lattice-based approach.

5.2 Issues

Issues on how to develop this lattice-based approach is detailed in Sections 6 and 7.

Section 6 describes how to generate a word lattice from an input sentence. This is done using a hybrid approach that combines a statistical model and *normalization dictionary*. The normalization dictionary is specifically a list of quadruplets: word surface form, surface POS tag, normal form, and normal POS tag (Table 3).

Table 3: Normalization dictionary. Columns represent entry ID, surface form, surface POS, normal form, and normal POS, respectively.

ID	Surf.	Surf. POS	Norm.	Norm. POS
A	すごい	ADJECTIVE	すごい	ADJECTIVE
B	すげえ	ADJECTIVE	すごい	ADJECTIVE
C	戻ろう	VERB	戻ろう	VERB
D	戻ろ	CONTR. VERB	戻ろう	VERB
E	うまい	ADJECTIVE	うまい	ADJECTIVE
F	うまいいいい	ADJECTIVE	うまい	ADJECTIVE

Table 4: Tag dictionary.

ID	Surf. form	Surf. POS
a	すごい (great)	ADJECTIVE
b	戻ろう (going to return)	VERB
c	戻ろ (gonna return)	CONTR. VERB
d	うまい (yummy)	ADJECTIVE

Section 7 describes a discriminative model for the lattice traversal. Our feature design as well as two training methods are presented.

6 Word Lattice Generation

In this section, we first describe a method of constructing a normalization dictionary then present a method of generating a word lattice from an input sentence.

6.1 Construction of normalization dictionary

Although large-scale normalization dictionaries are difficult to obtain, *tag dictionaries*, which list pairs of word surface forms and their surface POS tags (Table 4), are widely available in many languages including Japanese. Therefore, we use an existing tag dictionary to construct the normalization dictionary.

Due to space limitations, we give only a brief overview of our construction method, omitting its details. We note that our method uses hand-crafted rules similar to those used in (Sasano et al., 2013); hence, the proposal of this method is not an important contribution. To make our experimental results reproducible, our normalization dictionary, as well as a tool for constructing it, is released as supplementary material.

Our method of constructing the normalization dictionary takes three steps. The following explains each step using Tables 3 and 4 as running examples.

Step 1 A tag dictionary generally contains a small number of ill-spelled words, although well-spelled words constitute a vast majority. We identify such ill-spelled words by using a manually-tailored list of surface POS tags indicative of informal spelling (*e.g.*, CONTRACTED VERB). For example, entry (c) in Table 4 is identified as an ill-spelled word in this step.

Step 2 The tag dictionary is augmented with normal forms and normal POS tags to construct a small normalization dictionary. For ill-spelled words identified in step 1, the normal forms and normal POS tags are determined by hand-crafted rules. For example, the normal form is derived by appending the vowel character "う" /u/ to the surface form, if the surface POS tag is CONTRACTED VERB. This rule derives entry (D) in Table 3 from entry (c) in Table 4. For well-spelled words, on the other hand, the normal forms and normal POS tags are simply set the same as the surface forms and surface POS tags. For example, entries (A), (C), and (E) in Table 3 are generated from entries (a), (b), and (d) in Table 4, respectively.

Step 3 Because the normalization dictionary constructed in step 2 contains only a few ill-spelled words, it is expanded in this step. For this purpose, we use hand-crafted rules to derive ill-spelled words from the entries already registered in the normalization dictionary. Some rules are taken from (Sasano et al., 2013), while the others are newly tailored. In Table 3, for example, entry (B) is derived from entry (A) by applying the rule that substitutes "ごい" /goi/ with "げえ" /gee/.

A small problem that arises in step 3 is how to handle lengthened words, such as entry (F) in Table 3. While lengthened words can be easily derived using simple rules (Brody and Diakopoulos, 2011; Sasano et al., 2013), such rules infinitely increase the number of entries because an unlimited number of lengthened words can be derived by repeating characters. To address this problem, no lengthened words are added to the normalization dictionary in step 3. We instead use rules to skip repetitive characters in an input sentence when performing dictionary match.

6.2 A hybrid approach

A word lattice is generated using both a statistical method (Kaji and Kitsuregawa, 2013) and the normalization dictionary.

We begin by generating a word lattice which encodes only word surface forms and surface POS tags (*c.f.*, Figure 1) using the statistical method proposed by Kaji and Kitsuregawa (2013). Interested readers may refer to their paper for details.

Each edge in the lattice is then labeled with normal forms and normal POS tags. Note that a single edge can have more than one candidate normal form and normal POS tag. In such a case, new edges are accordingly added to the lattice.

The edges are labeled with normal forms and normal POS tags in the following manner. First, every edge is labeled with a normal form and normal POS tag that are identical with the surface form and surface POS tag. This is based on our observation that most words are well-spelled ones. The edge is not provided with further normal forms and normal POS tags, if the normalization dictionary contains a well-spelled word that has the same surface form as the edge. Otherwise, we allow the edge to have all pairs of normal forms and normal POS tags that are obtained by using the normalization dictionary.

7 Discriminative Lattice Traversal

This section explains a discriminative model for traversing the word lattice. The lattice traversal with a discriminative model can formally be written as

$$(\boldsymbol{w}, \boldsymbol{t}, \boldsymbol{v}, \boldsymbol{s}) = \underset{(\boldsymbol{w},\boldsymbol{t},\boldsymbol{v},\boldsymbol{s}) \in \mathcal{L}(x)}{\arg\max} \boldsymbol{f}(x, \boldsymbol{w}, \boldsymbol{t}, \boldsymbol{v}, \boldsymbol{s}) \cdot \boldsymbol{\theta}.$$

Here, x denotes an input sentence, $\boldsymbol{w}$, $\boldsymbol{t}$, $\boldsymbol{v}$, and $\boldsymbol{s}$ denote a sequence of word surface forms, surface POS tags, normal forms, and normal POS tags, respectively, $\mathcal{L}(x)$ represents a set of candidate analyses represented by the word lattice, and $\boldsymbol{f}(\cdot)$ and $\boldsymbol{\theta}$ are feature and weight vectors.

We now describe features, a decoding method, and two training methods.

7.1 Features

We use character-level and word-level features used for word segmentation and POS tagging in (Kaji and Kitsuregawa, 2013). To take advantage of joint model with lexical normalization, the word-level features are extracted from not only surface forms but also normal forms. See (Kaji and Kitsuregawa, 2013) for the original features.

In addition, several new features are introduced in this paper. We use the quadruplets (w_i, t_i, v_i, s_i)

and pairs of surface and normal POS tags (t_i, s_i) as binary features to capture probable mappings between ill-spelled words and their well-spelled equivalents. We use another binary feature indicating whether a quadruplet (w_i, t_i, v_i, s_i) is registered in the normalization dictionary. Also, we use a bigram language model feature, which prevents sentences from being normalized into ungrammatical and/or incomprehensible ones. The language model features are associated with normalized bigrams, $(v_{i-1}, s_{i-1}, v_i, s_i)$, and take as the values the logarithmic frequency $\log_{10}(f+1)$, where f represents the bigram frequency (Kaji and Kitsuregawa, 2011). Since it is difficult to obtain a precise value of f, it is approximated by the frequency of the surface bigram, $(w_{i-1}, t_{i-1}, w_i, t_i)$, calculated from a large raw corpus automatically analyzed using a system of joint word segmentation and POS tagging. See Section 8.1 for the raw corpus and system used in the experiments.

7.2 Decoding

It is easy to find the best analysis $(\boldsymbol{w}, \boldsymbol{t}, \boldsymbol{v}, \boldsymbol{s})$ among the candidates represented by the word lattice. Although we use several new features, we can still locate the best analysis by using the same dynamic programming algorithm as in previous studies (Kudo et al., 2004; Kaji and Kitsuregawa, 2013).

7.3 Training on a fully annotated corpus

It is straightforward to train the joint model provided with a fully annotated corpus, which is labeled with word surface forms, surface POS tags, normal forms, and normal POS tags.

We use structured perceptron (Collins, 2002) for the training (Algorithm 1). The training begins by initializing $\boldsymbol{\theta}$ as a zero vector (line 1). It then reads the annotated corpus $\mathcal{C}$ (line 2-9). Given a training example, $(x, \boldsymbol{w}, \boldsymbol{t}, \boldsymbol{v}, \boldsymbol{s}) \in \mathcal{C}$, the algorithm locates the best analysis, $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{s}})$, based on the current weight vector (line 4). If the best analysis differs from the oracle analysis, $(\boldsymbol{w}, \boldsymbol{t}, \boldsymbol{v}, \boldsymbol{s})$, the weight vector is updated (line 5-7). After going through the annotated corpus m times (m=10 in our experiment), the averaged weight vector is returned (line 10).

7.4 Training on a partially annotated corpus

Although the training with the perceptron algorithm requires a fully annotated corpus, it is labor-intensive to fully annotate sentences. This consideration motivates us to explore training our model with less supervision. We specifically explore using a corpus annotated with only word boundaries and POS tags.

Algorithm 1 Perceptron training

1: $\boldsymbol{\theta} \leftarrow \mathbf{0}$
2: **for** $i = 1 \ldots m$ **do**
3: **for** $(x, \boldsymbol{w}, \boldsymbol{t}, \boldsymbol{v}, \boldsymbol{s}) \in \mathcal{C}$ **do**
4: $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{s}}) \leftarrow \textsc{Decoding}(x, \boldsymbol{\theta})$
5: **if** $(\boldsymbol{w}, \boldsymbol{t}, \boldsymbol{v}, \boldsymbol{s}) \neq (\hat{\boldsymbol{w}}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{s}})$ **then**
6: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{f}(x, \boldsymbol{w}, \boldsymbol{t}, \boldsymbol{v}, \boldsymbol{s}) - \boldsymbol{f}(x, \hat{\boldsymbol{w}}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{s}})$
7: **end if**
8: **end for**
9: **end for**
10: **return** $\textsc{Average}(\boldsymbol{\theta})$

Algorithm 2 Latent perceptron training

1: $\boldsymbol{\theta} \leftarrow \mathbf{0}$
2: **for** $i = 1 \ldots m$ **do**
3: **for** $(x, \boldsymbol{w}, \boldsymbol{t}) \in \mathcal{C}'$ **do**
4: $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{s}}) \leftarrow \textsc{Decoding}(x, \boldsymbol{\theta})$
5: $(\boldsymbol{w}, \boldsymbol{t}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{s}}) \leftarrow \textsc{ConstrainedDecoding}(x, \boldsymbol{\theta})$
6: **if** $\boldsymbol{w} \neq \hat{\boldsymbol{w}}$ or $\boldsymbol{t} \neq \hat{\boldsymbol{t}}$ **then**
7: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{f}(x, \boldsymbol{w}, \boldsymbol{t}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{s}}) - \boldsymbol{f}(x, \hat{\boldsymbol{w}}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{s}})$
8: **end if**
9: **end for**
10: **end for**
11: **return** $\textsc{Average}(\boldsymbol{\theta})$

We use the latent perceptron algorithm (Sun et al., 2013) to train the joint model from such a partially annotated corpus (Algorithm 2). In this scenario, a training example is a sentence x paired with a sequence of word surface forms $\boldsymbol{w}$ and surface POS tags $\boldsymbol{t}$ (*c.f.*, line 3). Similarly to the perceptron algorithm, we locate the best analysis $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{s}})$ for a given training example, (line 4). We also locate the best analysis, $(\boldsymbol{w}, \boldsymbol{t}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{s}})$, among those having the same surface forms $\boldsymbol{w}$ and surface POS tags $\boldsymbol{t}$ as the training example (line 5). If the surface forms and surface POS tags of the former analysis differ from the annotations of the training example, parameter is updated by regarding the latter analysis as an oracle (line 6-8).

8 Experiments

We conducted experiments to investigate how the microblog corpus and joint model contribute to improving accuracy of word segmentation and POS tagging in the microblog domain.

8.1 Setting

We constructed the normalization dictionary from the JUMAN dictionary 7.0.[9] While JUMAN dic-

[9] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN

tionary contains 750,156 entries, the normalization dictionary contains 112,458,326 entries.

Some features taken from the previous study (Kaji and Kitsuregawa, 2013) are induced using a tag dictionary. For this we used two tag dictionaries. One is JUMAN dictionary 7.0 and the other is a tag dictionary constructed by listing surface forms and surface POS tags in the normalization dictionary.

To compute the language model features, one billion sentences from Twitter posts were analyzed using MeCab 0.996.[10] We used all bigrams appearing at least 10 times in the auto-analyzed sentences.

8.2 Results of word segmentation and POS tagging

We first investigated the performance of models trained on an existing annotated corpus form news texts. For this experiment, our joint model as well as three state-of-the-art models (Kudo et al., 2004)[11](Neubig et al., 2011)[12](Kaji and Kitsuregawa, 2013) were trained on Kyoto University Text corpus 4.0 (Kurohashi and Nagao, 1998). Since this training corpus is not annotated with normal forms and normal POS tags, our model was trained using the latent perceptron. Table 5 summarizes the word-level F_1-scores (Kudo et al., 2004) on our microblog corpus. The two columns represent the results for word segmentation (**Seg**) and joint word segmentation and POS tagging (**Seg+Tag**), respectively.

We also conducted 5-fold crossvalidation on our microblog corpus to evaluate performance improvement when these models are trained on microblog texts (Table 6). In addition to the models in Table 5, results of a rule-based system (Sasano et al., 2013)[13] and our joint model trained using the perceptron algorithm are also presented. Notice that **Proposed** and **Proposed (latent)** represent our model trained using perceptron and latent perceptron, respectively.

From Tables 5 and 6, as expected, we see that the models trained on news texts performed poorly on microblog texts, while their performance significantly boosted when trained on the microblog texts. This demonstrates the importance of corpus annotation. An exception was **Kudo04**. Its performance improved only slightly, even when it was trained on the microblog texts. We believe this is because their model uses dictionary-based rules to prune candidate analyses; thus, it could not perform well in the microblog domain, where out-of-vocabulary words are abundant.

Table 5: Performance of models trained on the news articles.

	Seg	Seg+Tag
Kudo04	81.8	71.0
Neubig11	80.5	69.1
Kaji13	**83.2**	73.1
Proposed (latent)	83.0	**73.9**

Table 6 also illustrates that our joint models achieved F_1-score better than the state-of-the-art models trained on the microblog texts. This shows that modeling the derivation process of ill-spelled words makes training easier. We conducted bootstrap resampling (with 1000 samples) to investigate the significance of the improvements achieved with our joint model. The results showed that all improvements over the baselines were statistically significant ($p < 0.01$). The difference between **Proposed** and **Proposed (latent)** were also statistically significant ($p < 0.01$).

The results of **Proposed (latent)** are interesting. Table 5 illustrates that our joint model performs well even when it is trained on a news corpus that rarely contains ill-spelled words and is not at all annotated with normal forms and normal POS tags. This indicates the robustness of our training method and the importance of modeling word derivation process in the microblog domain. In Table 6, we observed that **Proposed (latent)**, which uses less supervision, performed better than **Proposed**. The reason for this will be examined later.

In summary, we can conclude that both the microblog corpus and joint model significantly contribute to training accurate models for word segmentation and POS tagging in the microblog domain.

8.3 Results of lexical normalization

While the main goal with this study was to enhance word segmentation and POS tagging in the microblog domain, it is interesting to explore how well our joint model can normalize ill-spelled words.

Table 7 illustrates precision, recall, and F_1-score for the lexical normalization task. To put

[10] https://code.google.com/p/mecab
[11] https://code.google.com/p/mecab
[12] http://www.phontron.com/kytea/
[13] http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN

Table 6: Results of 5-fold cross-validation on microblog corpus.

	Seg	Seg+Tag
Kudo04	82.7	71.7
Neubig11	88.6	75.9
Kaji13	90.9	82.1
Sasano13	82.7	73.3
Proposed	91.3	83.2
Proposed (latent)	**91.4**	**83.7**

Table 7: Results of lexical normalization task in terms of precision, recall, and F_1-score.

	Precision	Recall	F_1
Neubig11	69.2	35.9	47.3
Proposed	**77.1**	**44.6**	**56.6**
Proposed (latent)	53.7	24.7	33.9

the results into context, we report on the baseline results of a tagging model proposed by Neubig et al. (2011). This baseline conducts lexical normalization by regarding it as two independent tagging tasks (*i.e.*, tasks of tagging normal forms and normal POS tags). The result of the baseline model is also obtained using 5-fold crossvalidation.

Table 7 illustrates that **Proposed** performed significantly better than the simple tagging model, **Neubig11**. This suggests the effectiveness of our joint model. On the other hand, **Proposed (latent)** performed poorly in this task. From this result, we can argue that **Proposed (latent)** can achieve superior performance in word segmentation and POS tagging (Table 6) because it gave up correctly normalizing ill-spelled words, focusing on word segmentation and POS tagging.

The experimental results so far suggest the following strategy for training our joint model. If accuracy of word segmentation and POS tagging is the main concern, we can use the latent perceptron. This approach has the advantage of being able to use a partially annotated corpus. On the other hand, if performance of lexical normalization is crucial, we have to use the standard perceptron algorithm.

8.4 Error analysis

We manually analyzed erroneous outputs and observed several tendencies.

We found that a word lattice sometimes missed the correct output. Such an error was, for example, observed in a sentence including many ill-spelled words, e.g., '周囲の目が、キニナリマス！ (be nervous about what other people think!)', where the part 'キニナリマス' is in ill-spelled words. Improving the lattice generation algorithm is considered necessary to achieve further performance gain.

Even if the correct analysis appears in the word lattice, our model sometimes failed to handle ill-spelled words, incorrectly analyzing them as out-of-vocabulary words. For example, the proposed method treated the phrase 'おやつたーいむ (snack time)' as a single out-of-vocabulary word, even though the correct analysis was found in the word lattice. More sophisticated features would be required to accurately distinguish between ill-spelled and out-of-vocabulary words.

9 Conclusion and Future Work

We presented our attempts towards developing an accurate model for word segmentation and POS tagging in the microblog domain. To this end, we, for the first time, developed an annotated corpus of microblogs. We also proposed a joint model with lexical normalization to handle orthographic diversity in the microblog text. Intensive experiments demonstrated that we could successfully improve the performance of word segmentation and POS tagging on microblog texts. We believe this study will have a large practical impact on a various research areas that target microblogs.

One limitation of our approach is that it cannot handle certain types of ill-spelled words. For example, the current model cannot handle the cases in which there are no one-to-one-mappings between well-spelled and ill-spelled words. Also, our model cannot handle spelling errors, which are considered relatively frequent in the microblog than news domains. The treatment of these problems would require further research.

Another future research is to speed-up our model. Since the joint model with lexical normalization significantly increases the search space, it is much slower than the original lattice-based model for word segmentation and POS tagging.

Acknowledgments

The authors would like to thank Naoki Yoshinaga for his help in developing the microblog corpus as well as fruitful discussions.

References

Samuel Brody and Nicholas Diakopoulos. 2011. Cooooooooooooooollllllllllll!!!!!!!!!!!!!! using word lengthening to detect sentiment in microblogs. In *Proceedings of EMNLP*, pages 562–570.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8.

Huiming Duan, Zhifang Sui, Ye Tian, and Wenjie Li. 2012. The CIPS-SIGHAN CLP 2012 Chinese word segmentation on microblog corpora bakeoff. In *Proceedings of the Second CIPS-SIGHAN Joint Conrerence on Chinese Language Processing*, pages 35–40.

Jennifer Foster, Ozlem Cetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. #hardtoparse: POS tagging and parsing the twitterverse. In *Proceedings of AAAI Workshop on Analysing Microtext*, pages 20–25.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of ACL*, pages 42–47.

Bo Han and Timothy Baldwin. 2011. Lexical normalization of short text messages: Makin sens a #twitter. In *Proceedings of ACL*, pages 368–378.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of EMNLP-CoNLL*, pages 421–432.

Chikara Hashimoto, Sadao Kurohashi, Daisuke Kawahara, Keiji Shinzato, and Masaaki Nagata. 2011. Construction of a blog corpus with syntactic, anaphoric, and semantic annotations (in Japanese). *Journal of Natural Language Processing*, 18(2):175–201.

Yijiu How and Min-Yen Kan. 2005. Optimizing predictive text entry for short message service on mobile phones. In *Proceedings of Human Computer Interfaces International*.

Kazushi Ikeda, Tadashi Yanagihara, Kazunori Matsumoto, and Yasuhiro Takishima. 2009. Unsupervised text normalization approach for morphological analysis of blog documents. In *Proceedings of Australasian Joint Conference on Advances in Artificial Intelligence*, pages 401–411.

Wenbin Jiang, Haitao Mi, and Qun Liu. 2008. Word lattice reranking for Chinese word segmentation and part-of-speech tagging. In *Proceedings of Coling*, pages 385–392.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent Twitter sentiment classification. In *Proceedings of ACL*, pages 151–160.

Nobuhiro Kaji and Masaru Kitsuregawa. 2011. Splitting noun compounds via monolingual and bilingual paraphrasing: A study on Japanese Katakana words. In *Proceedings of EMNLP*, pages 959–969.

Nobuhiro Kaji and Masaru Kitsuregawa. 2013. Efficient word lattice generation for joint word segmentation and POS tagging in Japanese. In *Proceedings of IJCNLP*, pages 153–161.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of EMNLP*, pages 230–237.

Sadao Kurohashi and Makoto Nagao. 1998. Building a Japanese parsed corpus while improving the parsing system. In *Proceedings of LREC*, pages 719–724.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2013. Paraphrasing 4 microblog normalization. In *Proceedings of EMNLP*, pages 73–84.

Xiaohua Liu, Ming Zhou, Xiangyang Zhou, Zhongyang Fu, and Furu Wei. 2012. Joint inference of named entity recognition and normalization for tweets. In *Proceedings of ACL*, pages 526–535.

Graham Neubig, Yousuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust adaptable Japanese morphological analysis. In *Proceedings of ACL*, pages 529–533.

Itsumi Saito, Kugatsu Sadamitsu, Hisako Asano, and Yoshihiro Matsuo. 2014. Morphological analysis for Japanese noisy text based on character-level and word-level normalization. In *Proceedings of COLING*, pages 1773–1782.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquak shakes Twitter users: real-time event detection by social sensors. In *Proceedings of WWW*, pages 851–860.

Ryohei Sasano, Sadao Kurohashi, and Manabu Okumura. 2013. A simple approach to unknown word processing in Japanese morphological analysis. In *Proceedings of IJCNLP*, pages 162–170.

Xu Sun, Takuya Matsuzaki, and Wenjie Li. 2013. Latent structured perceptrons for large-scale learning with hidden information. *IEEE Transactions on Knowledge and Data Engineering*, 25(9):2063–2075.

Aobo Wang and Min-Yen Kan. 2013. Mining informal language from Chinese microtext: Joint word recognition and segmentation. In *Proceedings of ACL*, pages 731–741.

Pidong Wang and Hwee Tou Ng. 2013. A beam-search decoder for normalization of social media text with application to machine translation. In *Proceedings of NAACL*, pages 471–481.

Aobo Wang, Min-Yen Kan, Daniel Andrade, Takashi Onishi, and Kai Ishikawa. 2013. Chinese informal word normalization: an experimental study. In *Proceedings of IJCNLP*, pages 127–135.

Yunqing Xia, Kam-Fai Wong, and Wenjie Li. 2006. A phonetic-based approach to Chinese chat text normalization. In *Proceedings of ACL*, pages 993–1000.

Yi Yang and Jacob Eisenstein. 2013. A log-linear model for unsupervised text normalization. In *Proceedings of EMNLP*, pages 61–72.

Congle Zhang, Tyler Baldwin, Howard Ho, Benny Kimelfeld, and Yunyao Li. 2013. Adaptive parser-centric text normalization. In *Proceedings of ACL*, pages 1159–1168.

Revisiting Embedding Features for Simple Semi-supervised Learning

Jiang Guo[†], **Wanxiang Che**[†], **Haifeng Wang**[‡], **Ting Liu**[†*]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
[‡]Baidu Inc., Beijing, China
{jguo, car, tliu}@ir.hit.edu.cn
wanghaifeng@baidu.com

Abstract

Recent work has shown success in using continuous word embeddings learned from unlabeled data as features to improve supervised NLP systems, which is regarded as a simple semi-supervised learning mechanism. However, fundamental problems on effectively incorporating the word embedding features within the framework of linear models remain. In this study, we investigate and analyze three different approaches, including a new proposed *distributional prototype* approach, for utilizing the embedding features. The presented approaches can be integrated into most of the classical linear models in NLP. Experiments on the task of named entity recognition show that each of the proposed approaches can better utilize the word embedding features, among which the *distributional prototype* approach performs the best. Moreover, the combination of the approaches provides additive improvements, outperforming the dense and continuous embedding features by nearly 2 points of F1 score.

1 Introduction

Learning generalized representation of words is an effective way of handling data sparsity caused by high-dimensional lexical features in NLP systems, such as named entity recognition (NER) and dependency parsing. As a typical low-dimensional and generalized word representation, Brown clustering of words has been studied for a long time. For example, Liang (2005) and Koo et al. (2008) used the Brown cluster features for semi-supervised learning of various NLP tasks and achieved significant improvements.

*Email correspondence.

Recent research has focused on a special family of word representations, named "word embeddings". Word embeddings are conventionally defined as dense, continuous, and low-dimensional vector representations of words. Word embeddings can be learned from large-scale unlabeled texts through context-predicting models (e.g., neural network language models) or spectral methods (e.g., canonical correlation analysis) in an unsupervised setting.

Compared with the so-called *one-hot* representation where each word is represented as a sparse vector of the same size of the vocabulary and only one dimension is on, word embedding preserves rich linguistic regularities of words with each dimension hopefully representing a latent feature. Similar words are expected to be distributed close to one another in the embedding space. Consequently, word embeddings can be beneficial for a variety of NLP applications in different ways, among which the most simple and general way is to be fed as features to enhance existing supervised NLP systems.

Previous work has demonstrated effectiveness of the continuous word embedding features in several tasks such as chunking and NER using generalized linear models (Turian et al., 2010).[1] However, there still remain two fundamental problems that should be addressed:

- Are the continuous embedding features fit for the generalized linear models that are most widely adopted in NLP?
- How can the generalized linear models better utilize the embedding features?

According to the results provided by Turian et

[1]Generalized linear models refer to the models that describe the data as a combination of linear basis functions, either directly in the input variables space or through some transformation of the probability distributions (e.g., *log-linear* models).

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 110–120,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

al. (2010), the embedding features brought significantly less improvement than Brown clustering features. This result is actually not reasonable because the expressing power of word embeddings is theoretically stronger than clustering-based representations which can be regarded as a kind of *one-hot* representation but over a low-dimensional vocabulary (Bengio et al., 2013).

Wang and Manning (2013) showed that linear architectures perform better in high-dimensional discrete feature space than non-linear ones, whereas non-linear architectures are more effective in low-dimensional and continuous feature space. Hence, the previous method that directly uses the continuous word embeddings as features in linear models (CRF) is inappropriate. Word embeddings may be better utilized in the linear modeling framework by smartly transforming the embeddings to some relatively higher dimensional and discrete representations.

Driven by this motivation, we present three different approaches: *binarization* (Section 3.2), *clustering* (Section 3.3) and a new proposed *distributional prototype* method (Section 3.4) for better incorporating the embeddings features. In the *binarization* approach, we directly binarize the continuous word embeddings by dimension. In the *clustering* approach, we cluster words based on their embeddings and use the resulting word cluster features instead. In the *distributional prototype* approach, we derive *task-specific* features from word embeddings by utilizing a set of automatically extracted *prototypes* for each target label.

We carefully compare and analyze these approaches in the task of NER. Experimental results are promising. With each of the three approaches, we achieve higher performance than directly using the continuous embedding features, among which the *distributional prototype* approach performs the best. Furthermore, by putting the most effective two of these features together, we finally outperform the continuous embedding features by nearly 2 points of F1 Score (86.21% vs. 88.11%).

The major contribution of this paper is twofold. (1) We investigate various approaches that can better utilize word embeddings for semi-supervised learning. (2) We propose a novel *distributional prototype* approach that shows the great potential of word embedding features. All the presented approaches can be easily integrated into most of the classical linear NLP models.

2 Semi-supervised Learning with Word Embeddings

Statistical modeling has achieved great success in most NLP tasks. However, there still remain some major unsolved problems and challenges, among which the most widely concerned is the data sparsity problem. Data sparsity in NLP is mainly caused by two factors, namely, the lack of labeled training data and the Zipf distribution of words. On the one hand, large-scale labeled training data are typically difficult to obtain, especially for structure prediction tasks, such as syntactic parsing. Therefore, the supervised models can only see limited examples and thus make biased estimation. On the other hand, the natural language words are Zipf distributed, which means that most of the words appear a few times or are completely absent in our texts. For these low-frequency words, the corresponding parameters usually cannot be fully trained.

More foundationally, the reason for the above factors lies in the high-dimensional and sparse lexical feature representation, which completely ignores the similarity between features, especially word features. To overcome this weakness, an effective way is to learn more generalized representations of words by exploiting the numerous unlabeled data, in a semi-supervised manner. After which, the generalized word representations can be used as extra features to facilitate the supervised systems.

Liang (2005) learned Brown clusters of words (Brown et al., 1992) from unlabeled data and use them as features to promote the supervised NER and Chinese word segmentation. Brown clusters of words can be seen as a generalized word representation distributed in a discrete and low-dimensional vocabulary space. Contextually similar words are grouped in the same cluster. The Brown clustering of words was also adopted in dependency parsing (Koo et al., 2008) and POS tagging for online conversational text (Owoputi et al., 2013), demonstrating significant improvements.

Recently, another kind of word representation named "word embeddings" has been widely studied (Bengio et al., 2003; Mnih and Hinton, 2008). Using word embeddings, we can evaluate the similarity of two words straightforward by computing the dot-product of two numerical vectors in the Hilbert space. Two *similar* words are expected to

be distributed close to each other.[2]

Word embeddings can be useful as input to an NLP model (mostly non-linear) or as additional features to enhance existing systems. Collobert et al. (2011) used word embeddings as input to a deep neural network for multi-task learning. Despite of the effectiveness, such non-linear models are hard to build and optimize. Besides, these architectures are often specialized for a certain task and not scalable to general tasks. A simple and more general way is to feed word embeddings as augmented features to an existing supervised system, which is similar to the semi-supervised learning with Brown clusters.

As discussed in Section 1, Turian et al. (2010) is the pioneering work on using word embedding features for semi-supervised learning. However, their approach cannot fully exploit the potential of word embeddings. We revisit this problem in this study and investigate three different approaches for better utilizing word embeddings in semi-supervised learning.

3 Approaches for Utilizing Embedding Features

3.1 Word Embedding Training

In this paper, we will consider a context-predicting model, more specifically, the *Skip-gram* model (Mikolov et al., 2013a; Mikolov et al., 2013b) for learning word embeddings, since it is much more efficient as well as memory-saving than other approaches.

Let's denote the embedding matrix to be learned by $C_{d\times N}$, where N is the vocabulary size and d is the dimension of word embeddings. Each column of C represents the embedding of a word. The *Skip-gram* model takes the current word w as input, and predicts the probability distribution of its context words within a fixed window size. Concretely, w is first mapped to its embedding v_w by selecting the corresponding column vector of C (or multiplying C with the *one-hot* vector of w). The probability of its context word c is then computed using a log-linear function:

$$P(c|w;\theta) = \frac{exp({v_c}^\top v_w)}{\sum_{c'\in V} exp({v_{c'}}^\top v_w)} \qquad (1)$$

where V is the vocabulary. The parameters θ are v_{w_i}, v_{c_i} for $w, c \in V$ and $i = 1, ..., d$. Then, the log-likelihood over the entire training dataset D can be computed as:

$$J(\theta) = \sum_{(w,c)\in D} \log p(c|w;\theta) \qquad (2)$$

The model can be trained by maximizing $J(\theta)$.

Here, we suppose that the word embeddings have already been trained from large-scale unlabeled texts. We will introduce various approaches for utilizing the word embeddings as features for semi-supervised learning. The main idea, as introduced in Section 1, is to transform the continuous word embeddings to some relatively higher dimensional and discrete representations. The direct use of continuous embeddings as features (Turian et al., 2010) will serve as our baseline setting.

3.2 Binarization of Embeddings

One fairly natural approach for converting the continuous-valued word embeddings to discrete values is binarization by dimension.

Formally, we aim to convert the continuous-valued embedding matrix $C_{d\times N}$, to another matrix $M_{d\times N}$ which is discrete-valued. There are various conversion functions. Here, we consider a simple one. For the i^{th} dimension of the word embeddings, we divide the corresponding row vector C_i into two halves for positive (C_{i+}) and negative (C_{i-}), respectively. The conversion function is then defined as follows:

$$M_{ij} = \phi(C_{ij}) = \begin{cases} U_+, & if\ C_{ij} \geq mean(C_{i+}) \\ B_-, & if\ C_{ij} \leq mean(C_{i-}) \\ 0, & otherwise \end{cases}$$

where $mean(v)$ is the mean value of vector v, U_+ is a string feature which turns on when the value (C_{ij}) falls into the upper part of the positive list. Similarly, B_- refers to the bottom part of the negative list. The insight behind ϕ is that we only consider the features with strong opinions (i.e., positive or negative) on each dimension and omit the values close to zero.

3.3 Clustering of Embeddings

Yu et al. (2013) introduced clustering embeddings to overcome the disadvantage that word embeddings are not suitable for linear models. They suggested that the high-dimensional cluster features make samples from different classes better separated by linear models.

[2] The term *similar* should be viewed depending on the specific task.

In this study, we again investigate this approach. Concretely, each word is treated as a single sample. The batch k-means clustering algorithm (Sculley, 2010) is used,[3] and each cluster is represented as the mean of the embeddings of words assigned to it. Similarities between words and clusters are measured by Euclidean distance.

Moreover, different number of clusters n contain information of different granularities. Therefore, we combine the cluster features of different ns to better utilize the embeddings.

3.4 Distributional Prototype Features

We propose a novel kind of embedding features, named distributional prototype features for supervised models. This is mainly inspired by *prototype-driven learning* (Haghighi and Klein, 2006) which was originally introduced as a primarily unsupervised approach for sequence modeling. In *prototype-driven learning*, a few prototypical examples are specified for each target label, which can be treated as an injection of prior knowledge. This sparse prototype information is then propagated across an unlabeled corpus through distributional similarities.

The basic motivation of the distributional prototype features is that similar words are supposed to be tagged with the same label. This hypothesis makes great sense in tasks such as NER and POS tagging. For example, suppose *Michael* is a prototype of the named entity (NE) type `PER`. Using the distributional similarity, we could link similar words to the same prototypes, so the word *David* can be linked to *Michael* because the two words have high similarity (exceeds a threshold). Using this link feature, the model will push *David* closer to `PER`.

To derive the distributional prototype features, first, we need to construct a few canonical examples (prototypes) for each target annotation label. We use the normalized pointwise mutual information (NPMI) (Bouma, 2009) between the label and word, which is a smoothing version of the standard PMI, to decide the prototypes of each label. Given the annotated training corpus, the NPMI between a label and word is computed as follows:

$$\lambda_n(label, word) = \frac{\lambda(label, word)}{-\ln p(label, word)} \quad (3)$$

[3] `code.google.com/p/sofia-ml`

NE Type	Prototypes
B-PER	Mark, Michael, David, Paul
I-PER	Akram, Ahmed, Khan, Younis
B-ORG	Reuters, U.N., Ajax, PSV
I-ORG	Newsroom, Inc, Corp, Party
B-LOC	U.S., Germany, Britain, Australia
I-LOC	States, Republic, Africa, Lanka
B-MISC	Russian, German, French, British
I-MISC	Cup, Open, League, OPEN
O	., ,, the, to

Table 1: Prototypes extracted from the CoNLL-2003 NER training data using NPMI.

where,

$$\lambda(label, word) = \ln \frac{p(label, word)}{p(label)p(word)} \quad (4)$$

is the standard PMI.

For each target label l (e.g., `PER`, `ORG`, `LOC`), we compute the NPMI of l and all words in the vocabulary, and the top m words are chosen as the prototypes of l. We should note that the prototypes are extracted fully automatically, without introducing additional human prior knowledge.

Table 1 shows the top four prototypes extracted from the NER training corpus of CoNLL-2003 shared task (Tjong Kim Sang and De Meulder, 2003), which contains four NE types, namely, `PER`, `ORG`, `LOC`, and `MISC`. Non-NEs are denoted by `O`. We convert the original annotation to the standard BIO-style. Thus, the final corpus contains nine labels in total.

Next, we introduce the prototypes as features to our supervised model. We denote the set of prototypes for all target labels by S_p. For each prototype $z \in S_p$, we add a predicate $proto = z$, which becomes active at each w if the distributional similarity between z and w ($DistSim(z, w)$) is above some threshold. $DistSim(z, w)$ can be efficiently calculated through the *cosine* similarity of the embeddings of z and w. Figure 1 gives an illustration of the distributional prototype features. Unlike previous embedding features or Brown clusters, the distributional prototype features are *task-specific* because the prototypes of each label are extracted from the training data.

Moreover, each prototype word is also its own prototype (since a word has maximum similarity to itself). Thus, if the prototype is closely related to a label, all the words that are distributionally

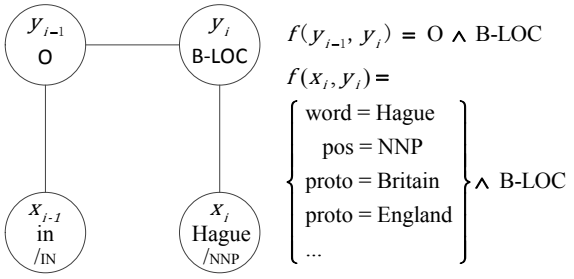


Figure 1: An example of distributional prototype features for NER.

similar to that prototype are pushed towards that label.

4 Supervised Evaluation Task

Various tasks can be considered to compare and analyze the effectiveness of the above three approaches. In this study, we partly follow Turian et al. (2010) and Yu et al. (2013), and take NER as the supervised evaluation task.

NER identifies and classifies the named entities such as the names of persons, locations, and organizations in text. The state-of-the-art systems typically treat NER as a sequence labeling problem, where each word is tagged either as a BIO-style NE or a non-NE category.

Here, we use the linear chain CRF model, which is most widely used for sequence modeling in the field of NLP. The CoNLL-2003 shared task dataset from the Reuters, which was used by Turian et al. (2010) and Yu et al. (2013), was chosen as our evaluation dataset. The training set contains 14,987 sentences, the development set contains 3,466 sentences and is used for parameter tuning, and the test set contains 3,684 sentences.

The baseline features are shown in Table 2.

4.1 Embedding Feature Templates

In this section, we introduce the embedding features to the baseline NER system, turning the supervised approach into a semi-supervised one.

Dense embedding features. The dense continuous embedding features can be fed directly to the CRF model. These embedding features can be seen as heterogeneous features from the existing baseline features, which are discrete. There is no effective way for dense embedding features to be combined internally or with other discrete features. So we only use the unigram embedding features following Turian et al. (2010). Concretely, the embedding feature template is:

Baseline NER Feature Templates
00: $w_{i+k}, -2 \leq k \leq 2$
01: $w_{i+k} \circ w_{i+k+1}, -2 \leq k \leq 1$
02: $t_{i+k}, -2 \leq k \leq 2$
03: $t_{i+k} \circ t_{i+k+1}, -2 \leq k \leq 1$
04: $chk_{i+k}, -2 \leq k \leq 2$
05: $chk_{i+k} \circ chk_{i+k+1}, -2 \leq k \leq 1$
06: $Prefix(w_{i+k}, l), -2 \leq k \leq 2, 1 \leq l \leq 4$
07: $Suffix(w_{i+k}, l), -2 \leq k \leq 2, 1 \leq l \leq 4$
08: $Type(w_{i+k}), -2 \leq k \leq 2$
Unigram Features
$y_i \circ 00 - 08$
Bigram Features
$y_{i-1} \circ y_i$

Table 2: Features used in the NER system. t is the POS tag. chk is the chunking tag. $Prefix$ and $Suffix$ are the first and last l characters of a word. $Type$ indicates if the word is all-capitalized, is-capitalized, all-digits, etc.

- $de_{i+k}[d]$, $-2 \leq k \leq 2$, d ranges over the dimensions of the dense word embedding de.

Binarized embedding features. The binarized embedding feature template is similar to the dense one. The only difference is that the feature values are discrete and we omit dimensions with zero value. Therefore, the feature template becomes:

- $bi_{i+k}[d]$, $-2 \leq k \leq 2$, where $bi_{i+k}[d] \neq 0$, d ranges over the dimensions of the binarized vector bi of word embedding.

In this way, the dimension of the binarized embedding feature space becomes $2 \times d$ compared with the originally d of the dense embeddings.

Compound cluster features. The advantage of the cluster features is that they can be combined internally or with other features to form compound features, which can be more discriminative. Furthermore, the number of resulting clusters n can be tuned, and different ns indicate different granularities. Concretely, the compound cluster feature template for each specific n is:

- $c_{i+k}, -2 \leq k \leq 2$.
- $c_{i+k} \circ c_{i+k+1}, -2 \leq k \leq 1$.
- $c_{i-1} \circ c_{i+1}$.

Distributional prototype features. The set of prototypes is again denoted by S_p, which is de-

cided by selecting the top m (NPMI) words as prototypes of each label, where m is tuned on the development set. For each word w_i in a sequence, we compute the distributional similarity between w_i and each prototype in S_p and select the prototypes zs that $DistSim(z, w) \geq \delta$. We set $\delta = 0.5$ without manual tuning. The distributional prototype feature template is then:

- $\{proto_{i+k}{=}z \mid DistSim(w_{i+k}, z) \geq \delta \;\&\; z \in S_p\}, -2 \leq k \leq 2$.

We only use the unigram features, since the number of active distributional prototype features varies for different words (positions). Hence, these features cannot be combined effectively.

4.2 Brown Clustering

Brown clustering has achieved great success in various NLP applications. At most time, it provides a strong baseline that is difficult to beat (Turian et al., 2010). Consequently, in our study, we conduct comparisons among the embedding features and the Brown clustering features, along with further investigations of their combination.

The Brown algorithm is a hierarchical clustering algorithm which optimizes a class-based bigram language model defined on the word clusters (Brown et al., 1992). The output of the Brown algorithm is a binary tree, where each word is uniquely identified by its path from the root. Thus each word can be represented as a bit-string with a specific length.

Following the setting of Owoputi et al. (2013), we will use the prefix features of hierarchical clusters to take advantage of the word similarity in different granularities. Concretely, the Brown cluster feature template is:

- bc_{i+k}, $-2 \leq k \leq 2$.
- $prefix(bc_{i+k}, p)$, $p \in \{2,4,6,...,16\}$, $-2 \leq k \leq 2$. $prefix$ takes the p-length prefix of the Brown cluster coding bc_{i+k}.

5 Experiments

5.1 Experimental Setting

We take the English Wikipedia until August 2012 as our unlabeled data to train the word embeddings.[4] Little pre-processing is conducted for the training of word embeddings. We remove paragraphs that contain non-roman characters and all MediaWiki markups. The resulting text is tokenized using the Stanford tokenizer,[5] and every word is converted to lowercase. The final dataset contains about 30 million sentences and 1.52 billion words. We use a dictionary that contains 212,779 most common words (frequency ≥ 80) in the dataset. An efficient open-source implementation of the *Skip-gram* model is adopted.[6] We apply the negative sampling[7] method for optimization, and the asynchronous stochastic gradient descent algorithm (Asynchronous SGD) for parallel weight updating. In this study, we set the dimension of the word embeddings to 50. Higher dimension is supposed to bring more improvements in semi-supervised learning, but its comparison is beyond the scope of this paper.

For the cluster features, we tune the number of clusters n from 500 to 3000 on the development set, and finally use the combination of $n = 500, 1000, 1500, 2000, 3000$, which achieves the best results. For the distributional prototype features, we use a fixed number of prototype words (m) for each target label. m is tuned on the development set and is finally set to 40.

We induce 1,000 brown clusters of words, the setting in prior work (Koo et al., 2008; Turian et al., 2010). The training data of brown clustering is the same with that of training word embeddings.

5.2 Results

Table 3 shows the performances of NER on the test dataset. Our baseline is slightly lower than that of Turian et al. (2010), because they use the BILOU encoding of NE types which outperforms BIO encoding (Ratinov and Roth, 2009).[8] Nonetheless, our conclusions hold. As we can see, all of the three approaches we investigate in this study achieve better performance than the direct use of the dense continuous embedding features.

To our surprise, even the binarized embedding features (BinarizedEmb) outperform the continuous version (DenseEmb). This provides clear evidence that directly using the dense continuous embeddings as features in CRF indeed cannot fully

[4] download.wikimedia.org.

[5] nlp.stanford.edu/software/tokenizer.shtml.

[6] code.google.com/p/word2vec/.

[7] More details are analyzed in (Goldberg and Levy, 2014).

[8] We use BIO encoding here in order to compare with most of the reported benchmarks.

Setting	F1
Baseline	83.43
+DenseEmb†	86.21
+BinarizedEmb	86.75
+ClusterEmb	86.90
+DistPrototype	**87.44**
+BinarizedEmb+ClusterEmb	87.56
+BinarizedEmb+DistPrototype	87.46
+ClusterEmb+DistPrototype	**88.11**
+Brown	87.49
+Brown+ClusterEmb	88.17
+Brown+DistPrototype	88.04
+Brown+ClusterEmb+DistPrototype	**88.58**
Finkel et al. (2005)	86.86
Krishnan and Manning (2006)	87.24
Ando and Zhang (2005)	89.31
Collobert et al. (2011)	88.67

Table 3: The performance of semi-supervised NER on the CoNLL-2003 test data, using various embedding features. † DenseEmb refers to the method used by Turian et al. (2010), i.e., the direct use of the dense and continuous embeddings.

exploit the potential of word embeddings. The compound cluster features (ClusterEmb) also outperform the DenseEmb. The same result is also shown in (Yu et al., 2013). Further, the distributional prototype features (DistPrototype) achieve the best performance among the three approaches (1.23% higher than DenseEmb).

We should note that the feature templates used for BinarizedEmb and DistPrototype are merely unigram features. However, for ClusterEmb, we form more complex features by combining the clusters of the context words. We also consider different number of clusters n, to take advantage of the different granularities. Consequently, the dimension of the cluster features is much higher than that of BinarizedEmb and DistPrototype.

We further combine the proposed features to see if they are complementary to each other. As shown in Table 3, the cluster and distributional prototype features are the most complementary, whereas the binarized embedding features seem to have large overlap with the distributional prototype features. By combining the cluster and distributional prototype features, we further push the performance to 88.11%, which is nearly two points higher than the performance of the dense embedding features (86.21%).[9]

We also compare the proposed features with the Brown cluster features. As shown in Table 3, the distributional prototype features alone achieve comparable performance with the Brown clusters. When the cluster and distributional prototype features are used together, we outperform the Brown clusters. This result is inspiring because we show that the embedding features indeed have stronger expressing power than the Brown clusters, as desired. Finally, by combining the Brown cluster features and the proposed embedding features, the performance can be improved further (88.58%). The binarized embedding features are not included in the final compound features because they are almost overlapped with the distributional prototype features in performance.

We also summarize some of the reported benchmarks that utilize unlabeled data (with no gazetteers used), including the Stanford NER tagger (Finkel et al. (2005) and Krishnan and Manning (2006)) with distributional similarity features. Ando and Zhang (2005) use unlabeled data for constructing auxiliary problems that are expected to capture a good feature representation of the target problem. Collobert et al. (2011) adjust the feature embeddings according to the specific task in a deep neural network architecture. We can see that both Ando and Zhang (2005) and Collobert et al. (2011) learn *task-specific* lexical features, which is similar to the proposed *distributional prototype* method in our study. We suggest this to be the main reason for the superiority of these methods.

Another advantage of the proposed discrete features over the dense continuous features is tagging efficiency. Table 4 shows the running time using different kinds of embedding features. We achieve a significant reduction of the tagging time per sentence when using the discrete features. This is mainly due to the dense/sparse battle. Although the dense embedding features are low-dimensional, the feature vector for each word is much denser than in the sparse and discrete feature space. Therefore, we actually need much more computation during decoding. Similar results can be observed in the comparison of the DistPrototype and ClusterEmb features, since the density of the DistPrototype features is higher. It is possible

[9] Statistical significant with p-value < 0.001 by two-tailed t-test.

Setting	Time (ms) / sent
Baseline	1.04
+DenseEmb	4.75
+BinarizedEmb	1.25
+ClusterEmb	1.16
+DistPrototype	2.31

Table 4: Running time of different features on a Intel(R) Xeon(R) E5620 2.40GHz machine.

to accelerate the DistPrototype, by increasing the threshold of $DistSim(z, w)$. However, this is indeed an issue of trade-off between efficiency and accuracy.

5.3 Analysis

In this section, we conduct analyses to show the reasons for the improvements.

5.3.1 Rare words

As discussed by Turian et al. (2010), much of the NER F1 is derived from decisions regarding rare words. Therefore, in order to show that the three proposed embedding features have stronger ability for handling rare words, we first conduct analysis for the tagging errors of words with different frequency in the unlabeled data. We assign the word frequencies to several buckets, and evaluate the per-token errors that occurred in each bucket. Results are shown in Figure 2. In most cases, all three embedding features result in fewer errors on rare words than the direct use of dense continuous embedding features.

Interestingly, we find that for words that are extremely rare (0–256), the binarized embedding features incur significantly fewer errors than other approaches. As we know, the embeddings for the rare words are close to their initial value, because they received few updates during training. Hence, these words are not fully trained. In this case, we would like to omit these features because their embeddings are not even trustable. However, all embedding features that we proposed except BinarizedEmb are unable to handle this.

In order to see how much we have utilized the embedding features in BinarizedEmb, we calculate the sparsity of the binarized embedding vectors, i.e., the ratio of zero values in each vector (Section 3.2). As demonstrated in Figure 3, the sparsity-frequency curve has good properties: higher sparsity for very rare words and very frequent words, while lower sparsity for mid-frequent words. It indicates that for words that are very rare or very frequent, BinarizedEmb just omit most of the features. This is reasonable also for the very frequent words, since they usually have rich and diverse context distributions and their embeddings cannot be well learned by our models (Huang et al., 2012).

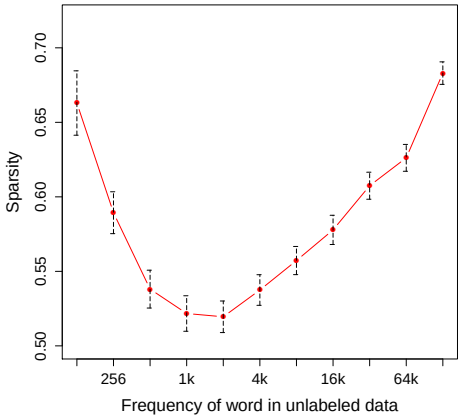


Figure 3: Sparsity (with confidence interval) of the binarized embedding vector w.r.t. word frequency in the unlabeled data.

Figure 2(b) further supports our analysis. BinarizedEmb also reduce much of the errors for the highly frequent words (32k-64k).

As expected, the distributional prototype features produce fewest errors in most cases. The main reason is that the prototype features are *task-specific*. The prototypes are extracted from the training data and contained indicative information of the target labels. By contrast, the other embedding features are simply derived from general word representations and are not specialized for certain tasks, such as NER.

5.3.2 Linear Separability

Another reason for the superiority of the proposed embedding features is that the high-dimensional discrete features are more linear separable than the low-dimensional continuous embeddings. To verify the hypothesis, we further carry out experiments to analyze the linear separability of the proposed discrete embedding features against dense continuous embeddings.

We formalize this problem as a binary classification task, to determine whether a word is an NE or not (NE identification). The linear support vector machine (SVM) is used to build the classifiers, using different embedding features respec-

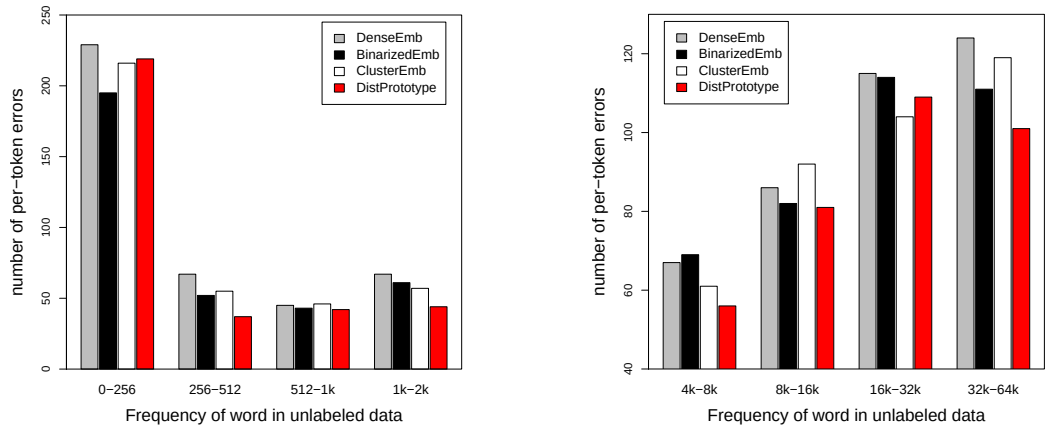


Figure 2: The number of per-token errors w.r.t. word frequency in the unlabeled data. (a) For rare words (frequency $\leq$ 2k). (b) For frequent words (frequency $\geq$ 4k).

Setting	Acc.	#features
DenseEmb	95.46	250
BinarizedEmb	94.10	500
ClusterEmb	**97.57**	482,635
DistPrototype	**96.09**	1,700
DistPrototype-binary	**96.82**	4,530

Table 5: Performance of the NE/non-NE classification on the CoNLL-2003 development dataset using different embedding features.

tively. We use the LIBLINEAR tool (Fan et al., 2008) as our SVM implementation. The penalty parameter C is tuned from 0.1 to 1.0 on the development dataset. The results are shown in Table 5. As we can see, NEs and non-NEs can be better separated using ClusterEmb or DistPrototype features. However, the BinarizedEmb features perform worse than the direct use of word embedding features. The reason might be inferred from the third column of Table 5. As demonstrated in Wang and Manning (2013), linear models are more effective in high-dimensional and discrete feature space. The dimension of the BinarizedEmb features remains small (500), which is merely twice the DenseEmb. By contrast, feature dimensions are much higher for ClusterEmb and DistPrototype, leading to better linear separability and thus can be better utilized by linear models.

We notice that the DistPrototype features perform significantly worse than ClusterEmb in NE identification. As described in Section 3.4, in previous experiments, we automatically extracted prototypes for each label, and propagated the information via distributional similarities. Intuitively, the prototypes we used should be more effective in determining fine-grained NE types than identifying whether a word is an NE. To verify this, we extract new prototypes considering only two labels, namely, NE and non-NE, using the same metric in Section 3.4. As shown in the last row of Table 5, higher performance is achieved.

6 Related Studies

Semi-supervised learning with generalized word representations is a simple and general way of improving supervised NLP systems. One common approach for inducing generalized word representations is to use clustering (e.g., Brown clustering) (Miller et al., 2004; Liang, 2005; Koo et al., 2008; Huang and Yates, 2009).

Aside from word clustering, word embeddings have been widely studied. Bengio et al. (2003) propose a feed-forward neural network based language model (NNLM), which uses an embedding layer to map each word to a dense continuous-valued and low-dimensional vector (parameters), and then use these vectors as the input to predict the probability distribution of the next word. The NNLM can be seen as a joint learning framework for language modeling and word representations.

Alternative models for learning word embeddings are mostly inspired by the feed-forward NNLM, including the Hierarchical Log-Bilinear Model (Mnih and Hinton, 2008), the recurrent neural network language model (Mikolov, 2012), the C&W model (Collobert et al., 2011), the log-linear models such as the CBOW and the Skip-

gram model (Mikolov et al., 2013a; Mikolov et al., 2013b).

Aside from the NNLMs, word embeddings can also be induced using spectral methods, such as latent semantic analysis and canonical correlation analysis (Dhillon et al., 2011). The spectral methods are generally faster but much more memory-consuming than NNLMs.

There has been a plenty of work that exploits word embeddings as features for semi-supervised learning, most of which take the continuous features directly in linear models (Turian et al., 2010; Guo et al., 2014). Yu et al. (2013) propose compound k-means cluster features based on word embeddings. They show that the high-dimensional discrete cluster features can be better utilized by linear models such as CRF. Wu et al. (2013) further apply the cluster features to transition-based dependency parsing.

7 Conclusion and Future Work

This paper revisits the problem of semi-supervised learning with word embeddings. We present three different approaches for a careful comparison and analysis. Using any of the three embedding features, we obtain higher performance than the direct use of continuous embeddings, among which the distributional prototype features perform the best, showing the great potential of word embeddings. Moreover, the combination of the proposed embedding features provides significant additive improvements.

We give detailed analysis about the experimental results. Analysis on rare words and linear separability provides convincing explanations for the performance of the embedding features.

For future work, we are exploring a novel and a theoretically more sounding approach of introducing embedding kernel into the linear models.

Acknowledgments

We are grateful to Mo Yu for the fruitful discussion on the implementation of the cluster-based embedding features. We also thank Ruiji Fu, Meishan Zhang, Sendong Zhao and the anonymous reviewers for their insightful comments and suggestions. This work was supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61133012 and 61370164.

References

Rie Kubota Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 1–9. Association for Computational Linguistics.

Yoshua Bengio, R. E. Jean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3(Feb):1137–1155.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

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

Paramveer S. Dhillon, Dean P. Foster, and Lyle H. Ungar. 2011. Multi-view learning of word embeddings via cca. In *NIPS*, volume 24 of *NIPS*, pages 199–207.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning sense-specific word embeddings by exploiting bilingual resources. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 497–507, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 320–327. Association for Computational Linguistics.

Fei Huang and Alexander Yates. 2009. Distributional representations for handling sparsity in supervised sequence-labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1, pages 495–503.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL), pages 873–882, Jeju Island, Korea. ACL.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In Kathleen McKeown, Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, editors, *Proc. of ACL-08: HLT*, Proc. of ACL-08: HLT, pages 595–603, Columbus, Ohio. ACL.

Vijay Krishnan and Christopher D Manning. 2006. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 1121–1128. Association for Computational Linguistics.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Master thesis, Massachusetts Institute of Technology.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proc. of the NIPS*, Proc. of the NIPS, pages 3111–3119, Nevada. MIT Press.

Tomas Mikolov. 2012. *Statistical Language Models Based on Neural Networks*. Ph. d. thesis, Brno University of Technology.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *HLT-NAACL*, volume 4, pages 337–342.

Andriy Mnih and Geoffrey E. Hinton. 2008. A scalable hierarchical distributed language model. In *Proc. of the NIPS*, Proc. of the NIPS, pages 1081–1088, Vancouver. MIT Press.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL-HLT*, pages 380–390.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 147–155, Stroudsburg, PA, USA. Association for Computational Linguistics.

D Sculley. 2010. Combined regression and ranking. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 979–988. ACM.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In Jan Hajic, Sandra Carberry, and Stephen Clark, editors, *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL), pages 384–394, Uppsala, Sweden. ACL.

Mengqiu Wang and Christopher D. Manning. 2013. Effect of non-linear deep architecture in sequence labeling. In *Proc. of the Sixth International Joint Conference on Natural Language Processing*, Proc. of the Sixth International Joint Conference on Natural Language Processing, pages 1285–1291, Nagoya, Japan. Asian Federation of Natural Language Processing.

Xianchao Wu, Jie Zhou, Yu Sun, Zhanyi Liu, Dianhai Yu, Hua Wu, and Haifeng Wang. 2013. Generalization of words for chinese dependency parsing. *IWPT-2013*, page 73.

Mo Yu, Tiejun Zhao, Daxiang Dong, Hao Tian, and Dianhai Yu. 2013. Compound embedding features for semi-supervised learning. In *Proc. of the NAACL-HLT*, Proc. of the NAACL-HLT, pages 563–568, Atlanta. NAACL.

Combining Punctuation and Disfluency Prediction: An Empirical Study

Xuancong Wang[1,3] **Khe Chai Sim**[2] **Hwee Tou Ng**[1,2]
[1]NUS Graduate School for Integrative Sciences and Engineering
[2]Department of Computer Science, National University of Singapore
[3]Human Language Technology, Institute for Infocomm Research, Singapore
xuancong84@gmail.com, {simkc, nght}@comp.nus.edu.sg

Abstract

Punctuation prediction and disfluency prediction can improve downstream natural language processing tasks such as machine translation and information extraction. Combining the two tasks can potentially improve the efficiency of the overall pipeline system and reduce error propagation. In this work[1], we compare various methods for combining punctuation prediction (PU) and disfluency prediction (DF) on the Switchboard corpus. We compare an isolated prediction approach with a cascade approach, a rescoring approach, and three joint model approaches. For the cascade approach, we show that the soft cascade method is better than the hard cascade method. We also use the cascade models to generate an n-best list, use the bi-directional cascade models to perform rescoring, and compare that with the results of the cascade models. For the joint model approach, we compare mixed-label Linear-chain Conditional Random Field (LCRF), cross-product LCRF and 2-layer Factorial Conditional Random Field (FCRF) with soft-cascade LCRF. Our results show that the various methods linking the two tasks are not significantly different from one another, although they perform better than the isolated prediction method by 0.5–1.5% in the F1 score. Moreover, the clique order of features also shows a marked difference.

1 Introduction

The raw output from automatic speech recognition (ASR) systems does not have sentence boundaries or punctuation symbols. Spontaneous speech also contains a significant proportion of disfluency. Researchers have shown that splitting input sequences into sentences and adding in punctuation symbols improve machine translation (Favre et al., 2008; Lu and Ng, 2010). Moreover, disfluencies in speech also introduce noise in downstream tasks like machine translation and information extraction (Wang et al., 2010). Thus, punctuation prediction (PU) and disfluency prediction (DF) are two important post-processing tasks for automatic speech recognition because they improve not only the readability of ASR output, but also the performance of downstream Natural Language Processing (NLP) tasks.

The task of punctuation prediction is to insert punctuation symbols into conversational speech texts. Punctuation prediction on long, unsegmented texts also achieves the purpose of sentence boundary prediction, because sentence boundaries are identified by sentence-end punctuation symbols: periods, question marks, and exclamation marks. Consider the following example,

How do you feel about the Viet Nam War ? Yeah , I saw that as well .

The question mark splits the sequence into two sentences. This paper deals with this task which is more challenging than that on text that has already been split into sentences.

The task of disfluency prediction is to identify word tokens that are spoken incorrectly due to speech disfluency. There are two main types of disfluencies: filler words and edit words. Filler words mainly include filled pauses (e.g., 'uh', 'um') and discourse markers (e.g., "I mean", "you know"). As they are insertions in spontaneous speech to indicate pauses or mark boundaries in discourse, they do not convey useful content information. Edit words are words that are spoken wrongly and then corrected by the speaker. For example, consider the following utterance:

[1]The research reported in this paper was carried out as part of the PhD thesis research of Xuancong Wang at the NUS Graduate School for Integrated Sciences and Engineering.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 121–130,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Edit Filler Repair

I want a flight ~~*to Boston*~~ *uh I mean* to Denver

The phrase "to Boston" forms the edit region to be replaced by "to Denver". The words "uh I mean" are filler words that serve to cue the listener about the error and subsequent corrections.

The motivation of combining the two tasks can be illustrated by the following two utterances:

~~*I am uh*~~ *I am not going with you .*

I am sorry . I am not going with you .

Notice that the bi-gram *"I am"* is repeated in both sentences. For the first utterance, if punctuation prediction is performed first, it might break the utterance both before and after "uh" so that the second-stage disfluency prediction will treat the whole utterance as three sentences, and thus may not be able to detect any disfluency because each one of the three sentences is legitimate on its own. On the other hand, for the second utterance, if disfluency prediction is performed first, it might mark *"I am sorry"* as disfluent in the first place and remove it before passing into the second-stage punctuation prediction. Therefore, no matter which task is performed first, certain utterances can always cause confusion.

There are many ways to combine the two tasks. For example, we can perform one task first followed by another, which is called the cascade approach. We can also mix the labels, or take the cross-product of the labels, or use joint prediction models. In this paper, we study the mutual influence between the two tasks and compare a variety of common state-of-the-art joint prediction techniques on this joint task.

In Section 2, we briefly introduce previous work on the two tasks. In Section 3, we describe our baseline system which performs punctuation and disfluency prediction separately (i.e., in isolation). In Section 4, we compare the soft cascade approach with the hard cascade approach. We also examine the effect of task order, i.e., performing which task first benefits more. In Section 5, we compare the cascade approach with bi-directional n-best rescoring. In Section 6, we compare the 2-layer Factorial CRF (Sutton et al., 2007) with the cross-product LCRF (Ng and Low, 2004), mixed-label LCRF (Stolcke et al., 1998), the cascade approach, and the baseline isolated prediction. Section 7 gives a summary of our overall findings. Section 8 gives the conclusion.

2 Previous Work

There were many works on punctuation prediction or disfluency prediction as an isolated task. For punctuation prediction, Huang and Zweig (2002) used maximum entropy model; Christensen et al. (2001) used finite state and multi-layer perceptron method; Liu et al. (2005) used conditional random fields; Lu and Ng (2010) proposed using dynamic conditional random fields for joint sentence boundary type and punctuation prediction; Wang et al. (2012) has added prosodic features for the dynamic conditional random field approach and Zhang et al. (2013) used transition-based parsing.

For disfluency prediction, Shriberg et al. (1997) uses purely prosodic features to perform the task. Johnson and Charniak (2004) proposed a TAG-based (Tree-Adjoining Grammar) noisy channel model. Maskey et al. (2006) proposed a phrase-level machine translation approach for this task. Georgila (2009) used integer linear programming (ILP) which can incorporate local and global constraints. Zwarts and Johnson (2011) has investigated the effect of using extra language models as features in the reranking stage. Qian and Liu (2013) proposed using weighted Max-margin Markov Networks (M3N) to balance precision and recall to further improve the F1-score. Wang et al. (2014) proposed a beam-search decoder which integrates M3N and achieved further improvements.

There were also some works that addressed both tasks. Liu et al. (2006) and Baron et al. (1998) carried out sentence unit (SU) and disfluency prediction as separate tasks. The difference between SU prediction and punctuation prediction is only in the non-sentence-end punctuation symbols such as commas. Stolcke et al. (1998) mixed sentence boundary labels with disfluency labels so that they do not predict punctuation on disfluent tokens. Kim (2004) performed joint SU and Interruption Point (IP) prediction, deriving edit and filler word regions from predicted IPs using a rule-based system as a separate step.

In this paper, we treat punctuation prediction and disfluency prediction as a joint prediction task, and compare various state-of-the-art joint prediction methods on this task.

3 The Baseline System

3.1 Experimental Setup

We use the Switchboard corpus (LDC99T42) in our experiment with the same train/develop/test split as (Qian and Liu, 2013) and (Johnson and Charniak, 2004). The corpus statistics are shown in Table 1. Since the proportion of exclamation marks and incomplete SU boundaries is too small, we convert all exclamation marks to periods and remove all incomplete SU boundaries (treat as no punctuation). In the Switchboard corpus, the utterances of each speaker have already been segmented into short sentences when used in (Qian and Liu, 2013; Johnson and Charniak, 2004). In our work, we concatenate the utterances of each speaker to form one long sequence of words for use as the input to punctuation prediction and disfluency prediction. This form of input where, utterances are not pre-segmented into short sentences, better reflects the real-world scenarios and provides a more realistic test setting for punctuation and disfluency prediction. Punctuation prediction also gives rise to sentence segmentation in this setting.

Data set	train	develop	test
# of tokens	1.3M	85.9K	65.5K
# of sentences	173.7K	10.1K	7.9K
# of sequences*	1854	174	134
# of edit words	63.6K	4.7K	3.7K
# of filler words	137.1K	9.6K	7.3K
# of Commas	52.7K	1.8K	2.1K
# of Periods	97.6K	6.5K	4.5K
# of Questions	6.8K	363	407
# of Exclamations	67	4	1
# of Incomplete	189	2	0

Table 1: Corpus statistics for all the experiments. *: each conversation produces two long/sentence-joined sequences, one from each speaker.

Our baseline system uses M3N (Taskar et al., 2004), one M3N for punctuation prediction and the other for disfluency prediction. We use the same set of punctuation and disfluency labels (as shown in Table 2) throughout this paper. To compare the various isolated, cascade, and joint prediction models, we use the same feature templates for both tasks as listed in Table 3. Since some of the feature templates require predicted filler labels and part-of-speech (POS) tags, we have trained a POS tagger and a filler predictor both using CRF (i.e., using the same approach as that in Qian and Liu (2013)). The same predicted POS tags and fillers are used for feature extraction in all the experiments in this paper for a fair comparison. The degradation on disfluency prediction due to the concatenation of utterances of each speaker is shown in Table 4. The pause duration features are extracted by running forced alignment on the corresponding Switchboard speech corpus (LDC97S62).

Task	Label	Meaning
Disfluency prediction	*E*	edit word
	F	filler word
	O	otherwise / fluent
Punctuation prediction	*Comma*	comma
	Period	full-stop
	QMark	question mark
	None	no punctuation

Table 2: Labels for punctuation prediction and disfluency prediction.

3.2 Features

We use the standard NLP features such as $F(w_{-1}w_0$='so that'), i.e., the word tokens at the previous and current node position are 'so' and 'that' respectively. Each feature is associated with a *clique order*. For example, since the clique order of this feature template is 2 (see Table 3), its feature functions can be $f(w_{-1}w_0$='so that', y_0='*F*', y_{-1}='*O*', $t)$. The example has a value of 1 only when the words at node $t-1$ and t are 'so that', and the labels at node t and $t-1$ are *'F'* and *'O'* respectively. The maximum length of the y history is called the *clique order* of the feature (in this feature function, it is 2 since only y_0 and y_{-1} are covered). The feature templates are listed in Table 3. w_i refers to the word at the i^{th} position relative to the current node; *window size* is the maximum span of words centered at the current word that the template covers, e.g., $w_{-1}w_0$ with a window size of 9 means $w_{-4}w_{-3}$, $w_{-3}w_{-2}$, ..., w_3w_4; p_i refers to the POS tag at the i^{th} position relative to the current node; $w_{i\sim j}$ refers to any word from the i^{th} position to the j^{th} position relative to the current node, this template can capture word pairs which can potentially indicate a repair, e.g., "was ... is ...", the speaker may have spoken any

word(s) in between and it is very difficult for the standard n-gram features to capture all possible variations; $w_{i,\neq F}$ refers to the i^{th} non-filler word with respect to the current position, this template can extract n-gram features skipping filler words; the multi-pair comparison function $I(a, b, c, ...)$ indicates whether each pair (a and b, b and c, and so on) is identical, for example, if $a = b \neq c = d$, it will output "101" ('1' for being equal, '0' for being unequal), this feature template can capture consecutive word/POS repetitions which can further improve upon the standard repetition features; and *ngram-score* features are the natural logarithm of the following 8 probabilities: $P(w_{-3}, w_{-2}, w_{-1}, w_0)$, $P(w_0|w_{-3}, w_{-2}, w_{-1})$, $P(w_{-3}, w_{-2}, w_{-1})$, $P(\langle/s\rangle|w_{-3}, w_{-2}, w_{-1})$, $P(w_{-3})$, $P(w_{-2})$, $P(w_{-1})$ and $P(w_0)$ (where "$\langle/s\rangle$" denotes sentence-end).

Feature Template	Window Size	Clique Order
w_0	9	1
$w_{-1}w_0$	9	2
$w_{-2}w_{-1}w_0$	9	2
p_0	9	1
$p_{-1}p_0$	9	2
$p_{-2}p_{-1}p_0$	9	2
$w_0w_{-6\sim-1}$, $w_0w_{1\sim6}$	1	1
$I(w_i, w_j)$	21	2
$I(w_i, w_j, w_{i+1}, w_{j+1})$	21	2
$I(w_i, w_j)(w_i$ if w_i=$w_j)$	21	2
$I(p_i, p_j)$	21	3
$I(p_i, p_j, p_{i+1}, p_{j+1})$	21	3
$I(p_i, p_j)(p_i$ if p_i=$p_j)$	21	3
$p_{-1}w_0$	5	2
$w_{-1}p_0$	5	2
$w_{-2,\neq F}w_{-1,\neq F}$	1	2
$w_{-3,\neq F}w_{-2,\neq F}w_{-1,\neq F}$	1	2
$p_{-2,\neq F}p_{-1,\neq F}$	1	2
$p_{-3,\neq F}p_{-2,\neq F}p_{-1,\neq F}$	1	2
ngram-score features	1	3
pause duration before w_0	1	3
pause duration after w_0	1	3
transitions	1	3

Table 3: Feature templates for disfluency prediction, or punctuation prediction, or joint prediction for all the experiments in this paper.

The performance of the system can be further improved by adding additional prosodic features (Savova and Bachenko, 2003; Shriberg et al., 1998; Christensen et al., 2001) apart from pause durations. However, since in this work we focus on model-level comparison, we do not use other prosodic features for simplicity.

3.3 Evaluation and Results

Experiment	F1 (PU)	F1 (DF)
Short sentences, with precision/recall balancing, clique order of features up to 3, and labels {*E,F,O*}	N.A.	84.7
Short sentences, with precision/recall balancing, clique order of features up to 3, and labels {*E,O*}	N.A.	84.3
Join utterances into long sentences	71.1	79.2
Join utterances into long sentences + remove precision/recall balancing	71.1	78.2
Join utterances into long sentences + remove precision/recall balancing + reduce clique order of all features	68.5	76.4

Table 4: Baseline results showing the degradation by joining utterances into long sentences, removing precision/recall balancing, and reducing the clique order of features. All models are trained using M3N.

We use the standard F1 score as our evaluation metric and this is similar to that in Qian and Liu (2013). For training, we set the frequency pruning threshold to 5 to control the number of parameters. The regularization parameter is tuned on the development set. Since the toolkits used to run different experiments have slightly different limitations, in order to make fair comparisons across different toolkits, we do not use weighting to balance precision and recall when training M3N and we have reduced the clique order of transition features to two and all the other features to one in some of our experiments. Since the performance of filler word prediction on this dataset is already very high, (>97%), we only focus on the F1 score of edit word prediction in this paper when reporting the performance of disfluency prediction. Table 4 shows our baseline results. Our preliminary study shows the following gen-

eral trends: (i) for disfluency prediction: joining utterances into long sentences will cause a 5–6% drop in F1 score; removing precision/recall balance in M3N will cause about 1% drop in F1 score; and reducing the clique order in Table 3 will cause about 1–2% drop in F1 score; and (ii) for punctuation prediction: removing precision/recall balance in M3N will cause negligible drop in F1 score; and reducing clique order will cause about 2–3% drop in F1 score. Conventionally, the degradation from reducing the clique orders can be mostly compensated by using the BIES (Begin, Inside, End, and Single) labeling scheme. In this work, for consistency and comparability across various experiments, we will stick to the same set of labels in Table 2.

4 The Cascade Approach

Instead of decomposing the joint prediction of punctuation and disfluency into two independent tasks, the cascade approach considers one task to be conditionally dependent on the other task such that the predictions are performed in sequence, where the results from the first step is used in the second step. In this paper, we compare two types of cascade: hard cascade versus soft cascade.

4.1 Hard Cascade

For the hard cascade, we use the output from the first step to modify the input sequence *before* extracting features for the second step. For PU→DF (PUnctuation prediction followed by DisFluency prediction), we split the input sequence into sentences according to the sentence-end punctuation symbols predicted by the first step, and then perform the DF prediction on the short/sentence-split sequences in the second step. For DF→PU, we remove the edit and filler words predicted by the first step, and then predict the punctuations using the cleaned-up input sequence. The hard cascade method may be helpful because the disfluency prediction on short/sentence-split sequences is better than on long/sentence-joined sequences (see the second and third rows in Table 4). On the other hand, the punctuation prediction on fluent text is more accurate than that on non-fluent text based on our preliminary study.

For this experiment, four models are trained using M3N without balancing precision/recall. For the first step, two models are trained on long/sentence-joined sequences with disfluent tokens - one for PU prediction and the other for DF prediction. These are simply the isolated baseline systems. For the second step, the DF prediction model is trained on the short/sentence-split sequences with disfluent tokens while the PU prediction model is trained on the long/sentence-joined sequences with disfluent tokens removed. Note that in the second step of DF→PU, punctuation labels are predicted only for the fluent tokens since the disfluent tokens predicted by the first step has already been removed. Therefore, during evaluation, if the first step makes a false positive by predicting a fluent token as an edit or filler, we set its punctuation label to the neutral label, *None*. All the four models are trained using the same feature templates as shown in Table 3. The regularization parameter is tuned on the development set.

4.2 Soft Cascade

For the soft cascade method, we use the labels predicted from the first step as additional features for the second step. For PU→DF, we model the joint probability as:

$$P(\text{DF}, \text{PU}|\mathbf{x}) = P(\text{PU}|\mathbf{x}) \times P(\text{DF}|\text{PU}, \mathbf{x}) \quad (1)$$

Likewise, we model the joint probability for DF→PU as:

$$P(\text{DF}, \text{PU}|\mathbf{x}) = P(\text{DF}|\mathbf{x}) \times P(\text{PU}|\text{DF}, \mathbf{x}) \quad (2)$$

For this experiment, four models are trained using M3N without balancing precision/recall. As with the case of hard cascade, the two models used in the first step are simply the isolated baseline systems. For the second step, in addition to the feature templates in Table 3, we also pass on the labels (at the previous, current and next position) predicted by the first step as three third-order-clique features. We also tune the regularization parameter on the development set to obtain the best model.

4.3 Experimental Results

Table 5 compares the performance of the hard and soft cascade methods with the isolated baseline systems. In addition, we have also included the results of using the true labels in place of the labels predicted by the first step to indicate the upper-bound performance of the cascade approaches. The results show that both the hard and soft cascade methods outperform the baseline systems, with the latter giving a better performance

Experiment	F1 for PU	F1 for DF
isolated baseline	71.1	78.2
hard cascade	71.2	79.1
hard cascade (using true labels)	72.6	83.5
soft cascade	71.6	79.6
soft cascade (using true labels)	72.1	82.7

Table 5: Performance comparison between the hard cascade method and the soft cascade method with respect to the baseline isolated prediction. All models are trained using M3N without balancing precision and recall.

(statistical significance at p=0.01). However, hard cascade has a higher upper-bound than soft cascade. This observation can be explained as follows.

For hard cascade, the input sequence is modified prior to feature extraction. Therefore, many of the features generated by the feature templates given in Table 3 will be affected by these modifications. So, provided that the modifications are based on the correct information, the resulting features will not contain unwanted artefacts caused by the absence of the sentence boundary information for the presence of disfluencies. For example, in *"do you do you feel that it was worthy"*, the punctuation prediction system tends to insert a sentence-end punctuation after the first *"do you"* because the speaker restarts the sentence.

If the disfluency was correctly predicted in the first step, then the hard cascade method would have removed the first *"do you"* and eliminated the confusion. Similarly, in "*I 'm sorry . I 'm not going with you tomorrow .* ", the first "*I 'm*" is likely to be incorrectly detected as disfluent tokens since consecutive repetitions are a strong indication of disfluency. In the case of hard cascade, PU→DF, the input sequence would have been split into sentences and the repetition feature would not be activated. However, since the hard cascade method has a greater influence on the features for the second step, it is also more sensitive to the prediction errors from the first step.

Another observation from Table 5 is that the improvement of the soft cascade over the isolate baseline is much larger on DF (1.4% absolute) than that on PU (only 0.5% absolute). The same holds true for the hard cascade, despite the fact that there are more DF labels than PU labels in this corpus (see Table 1) and the first step prediction is more accurate on DF than on PU. This suggests that their mutual influence is not symmetrical, in the way that the output from punctuation prediction provides more useful information for disfluency prediction than the other way round.

5 The Rescoring Approach

In Section 4, we have described that the two tasks can be cascaded in either order, i.e., PU→DF and DF→PU. However, the performance of the second step greatly depends on that of the first step. In order to reduce sensitivity to the errors made in the first step, one simple approach is to propagate multiple hypotheses from the first step to the second step to obtain a list of joint hypotheses (with both the DF and PU labels). We then rerank these hypotheses based on the joint probability and pick the best. We call this the rescoring approach. From (1) and (2), the joint probabilities can be expressed in terms of the probabilities generated by four models: $P(\text{PU}|\mathbf{x})$, $P(\text{DF}|\text{PU}, \mathbf{x})$, $P(\text{DF}|\mathbf{x})$, and $P(\text{PU}|\text{DF}, \mathbf{x})$. We can combine the four models to form the following joint probability function for rescoring:

$$P(\text{DF}, \text{PU}|\mathbf{x}) = P(\text{DF}|\mathbf{x})^{\alpha_1} \times P(\text{PU}|\text{DF}, \mathbf{x})^{\alpha_2} \times P(\text{PU}|\mathbf{x})^{\beta_1} \times P(\text{DF}|\text{PU}, \mathbf{x})^{\beta_2}$$

where α_1, α_2, β_1, and β_2 are used to weight the relative importance between (1) and (2); and between the first and second steps. In practice, the probabilities are computed in the log domain where the above expression becomes a weighted sum of the log probabilities. A similar rescoring approach using two models is described in Shi and Wang (2007).

The experimental framework is shown in Figure 1. For PU→DF, we first use $P(\text{PU}|\mathbf{x})$ to generate an n-best list. Then, for each hypothesis in the n-best list, we use $P(\text{DF}|\text{PU}, \mathbf{x})$ to obtain another n-best list. So we have n^2-best joint hypotheses. We do the same for DF→PU to obtain another n^2-best joint hypotheses. We rescore the $2n^2$-best list using the four models. The four weights α_1, α_2, β_1, and β_2 are tuned to optimize the overall F1 score on the development set. We used the MERT (minimum-error-rate training, (Och, 2003)) algorithm to tune the weights. We also vary the size of n.

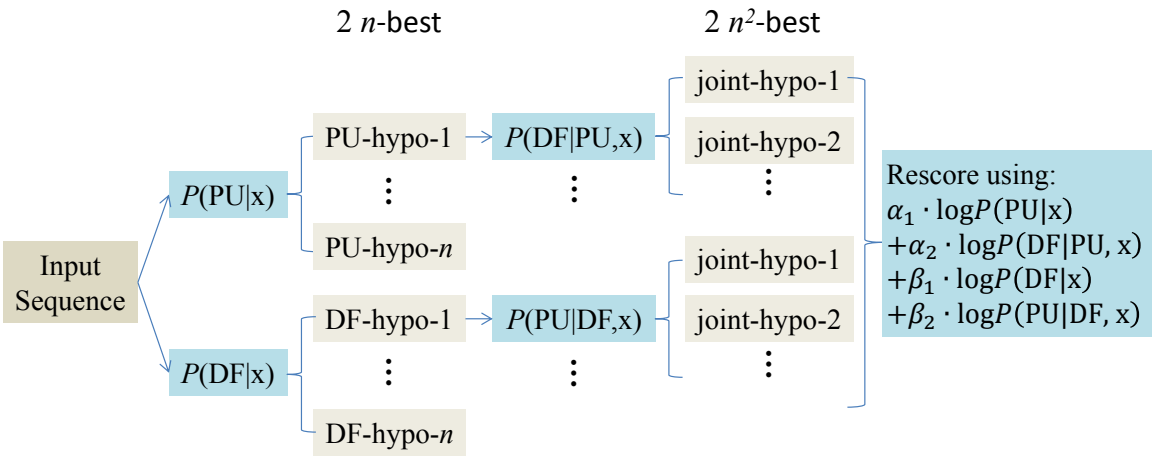


Figure 1: Illustration of the rescoring pipeline framework using the four M3N models used in the soft-cascade method: $P(\text{PU}|\mathbf{x})$, $P(\text{DF}|\text{PU}, \mathbf{x})$, $P(\text{DF}|\mathbf{x})$ and $P(\text{PU}|\text{DF}, \mathbf{x})$

The results shown in Table 6 suggest that the rescoring method does not improve over the soft-cascade baseline. This can be due to the fact that we are using the same four models for the soft-cascade and the rescoring methods. It may be possible that the information contained in the two models for the soft-cascade PU→DF mostly overlaps with the information contained in the other two models for the soft-cascade DF→PU since all the four models are trained using the same features. Thus, no additional information is gained by combining the four models.

6 The Joint Approach

In this section, we compare 2-layer FCRF (Lu and Ng, 2010) with mixed-label LCRF (Stolcke et al., 1998) and cross-product LCRF on the joint prediction task. For the 2-layer FCRF, we use punctuation labels for the first layer and disfluency labels for the second layer (see Table 2). For the mixed-label LCRF, we split the neutral label $\{O\}$ into $\{$*Comma, Period, QMark, None*$\}$ so that we have six labels in total, $\{$*E, F, Comma, Period, QMark, None*$\}$. In this approach, disfluent tokens do not have punctuation labels because in real applications, if we just want to get the cleaned-up/fluent text with punctuations, we do not need to predict punctuations on disfluent tokens as they will be removed during the clean-up process. Since this approach does not predict punctuation labels on disfluent tokens, its punctuation F1 score is only evaluated on those fluent tokens. For the cross-product LCRF, we compose each of the three disfluency labels with the four punctuation labels to get 12 PU-DF-joint labels (Ng and Low, 2004). Figure 2 shows a comparison of these three models in the joint prediction of punctuation and disfluency. All the LCRF and FCRF models are trained using the GRMM toolkit (Sutton, 2006). We use the same feature templates (Table 3) to generate all the features for the toolkit. However, to reduce the training time, we have set clique order to 2 for the transitions and 1 for all other features. We tune the Gaussian prior variance on the development set for all the experiments to obtain the best model for testing.

Table 7 shows the comparison of results. On DF alone, the improvement of the cross-product LCRF over the mixed-label LCRF, and the improvement of the mixed-label LCRF over the isolated baseline are not statistically significant. However, if we test the statistical significance on the overall performance of both PU and DF, both the 2-layer FCRF and the cross-product LCRF perform better than the mixed-label LCRF. And we also obtain the same conclusion as Stolcke et al. (1998) that mixed-label LCRF performs better than isolated prediction. However, for the comparison between the 2-layer FCRF and the cross-product LCRF, although the 2-layer FCRF performs better than the cross-product LCRF on disfluency prediction, it does worse on punctuation prediction. Overall, the two methods perform about the same, their difference is not statistically significant. In addition, both the 2-layer FCRF and the cross-product LCRF slightly outperform the soft cascade method (statistical significance at p=0.04).

Experiment	F1 for PU	F1 for DF
isolated baseline	71.1	78.2
soft-cascade	71.6	79.6
rescore n=1	71.5 (72.5)	79.3 (81.1)
rescore n=2	71.2 (73.0)	79.3 (81.8)
rescore n=3	71.2 (73.3)	79.9 (82.6)
rescore n=4	71.2 (73.6)	79.8 (82.8)
rescore n=5	71.2 (73.9)	79.4 (83.3)
rescore n=6	71.1 (74.0)	79.6 (83.5)
rescore n=8	71.2 (74.2)	79.8 (84.0)
rescore n=10 *	71.2 (74.4)	79.8 (84.3)
rescore n=12	71.1 (74.5)	79.7 (84.6)
rescore n=15	71.2 (74.8)	79.8 (84.9)
rescore n=18	71.1 (74.9)	79.7 (85.1)
rescore n=25	70.7 (75.2)	79.3 (85.5)

Table 6: Performance comparison between the rescoring method and the soft-cascade method with respect to the baseline isolated prediction. The rescoring is done on $2n^2$ hypotheses. All models are trained using M3N without balancing precision and recall. Figures in the bracket are the oracle F1 scores of the $2n^2$ hypotheses. *:on the development set, the best overall result is obtained at $n = 10$.

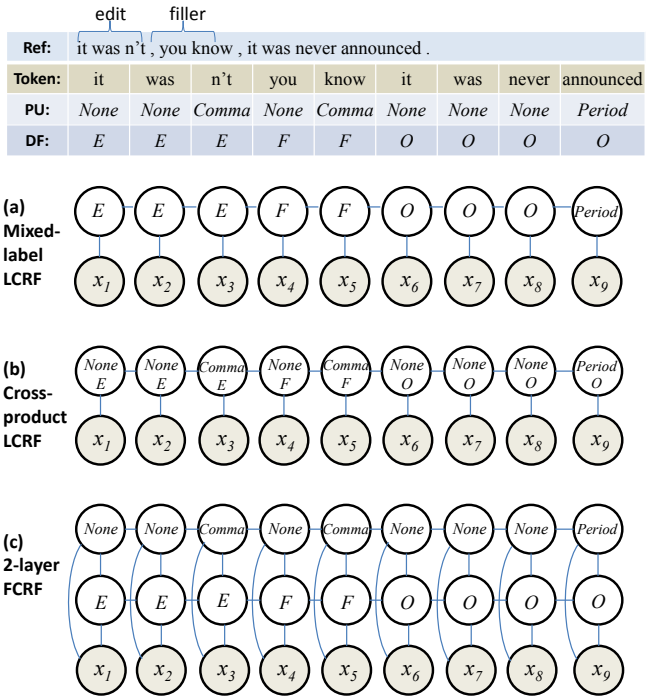


Figure 2: Illustration using (a) mixed-label LCRF; (b) cross-product LCRF; and (c) 2-layer FCRF, for joint punctuation (PU) and disfluency (DF) prediction. Shaded nodes are observations and unshaded nodes are variables to be predicted.

7 Discussion

In this section, we will summarise our observations based on the empirical studies that we have conducted in this paper.

Firstly, punctuation prediction and disfluency prediction do influence each other. The output from one task does provide useful information that can improve the other task. All the approaches studied in this work, which link the two tasks together, perform better than their corresponding isolated prediction baseline.

Experiment	F1 for PU	F1 for DF
isolated baseline	68.7	77.0
soft cascade	69.0	77.5
mixed-label LCRF	69.0	77.2
cross-product LCRF	69.9	77.3
2-layer FCRF	69.2	77.8

Table 7: Performance comparison among 2-layer FCRF, mixed-label LCRF and cross-product LCRF, with respect to the soft-cascade and the isolated prediction baseline. All models are trained using GRMM (Sutton, 2006), with reduced clique orders.

Secondly, as compared to the soft cascade, the hard cascade passes more information from the first step into the second step, and thus is much more sensitive to errors in the first step. In practice, unless the first step has very high accuracy, soft cascade is expected to do better than hard cascade.

Thirdly, if we train a model using a fine-grained label set but test it on the same coarse-grained label set, we are very likely to get improvement. For example:

- The edit word F1 for mixed edit and filler prediction using $\{E, F, O\}$ is better than that for edit prediction using $\{E, O\}$ (see the second and third rows in Table 4). This is because the former actually splits the O in the latter into F and O. Thus, it has a finer label granularity.

- Disfluency prediction using mixed-label LCRF (using label set $\{E, F, Comma, Period, Question, None\}$) performs better than that using isolated LCRF (using label set $\{E, F, O\}$) (see the second and fourth rows in Table 7). This is because the former dis-

tinguishes between different punctuations for fluent tokens and thus has a finer label granularity.

- Both the cross-product LCRF and 2-layer FCRF perform better than mixed-label LCRF because the former two distinguish between different punctuations for edit, filler and fluent tokens while the latter distinguishes between different punctuations only for fluent tokens. Thus, the former has a much finer label granularity.

From the above comparisons, we can see that increasing the label granularity can greatly improve the accuracy of a model. However, this may also increase the model complexity dramatically, especially when higher clique order is used. Although the joint approach (2-layer FCRF and cross-product LCRF) are better than the soft-cascade approach, they cannot be easily scaled up to using higher order cliques, which greatly limits their potential. In practice, the soft cascade approach offers a simpler and more efficient way to achieve a joint prediction of punctuations and disfluencies.

8 Conclusion

In general, punctuation prediction and disfluency prediction can improve downstream NLP tasks. Combining the two tasks can potentially improve the efficiency of the overall framework and minimize error propagation. In this work, we have carried out an empirical study on the various methods for combining the two tasks. Our results show that the various methods linking the two tasks perform better than the isolated prediction. This means that punctuation prediction and disfluency prediction do influence each other, and the prediction outcome in one task can provide useful information that helps to improve the other task. Specifically, we compare the cascade models and the joint prediction models. For the cascade approach, we show that soft cascade is less sensitive to prediction errors in the first step, and thus performs better than hard cascade. For joint model approach, we show that, when clique order of one is used, all the three joint model approaches perform significantly better than the isolated prediction baseline. Moreover, the 2-layer FCRF and the cross-product LCRF perform slightly better than the mix-label LCRF and the soft-cascade approach, suggesting that modelling at a finer label granularity is potentially beneficial. However, the soft cascade approach is more efficient than the joint approach when a higher clique order is used.

Acknowledgments

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

References

Don Baron, Elizabeth Shriberg, and Andreas Stolcke. 2002. Automatic punctuation and disfluency detection in multi-party meetings using prosodic and lexical cues. In *Proc. of ICSLP.*

Heidi Christensen, Yoshihiko Gotoh, and Steve Renals. 2001. Punctuation annotation using statistical prosody models. In *ISCA Tutorial and Research Workshop (ITRW) on Prosody in Speech Recognition and Understanding.*

Benoit Favre, Ralph Grishman, Dustin Hillard, Heng Ji, Dilek Hakkani-Tur, and Mari Ostendorf. 2008. Punctuating speech for information extraction. In *Proc. of ICASSP.*

Kallirroi Georgila. 2009. Using integer linear programming for detecting speech disfluencies. In *Proc. of NAACL.*

Jing Huang and Geoffrey Zweig. 2002. Maximum entropy model for punctuation annotation from speech. In *Proc. of INTERSPEECH.*

Mark Johnson and Eugene Charniak. 2004. A TAG-based noisy-channel model of speech repairs. In *Proc. of ACL.*

Joungbum Kim. 2004. Automatic detection of sentence boundaries, disfluencies, and conversational fillers in spontaneous speech. Master dissertation of University of Washington.

Yang Liu, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. 2005. Using conditional random fields for sentence boundary detection in speech. In *Proc. of ACL.*

Yang Liu, Elizabeth Shriberg, Andreas Stolcke, Dustin Hillard, Mari Ostendorf, and Mary Harper. 2006. Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1526–1540.

Wei Lu and Hwee Tou Ng. 2010. Better punctuation prediction with dynamic conditional random fields. In *Proc. of EMNLP.*

Sameer Maskey, Bowen Zhou, and Yuqing Gao. 2006. A phrase-level machine translation approach for disfluency detection using weighted finite state transducers. In *Proc. of INTERSPEECH.*

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? Word-based or character-based? In *Proc. of EMNLP.*

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL.*

Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In *Proc. of NAACL.*

Guergana Savova, Joan Bachenko. 2003. Prosodic features of four types of disfluencies. In *ISCA Tutorial and Research Workshop on Disfluency in Spontaneous Speech.*

Yanxin Shi and Mengqiu Wang. 2007. A dual-layer CRFs based joint decoding method for cascaded segmentation and labeling tasks. In *Proc. of IJCAI.*

Elizabeth Shriberg, Rebecca Bates and Andreas Stolcke. 1997. A prosody-only decision-tree model for disfluency detection. In *Proc. of Eurospeech.*

Elizabeth Shriberg, Andreas Stolcke, Daniel Jurafsky, Noah Coccaro, Marie Meteer, Rebecca Bates, Paul Taylor, Klaus Ries, Rachel Martin, and Carol Van Ess-Dykema. 1998. Can prosody aid the automatic classification of dialog acts in conversational speech? In *Language and speech 41, no. 3-4: 443-492.*

Andreas Stolcke, Elizabeth Shriberg, Rebecca A. Bates, Mari Ostendorf, Dilek Hakkani, Madelaine Plauche, Gokhan Tur, and Yu Lu. 1998. Automatic detection of sentence boundaries and disfluencies based on recognized words. In *Proc. of ICSLP.*

Charles Sutton. 2006. GRMM: GRaphical Models in Mallet. *http://mallet.cs.umass.edu/grmm/*

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data. In *Journal of Machine Learning Research*, 8: 693–723.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin Markov networks. In *Proc. of NIPS.*

Wen Wang, Gokhan Tur, Jing Zheng, and Necip Fazil Ayan. 2010. Automatic disfluency removal for improving spoken language translation. In *Proc. of ICASSP.*

Xuancong Wang, Hwee Tou Ng, and Khe Chai Sim. 2012. Dynamic conditional random fields for joint sentence boundary and punctuation prediction. In *Proc. of Interspeech.*

Xuancong Wang, Hwee Tou Ng, and Khe Chai Sim. 2014. A beam-search decoder for disfluency detection. In *Proc. of COLING.*

Dongdong Zhang, Shuangzhi Wu, Nan Yang, and Mu Li. 2013. Punctuation prediction with transition-based parsing. In *Proc. of ACL.*

Simon Zwarts and Mark Johnson. 2011. The impact of language models and loss functions on repair disfluency detection. In *Proc. of ACL.*

Submodularity for Data Selection in Statistical Machine Translation

Katrin Kirchhoff
Department of Electrical Engineering
University of Washington
Seattle, WA, USA
kk2@u.washington.edu

Jeff Bilmes
Department of Electrical Engineering
University of Washington
Seattle, WA, USA
bilmes@u.washington.edu

Abstract

We introduce submodular optimization to the problem of training data subset selection for statistical machine translation (SMT). By explicitly formulating data selection as a submodular program, we obtain fast scalable selection algorithms with mathematical performance guarantees, resulting in a unified framework that clarifies existing approaches and also makes both new and many previous approaches easily accessible. We present a new class of submodular functions designed specifically for SMT and evaluate them on two different translation tasks. Our results show that our best submodular method significantly outperforms several baseline methods, including the widely-used cross-entropy based data selection method. In addition, our approach easily scales to large data sets and is applicable to other data selection problems in natural language processing.

1 Introduction

SMT has made significant progress over the last decade, not least due to the availability of increasingly larger data sets. Large-scale SMT systems are now routinely trained on millions of sentences of parallel data, and billions of words of monolingual data for language modeling. Large data sets are often beneficial, but they do create certain other problems. First, they place higher demands on computational resources (storage and compute). Hence, existing software infrastructure may need to be adapted and optimized to handle such large data sets. Second, experimental turn-around time is increased as well, making it more difficult to quickly train, fine-tune, and evaluate novel modeling approaches. Most importantly, however, SMT performance does not increase linearly with the training data size but levels off after a certain point. This is because the additional training data may be noisy, irrelevant to the task at hand, or inherently redundant. Thus, a linear increase in the amount of training data typically leads to a sublinear increase in performance, an effect known as diminishing returns. Several recent papers (Bloodgood and Callison-Burch, 2010; Turchi et al., 2012a; Turchi et al., 2012b) have amply demonstrated this effect.

A way to counteract this is to perform *data subset selection*, i.e., choose a subset of the available training data to optimize a particular quality criterion. One scheme is to select a subset that expresses as much of the information in the original data set as possible - i.e., the data set should be "summarized" by excluding redundant information. Another scheme, popular in the context of SMT, is to subselect the original training set to match the properties of a particular test set.

In this paper, we introduce submodularity for subselecting SMT training data, a methodology that follows both of the above schemes.[1] Submodular functions (Fujishige, 2005) are a class of discrete set functions having the property of diminishing returns. They occur naturally in a wide range of problems in a diverse set of fields including economics, game theory, operations research, circuit theory, and more recently machine learning. Submodular functions share certain properties with convexity (e.g., naturalness and mathematical tractability) although submodularity is still quite distinct from convexity.

We present a novel class of submodular functions particularly suited for SMT subselection and evaluate it against state-of-the-art baseline methods on two different translation tasks, showing that our method outperforms them significantly in most cases. While many approaches to SMT data selection have been developed previously (a detailed overview is provided in Section 3), many of them are heuristic and do not offer performance guarantees. Certain previous approaches, however, have

[1] As far as we know, submodularity has not before been explicitly utilized for SMT subset selection.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 131–141,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

inadvertently made use of submodular methods. This, in addition to our own positive results, provides strong evidence that submodularity is a natural and practical framework for data subset selection in SMT and related fields.

An additional advantage of this framework is that many submodular programs (e.g., the greedy procedure reviewed in Section 2) are fast and scalable to large data sets. By contrast, trying to solve a submodular problem using, say, an integer-linear programming (ILP) procedure, would lead to impenetrable scalability problems.

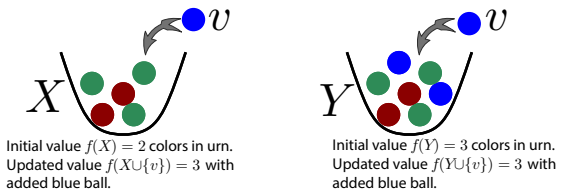


Figure 1: $f(Y)$ measures the number of distinct colors in the set of balls Y, and hence is submodular.

This paper makes several contributions: First, we present a brief overview of submodular functions (Section 2) and their potential application to natural language processing (NLP). Next we review previous approaches to MT data selection (Section 3) and analyze them with respect to their submodular properties. We find that some previous approaches are submodular in nature although this connection was not heretofore made explicit. Section 4 details our new approach. We discuss desirable properties of an SMT data selection objective and present a new class of submodular functions tailored towards this problem. Section 5 presents the data and systems used for the experiments, and results are reported in Section 6. Section 7 then concludes.

2 Submodular Functions/Optimization

Submodular functions (Edmonds, 1970; Fujishige, 2005), are widely used in mathematics, economics, circuit theory (Narayanan, 1997), and operations research. More recently, they have attracted much interest in machine learning (e.g., (Narasimhan and Bilmes, 2004; Kolmogorov and Zabih, 2004; Krause et al., 2008; Krause and Guestrin, 2011; Jegelka and Bilmes, 2011; Iyer and Bilmes, 2013)), where they have been applied to a variety of problems. In natural language and speech processing, they have been applied to document summarization (Lin and Bilmes, 2011; Lin and Bilmes, 2012) and speech data selection (Wei et al., 2013).

We are given a finite size-n set of objects V (i.e., $|V| = n$). A valuation function $f : 2^V \rightarrow \mathbb{R}_+$ is defined that returns a non-negative real value for any subset $X \subseteq V$. The function f is said to be *submodular* if it satisfies the property of diminishing returns: namely, for all $X \subseteq Y$ and $v \notin Y$, we must have:

$$f(X \cup \{v\}) - f(X) \geq f(Y \cup \{v\}) - f(Y). \quad (1)$$

This means that the incremental value (or gain) of element v decreases when the context in which v is considered grows from X to $Y \supseteq X$. We define the "gain" as $f(v|X) \triangleq f(X \cup \{v\}) - f(X)$. Hence, f is submodular if $f(v|X) \geq f(v|Y)$. We note that a function $m : 2^V \rightarrow \mathbb{R}_+$ is said to be *modular* if it satisfies the above with equality, meaning $m(v|X) = m(v|Y)$ for all $X \subseteq Y \subseteq V \setminus \{v\}$. If m is modular and $m(\emptyset) = 0$, it can be written as $m(X) = \sum_{x \in X} m(x)$ and, moreover, is seen simply as a n-dimensional vector $m \in \mathbb{R}^V$.

As an example, suppose we have a set V of balls and $f(X)$ counts the number of colors present in any subset $X \subseteq V$. In Figure 1, $|X| = 5$ and $f(X) = 2$, $|Y| = 7$ and $f(Y) = 3$, and $X \subset Y$. Adding v (a blue ball) to X has a unity gain $f(v|X) = 1$ but since a blue ball exists in Y, we have $f(v|Y) = 0 < f(v|X) = 1$.

Submodularity is a natural model for data subset selection in SMT. In this case, each $v \in V$ is a distinct training data sentence and V corresponds to a training set. An important characteristic of any good model for this problem is that we wish to decrease the "value" of a sentence $v \in V$ based on how much that sentence has in common with those sentences, say X, that have already been chosen. The value $f(v|X)$ of a given sentence v in a context of previously chosen sentences $X \subseteq V$ further diminishes as the context grows $Y \supseteq X$. When, for example, a sentence's value is represented as the value of its set of features (e.g., n-grams), it is natural for those features' values to be discounted based on how much representation of those features already exists in a previously chosen subset. This corresponds to submodularity, which can easily be expressed mathematically by functions such as Eqn. (4) below.

Not only are submodular functions natural for SMT subset selection, they can also be optimized efficiently and scalably such that the result has mathematical performance guarantees. In the remainder of this paper we will assume that f is not only submodular, but also non-negative ($f(X) \geq 0$ for all X), and *monotone non-decreasing* ($f(X) \leq f(Y)$ for all $X \subseteq Y$). Such functions are trivial to uselessly maximize, since $f(V)$ is the largest possible valuation. Typically, however, we wish to have

Algorithm 1: The Greedy Algorithm

Input: Submodular function $f : 2^V \rightarrow \mathbb{R}_+$, cost vector m, budget b, finite set V.
Output: X_k where k is the number of iterations.
Set $X_0 \leftarrow \emptyset$; $i \leftarrow 0$;
while $m(X_i) < b$ **do**
    Choose v_i as follows:
    $v_i \in \left\{ \text{argmax}_{v \in V \setminus X_i} \frac{f(\{v\}|X_i)}{m(v)} \right\}$;
    $X_{i+1} \leftarrow X_i \cup \{v_i\}$; $i \leftarrow i+1$;

a valuable subset of bounded and small cost, where cost is measured based on a modular function $m(X)$. For example, the cost $m(v)$ of a sentence $v \in V$ might be its length, so $m(X) = \sum_{x \in X} m(x)$ is a sum of sentence lengths. This leads to the following optimization problem:

$$X^* \in \underset{X \subseteq V, m(X) \leq b}{\text{argmax}} \; f(X), \tag{2}$$

where b is a known budget. Solving this problem exactly is NP-complete (Feige, 1998), and expressing it as an ILP procedure renders it impractical for large data sizes. When f is submodular the cost is just size ($m(X) = |X|$), then the simple greedy algorithm (detailed below) will have a **worst-case guarantee** of $f(\tilde{X}^*) \geq (1 - 1/e) f(X_{\text{opt}}) \approx 0.63 f(X_{\text{opt}})$ where X_{opt} is the optimal and $\tilde{X}^*$ is the greedy solution (Nemhauser et al., 1978).

This constant factor guarantee has practical importance. First, a constant factor guarantee stays the same as n grows, so the relative worst-case quality of the solution is the same for small and for big problem instances. Second, the worst-case result is achieved only by very contrived and unrealistic function instances — the typical case is almost always much better. Third, the worst-case guarantee improves depending on the "curvature" $\kappa \in [0,1]$ of the submodular function (Conforti and Cornuejols, 1984). When the submodular function is not fully curved ($\kappa < 1$, something true of the functions used in this paper), the worst case guarantee is better, namely $\frac{1}{\kappa}(1 - e^{-\kappa})$ (e.g., a function f with $\kappa = 0.2$ has a worst-case guarantee of 0.91). Lastly, when the cost m is not just cardinality but an arbitrary non-negative modular function, a greedy algorithm has similar guarantees (Sviridenko, 2004), and a scalable variant has a worst-case guarantee of $1 - 1/\sqrt{e}$ (Lin and Bilmes, 2010).

The basic greedy algorithm has a very simple form. Starting with $X \leftarrow \emptyset$, we repeat the operation $X \leftarrow X \cup \text{argmax}_{v \in V \setminus X} \frac{f(v|X)}{m(v)}$ until the budget is exceeded ($m(X) > b$) and then backoff to the previous iteration (complete details are given in Algorithm 1). While the algorithm has complexity $O(n^2)$, there is an accelerated instance of this algorithm (Minoux, 1978; Leskovec et al., 2007) that has empirical computational complexity of $O(n \log n)$ where $n = |V|$. The greedy algorithm, therefore, scales practically to very large n. Recently, still much faster (Wei et al., 2014) and also parallel distributed (Mirzasoleiman et al., 2013) greedy procedures have been advanced offering still better scalability.

There are many submodular functions that are appropriate for subset selection (Lin and Bilmes, 2011; Lin and Bilmes, 2012). Some of them are graph-based, where we are given a non-negative weighted graph $G = (V, E, w)$ and $w : E \rightarrow \mathbb{R}_+$ is a set of edge weights (i.e., $w(x,y)$ is a non-negative similarity score between sentences x and y). A submodular function is obtained via a *graph cut* function $f(X) = \sum_{x \in X, y \in V \setminus X} w(x,y)$ or via a *monotone truncated graph cut* function $f(X) = \sum_{v \in V} \min(C_v(X), \alpha C_v(V))$ where $\alpha \in (0,1)$ is a scalar parameter and $C_v(X) = \sum_{x \in X} w(v,x)$ is a v-specific modular function. Alternatively, the class of *facility location functions* $f(X) = \sum_{v \in V} \max_{x \in X} w(x,v)$ have been widely and successfully used in the field of operations research, and are also applicable here.

In the worse case, the required graph construction has a worst-case complexity of $O(n^2)$. While sparse graphs can be used, this can be prohibitive when $n = |V|$ gets large. Another class of submodular functions that does not have this problem is based on a weighted *bipartite* graph $G = (V, U, E, w)$ where V are the left vertices, U are the right vertices, $E \subseteq V \times U$ is a set of edges, and $w : U \rightarrow \mathbb{R}_+$ is a set of non-negative weights on the vertices U. For $X \subseteq V$, the *bipartite neighborhood* function is defined as:

$$f(X) = w(\{u \in U : \exists x \in X \text{ with } (x,u) \in E\}) \tag{3}$$

This function is interesting for NLP applications since U can be seen as a set of "features" of the elements $v \in V$ (i.e., if V is a set of sentences, U can be the collective set of n-grams for multiple values of n, and $f(X)$ is the weight of the n-grams contained collectively in sentences X).[2] Given a set $X \subseteq V$,

[2] To be consistent with standard notation in previous literature, we overload the use of n in "n-grams" and the size of our set "$n = |V|$", even though the two ns have no relationship with each other.

we get value from the features of the elements $x \in X$, but we get credit for each feature only one time — once a given object $x \in X$ has a given feature $u \in U$, any additions to X by elements also having feature u offer no further credit via that feature.

Another interesting class of submodular functions, allowing additional credit from an element even when its features already exist in X, are what we call *feature-based* submodular functions. They involve sums of non-decreasing concave functions applied to modular functions (Stobbe and Krause, 2010) and take the following form:

$$f(X) = \sum_{u \in U} w_u \phi_u(m_u(X)) \quad (4)$$

where $w_u > 0$ is a *feature weight*, $m_u(X) = \sum_{x \in X} m_u(x)$ is a non-negative modular function specific to feature u, $m_u(x)$ is a *relevance score* (a non-negative scalar score indicating the relevance of feature u in object x), and ϕ_u is a u-specific non-negative non-decreasing *concave function*. The gain is $f(v|X) = \sum_{u \in U} \Big(\phi(m_u(X \cup \{v\})) - \phi(m_u(X)) \Big)$, and thanks to ϕ_u's concavity, the term $\phi(m_u(X \cup \{v\})) - \phi(m_u(X))$ for each feature $u \in U$ is decaying as X grows. The rate of decay, and hence the degree of diminishing returns and ultimately the measure of redundancy of the information provided by the feature, is controlled by the concave function. The rate of decay is also related to the curvature κ of the submodular function (c.f. §2), with more aggressive decay having higher curvature (and a worse worst-case guarantee). The decay is a modeling choice that should be decided based on a given application.

Feature-based functions have the advantage that they do not require the construction of a pairwise graph; they have a cost of only $O(n|U|)$, which is **linear in the data size** and therefore scalable to large data set sizes.

We utilize this class for our subset selection experiments described in Section 4, where we use one global concave function $\phi_u = \phi$ for all $u \in U$. In this work we chose one particular set of features U. However, given the large body of research into NLP feature engineering (Jurafsky and Martin, 2009), this class is extensible beyond just this set, which makes it suitable for many other NLP applications.

Before describing our SMT-specific functions in detail, we review previous work on subset selection for SMT in the context of submodularity.

3 Previous Approaches

There have been many previous approaches to data subset selection in SMT. In this section, we show that some of them in fact correspond to submodular methods, thus introducing a connection between submodularity and the practical problem of SMT data selection. The fact that submodularity is implicitly and unintentionally used in previous work suggests that it is natural for this problem.

A currently widely-used data selection method in SMT (which we also use as a baseline in Section 6) uses the cross-entropy between two language models (Moore and Lewis, 2010), one trained on the test set of interest, and another trained on a large set of generic or out-of-domain training data. We call this the *cross-entropy method*. This method trains a test-set specific (or in-domain) language model, LM_{in}, and a generic (out-of- or mixed-domain) language model, LM_{out}. Each sentence $x \in V$ in the training data is given a probability score with both language models and then ranked in descending order based on the log ratio

$$m_{\text{ce}}(x) = \frac{1}{\ell(x)} \log[\Pr(x|\text{LM}_{\text{in}})/\Pr(x|\text{LM}_{\text{out}})] \quad (5)$$

where $\ell(x)$ is the length of sentence x. Finally, the top N sentences are chosen. In (Axelrod et al., 2011) this method is extended to take both sides of the parallel corpus into account rather than just the source side. The cross-entropy approach values each sentence individually, without regard to any interaction with already selected sentences. This approach, therefore, is modular (a special case of submodular) and values a set X via $m(X) = \sum_{x \in X} m(x)$. Moreover, the thresholding method for choosing a subset corresponds exactly to the optimization problem in Eqn. (2) where $f \leftarrow m$ and the budget b is set to the sum of the top N sentence scores. Thanks to modularity, the problem is no longer NP-complete, and the threshold method solves Eqn. (2) exactly. On the other hand, a modular function does not have the diminishing returns property, and thus has no chance to represent interaction or redundancy between sentences. The chosen subset, therefore, might have an enormous overrepresentation of one aspect of the training data while having minimal or no representation of another aspect, a major vulnerability of this approach.

Other methods use information retrieval (Hildebrand et al., 2005; Lü et al., 2007) which can also be described as modular function optimization (e.g., take the top k scoring sentences). Duplicate

sentence removal is easily represented by a feature-based submodular function, Equation (4), where there is one sentence-specific feature per sentence and where $\phi_u(m_u(X)) = \min(|X \cap \{u\}|, 1)$ — once a sentence is chosen, its contribution is saturated so any duplicate sentence has a gain of zero. Also, the unseen n-gram function of (Eck et al., 2005; Bloodgood and Callison-Burch, 2010) corresponds to a *bipartite neighborhood* submodular function, with a weight function defined based on n-gram counts. Moreover their functions are optimized using the greedy algorithm; hence they in fact have a $1-1/e$ guarantee. Other methods have noted and dealt with the existence of redundancy in phrase-based systems (Ittycheriah and Roukos, 2007) by limiting the set of phrases — submodular optimization inherently removes redundancy. Also, (Callison-Burch et al., 2005; Lopez, 2007) involve modular functions but where selection is over subsets of phrases (rather than sentences as in our current work) and where multiple selections occur, each specific to an individual test set sentence rather than the entire test set.

In the feature-decay method, presented in (Biçici, 2011; Biçici and Yuret, 2011; Biçici, 2013), the value of a sentence is based on its decomposition into a set of feature values. As sentences are added to a set, the feature decay approach in general diminishes the value of each feature depending on how much of that feature has already been covered by those sentences previously chosen — the papers define a set of *feature decay functions* for this purpose.

Our analysis of (Biçici, 2011; Biçici and Yuret, 2011; Biçici, 2013), from the perspective of submodularity, has revealed an interesting connection. The feature decay functions used in these papers turn out to be derivatives of non-decreasing concave functions. For example, in one case $\phi'(a) = 1/(1+a)$ which is the derivative of the concave function $\phi(a) = \ln(1+a)$. We are given a constant initialization w_u for feature $u \in U$ — in the papers, they set either $w_u \leftarrow 1$, or $w_u \leftarrow \log(m(V)/m_u(V))$, or $w_u \leftarrow \log(m(V)/(1+m_u(V)))$, where $m(V) = \sum_u m_u(V)$, and where $m_u(X) = \sum_{x \in X} m_u(x)$ is the count of feature u within the set of sentences $X \subseteq V$. This yields the submodular feature function $f_u(X) = w_u \phi(m_u(X))$. The value of sentence v as measured by feature u in the context of X is the gain $f_u(v|X)$, which is a discrete derivative corresponding to $w_u/(1+m_u(X \cup \{v\}))$. An alternative decay function they define is given as $\phi'(a) = 1/(1+b^a)$ for a base b (they set $b \leftarrow 2$) which is the derivative of the following non-decreasing concave function:

$$\phi(a) = \left[1 - \frac{1}{\ln(b)} \ln\Big(1 + \exp\big(-a \ln(b)\big)\Big)\right] \quad (6)$$

We note that this function is saturating, meaning that it quickly reaches its asymptote at its maximum possible value. We can, once again, define a function specific for feature $u \in U$ as $f_u(X) = w_u \phi(m_u(X))$ with a gain $f_u(v|X)$ being a discrete derivative corresponding to $w_u/(1+b^{m_u(X \cup \{v\})})$.

The connection between this work and submodularity is not complete, however, without considering the method used for optimization. In fact, Algorithm 1 of (Biçici and Yuret, 2011) is precisely the accelerated greedy algorithm of (Minoux, 1978) applied to the submodular function corresponding to $f(X) = \sum_{u \in U} f_u(X)$, and Algorithm 1 of (Biçici, 2013) is the cost-normalized variant of this greedy algorithm corresponding to a knapsack constraint (Sviridenko, 2004). Thus, our analysis shows that these methods also have a $1-1/e$ performance guarantee and also the $O(n \log n)$ empirical complexity mentioned in Section 2. This is an important connection, as it furthers the evidence that submodularity is natural for the problem of SMT subset selection. This also increases the accessibility of this method since we may view it as a special case of Equation (4).

Another class of approaches focuses on active learning. In (Haffari et al., 2009) a large corpus of noisy parallel data is created automatically; a smaller set of samples is then selected from this set that receive human translations. A combination of several "informativeness" scores is computed on a sentence-level basis, and samples are selected via hierarchical adaptive sampling (Dasgupta and Hsu, 2008). In (Mandal et al., 2008) a measure of disagreement between different MT systems, as well as an entropy-based criterion are used to select additional data for annotation. In (Bloodgood and Callison-Burch, 2010) and (Ambati et al., 2010), active learning is combined with crowd-sourced annotations to produce large, human-translated data sets that are as informative as possible. In (Cao and Khudanpur, 2012), samples are selected for discriminative training of an MT system according to a greedy algorithm that tries to maximize overall quality. These methods address a different scenario (data selection for annotation or discriminative training) than the one considered here; however, we also note that the actual selection techniques employed in these papers do not appear to be submodular.

4 Novel Submodular Functions for SMT

In this section, we design a parameterized class of submodular functions useful for SMT training data subset selection. By staying within the realm of submodularity, we retain the advantages of the greedy algorithm, its theoretical performance assurances, and its scalability properties. At the same time this opens the door to a general framework for quickly exploring a much larger class of functions (with the same desirable properties) than before.

It is important to note that we are using submodularity as a "model" of the selection process, and the submodular objective acts as a surrogate for the actual SMT objective function. Thus, the mathematical guarantee we have is in terms of the surrogate objective rather than the true SMT objective. Evaluating one point of the actual SMT objective would require the complete training and testing of an SMT system, so even an algorithm as efficient as Algorithm 1 would be infeasible, even on small data. It is therefore important to design a natural and scalable surrogate objective.

We do not consider the graph-based functions discussed in Section 2 here since they require a pairwise similarity matrix over all training sentences and thus have $O(n^2)$ worst-case complexity. For large tasks with millions or even billions of sentences, this eventually becomes impractical. Instead we focus on feature-based functions of the type presented in Eqn. (4), where each sentence is represented as a set of features rather than as a vertex in a graph. In this function there are four components to specify: 1) U, the *linguistic feature set*; 2) $m_u(x)$, the *relevance scores* for each feature u and sentence x; 3) w_u, the *feature weights*; and 4) ϕ, the *concave function* (we use one concave function, so $\phi_u = \phi$ for all $u \in U$).

Feature set: U is the set of n-grams from either the source language U^{src}, or from both the source and target language $U^{\text{src}} \cup U^{\text{tgt}}$ (see Section 6); since we are interested in selecting a training set that matches a given test set, we use the set of n-grams that occur both in the training set and in the development/test data (for target features, only development set features are used). I.e., $U^{\text{src}} = (U^{\text{src}}_{\text{dev}} \cup U^{\text{src}}_{\text{test}}) \cap U^{\text{src}}_{\text{train}}$ and $U^{\text{tgt}} = U^{\text{tgt}}_{\text{dev}} \cap U^{\text{tgt}}_{\text{train}}$.

Relevance scores: A feature u within a sentence x should be valued based on how salient that feature is within the "document" in which it occurs; here, the "document" is the set of training sentences. This is a task well suited to TFIDF. As an alternative to raw feature counts we thus also consider scores of the form $m_u(x) \leftarrow \text{tfidf}(u,x) = \text{tf}(u,x) \times \text{idf}^{\text{trn}}(u)$, where $\text{tf}(u,x)$ and $\text{idf}^{\text{trn}}(u)$ are defined as usual.

Feature weights: We wish to select those training samples that contain features occurring frequently in the test data while avoiding the over-selection of features that are very frequent in the training data because those are likely to be translated correctly anyway. This is similar to the approach in (Moore and Lewis, 2010) (see Equation (5)), where a log-probability ratio of in-domain to out-of-domain language model is utilized. In the present case, we need a value that is specific to feature $u \in U$; a natural approach is to use the ratio of counts $c^{\text{tst}}(u)/c^{\text{trn}}(u)$ where $c^{\text{tst}}(u)$ is the raw count of u in the development/test data, and $c^{\text{trn}}(u)$ is its raw count in the training data (note that $c^{\text{trn}}(u)$ is never zero due to the way U is defined). As an additional factor we allow feature length to have an influence. In general, longer n-grams might be considered more valuable since they typically lead to better translations and are more relevant for BLEU. Thus, we include a reward term for longer n-grams in the form of $\beta^{|u|}$ where $\beta \geq 1$ and $|u|$ is the length of feature u. This gives greater weight to longer n-grams when $\beta > 1$.

Concave function: It is imperative to find the right form of concave function since, as described in Section 2, the concave shape determines the degree to which redundancy and diminishing returns are represented. Intuitively, when the shape of the concave function for a feature becomes "flat" rapidly, that feature quickly looses its ability to provide additional value to a candidate subset. Many different concave functions were tested for ϕ, including one of the two functions implicit in (Biçici and Yuret, 2011) and derived in Section 3, and a variety of roots of the form $\phi(a) = a^{\alpha}$ for $0 < \alpha < 1$. In Table 2, for example, we find evidence that the simple square root $\phi(a) = \sqrt{a}$ performs slightly better than the log function. The square root is much less curved and decays much more gradually than either of the two functions implicit in (Biçici and Yuret, 2011), of which one is a log form and the other is even more curved and quickly saturates (see §3). The square root function yields a less curved submodular function, in the sense of (Conforti and Cornuejols, 1984), resulting in better worst-case guarantees. Indeed, Table 1 in (Biçici and Yuret, 2011) corroborates by showing that the more curved saturating function does worse than the less curved log function.

Four Components Together: Different instantia-

tions of the four components discussed above will result in different submodular functions of the general class defined in Eqn. (4). Particular settings of these general parameters produce the methods considered in (Biçici and Yuret, 2011), thus making that approach easily accessible once the general submodular framework is set up. As a very special case, this is also true of the cross-entropy method (Moore and Lewis, 2010), where $|U| = 1$, $m_u(x) \leftarrow \exp(m_{\text{ce}}(x))$ of Equation (5) [3], $w_u \leftarrow 1$, and $\phi(a) = a$ is the identity function. In Section 6, we specify the parameter settings used in our experiments.

Task	Train	Dev	Test	LM
NIST	189M	48k	49k	2.5B
Europarl	52.8M	57.7k	58.1k	53M

Table 1: Data set sizes (number of source-side words) for MT tasks. LM = language model data.

5 Data and Systems

We evaluate our approach on the NIST Arabic-English translation task, using the NIST 2006 set for development and the NIST 2009 set for evaluation. The training data consists of all Modern Standard Arabic-English parallel LDC corpora permitted in the NIST evaluations (minus the restricted time periods). Together these sets form a mixed-domain training set containing relevant in-domain data similar to the NIST data sets but also less relevant data (e.g., the UN parallel corpora); we thus expect data selection to work well on this task. Additional English language modeling data was drawn from several other LDC corpora (English Gigaword, AQUAINT, HARD, ANC/DCI and the American National Corpus). Preprocessing included conversion of the Arabic data to Buckwalter format, tokenization, spelling normalization, and morphological segmentation using MADA (Habash et al., 2009). Numbers and URLs were replaced with variables. The English data was tokenized and lowercased. Postprocessing involved recasing the translation output, replacing variable names with their original corresponding tokens, and normalizing spelling and stray punctuation marks. The recasing model is an SMT system without reordering, trained on parallel cased and lowercased versions of the training data. The recasing model remains fixed for all experiments and is not retrained for different sizes of the training data. Evaluation follows the NIST guidelines and was done by computing BLEU scores using the official NIST evaluation tool mteval-v13a.pl with the $-c$ flag for case-sensitive scoring. In addition to the NIST task we also applied our method to the Europarl German-English translation task. The training data comes from the Europarl-v7 collection[4]; the development set is the 2006 dev set, and the test set is the 2007 test set. The number of reference translations is 1. The German data was preprocessed by tokenization, lower-casing, splitting noun compounds and lemmatization to address morphological variation in German. The English side was tokenized and lowercased. Evaluation was done by computing BLEU on the lowercased versions of the data. Since test and training data for this task come from largely the same domain we expect the training data to be less redundant or irrelevant; nevertheless it will be interesting to see how much different data selection methods can contribute even to in-domain translation tasks. The sizes of the various data sets are shown in Table 1.

All translation systems were trained using the GIZA++/Moses infrastructure (Koehn et al., 2007). The translation model is a standard phrase-based model with a maximum phrase length of 7. Since a large number of experiments had to be run for this study, more complex hierarchical or syntax-based translation models were deliberately excluded in order to limit the experimental turn-around time needed for each experiment. The reordering model is a hierarchical model according to (Galley and Manning, 2008). The feature weights in the log-linear function were optimized on the development set BLEU score using minimum error-rate training. The language models for the NIST task (5-grams) were trained on three different data sources (Gigaword, GALE data, and all remaining corpora), which were then interpolated into a single model. The interpolation weights were optimized separately for the two different genres present in the NIST task (newswire and web text). All models used Witten-Bell discounting and interpolation of higher-order and lower-order models. Language models remain fixed for all experiments, i.e., the language model training data is not subselected since we were interested in the effect of data subset selection on the translation model only. The language model for the Europarl system was a 5-gram trained on Europarl data only.

[3] Due to modularity, any monotone increasing transformation from $m_{\text{ce}}(x)$ to $m_u(x)$ that ensures $m_u(x) \geq 0$ is equivalent.

[4] http://http://www.statmt.org/europarl/

Method	Data Subset Sizes			
	10%	20%	30%	40%
Rand	0.3991 (± 0.004)	0.4142 (± 0.003)	0.4205 (± 0.002)	0.4220 (± 0.002)
Xent	0.4235 (± 0.004)	0.4292 (± 0.002)	0.4290 (± 0.003)	0.4292 (± 0.001)
SM-1	**0.4309** (± 0.000)	**0.4367** (± 0.001)	0.4330 (± 0.004)	**0.4351** (± 0.002)
SM-2	**0.4330*** (± 0.001)	**0.4395*** (± 0.003)	0.4333 (± 0.001)	**0.4366*** (± 0.003)
SM-3	**0.4313*** (± 0.002)	**0.4338** (± 0.002)	**0.4361*** (± 0.002)	**0.4351** (± 0.003)
SM-4	0.4276 (± 0.003)	0.4303 (± 0.002)	0.4324 (± 0.002)	0.4329 (± 0.000)
SM-5	0.4285 (± 0.004)	**0.4356** (± 0.002)	0.4333 (± 0.003)	**0.4324** (± 0.002)
SM-6	**0.4302*** (± 0.004)	**0.4334** (± 0.003)	**0.4371*** (± 0.002)	**0.4349** (± 0.003)
SM-7	**0.4295** (± 0.002)	**0.4374** (± 0.002)	**0.4344** (± 0.001)	0.4314 (± 0.0004)
SM-8	**0.4304*** (± 0.002)	**0.4323** (± 0.000)	**0.4358** (± 0.003)	0.4337 (± 0.001)
100%	0.4257			

Table 2: BLEU scores (standard deviations) on the NIST 2009 (Ara-En) test set for random (Rand), cross-entropy (Xent), and submodular (SM) data selection methods defined in Table 4. 100% = system using all of the training data. Boldface numbers indicate a statistically significant improvement ($p \leq 0.05$) over the median Xent system. Starred scores are also significantly better than SM-5.

Method	Data Subset Sizes			
	10%	20%	30%	40%
Rand	0.2590 (± 0.003)	0.2652 (± 0.001)	0.2677 (± 0.002)	0.2697 (± 0.001)
Xent	0.2639 (± 0.002)	0.2687 (± 0.002)	0.2704 (± 0.001)	0.2723 (± 0.001)
SM-5	0.2653 (± 0.001)	**0.2727** (± 0.000)	0.2697 (± 0.002)	0.2720 (± 0.002)
SM-6	**0.2697**$*$ (± 0.001)	**0.2700** (± 0.002)	**0.2740**$*$ (± 0.002)	0.2723 (± 0.000)
100%	0.2651			

Table 3: BLEU scores (standard deviation) on the Europarl translation task for random (Rand), cross-entropy (Xent), and submodular (SM) data selection methods. 100% = system using all of the training data. Boldface numbers indicate a statistically significant improvement ($p \leq 0.05$) over the median Xent system. Starred scores are significantly better than SM-5.

6 Experiments

	Function parameters			
	$w(u)$	$\phi(a)$	$m_u(x)$	U
SM-1	$\frac{c^{\text{tst}}(u)}{c^{\text{trn}}(u)}\beta^{\lvert u\rvert}$	$\sqrt{a}$	tfidf(u,x)	U^{src}
SM-2	$\sqrt{\frac{c^{\text{tst}}(u)}{c^{\text{trn}}(u)}\beta^{\lvert u\rvert}}$	$\sqrt{a}$	tfidf(u,x)	$U^{\text{src}} \cup U^{\text{tgt}}$
SM-3	$\frac{c^{\text{tst}}(u)}{c^{\text{trn}}(u)}\beta^{\lvert u\rvert}$	$\sqrt{a}$	c(u,x)	U^{src}
SM-4	$c^{\text{tst}}(u)$	$\sqrt{a}$	tfidf(u,x)	U^{src}
SM-5	1	$ln(1+a)$	c(u,x)	U^{src}
SM-6	$\sqrt{\frac{c^{\text{tst}}(u)}{c^{\text{trn}}(u)}}$	$\sqrt{a}$	tfidf(u,x)	U^{src}
SM-7	$\frac{c^{\text{tst}}(u)}{c^{\text{trn}}(u)}$	$\sqrt{(a)}$	tfidf(u,x)	$U^{\text{src}} \cup U^{\text{tgt}}$
SM-8	$\frac{c^{\text{tst}}(u)}{c^{\text{trn}}(u)}$	$ln(1+a)$	tfidf(u,x)	$U^{\text{src}} \cup U^{\text{tgt}}$

Table 4: Different instantiations of the general submodular function in Eq. 4 ($\beta = 1.5$ in all cases).

We first trained a baseline system on 100% of the training data. Different data selection methods were then used to select subsets of 10%, 20%, 30% and 40% of the data. While not reported in the tables, above 40%, the performance slowly drops to the 100% performance.

The first baseline selection method utilizes random data selection, for which 3 different data sets of the specified size were drawn randomly from the training data. Individual systems were trained on all random subsets of the same size, and their scores were averaged. The second baseline is the cross-entropy method by (Moore and Lewis, 2010). In-domain language models were trained on the combined development and test data, and out-of-domain models were trained on an equivalent amount of data drawn randomly from the training set. Sentences were ranked by the function in Eq. 5, and the top k percent were chosen. The order of the n-gram models was optimized on the development set and was found to be 3. Larger model orders resulted in worse performance, possibly due to the

limited size of the data used for their training. Since this method also involves random data selection, we report the average BLEU score over 5 different trials. For the submodular selection method, Table 4 shows the different values that were tested for the four components listed in Section 4. The combination was optimized on the development set. The selection algorithm (Alg. 1) runs within a few minutes on our complete training set of 189M words.

Results on the NIST 2009 test set are shown in Table 2. The scores for the submodular systems are averages over 3 different runs of MERT tuning. Random data subset selection (Row 1) falls short of the baseline system using 100% of the training data. The cross-entropy method (Row 2) surpasses the performance of the baseline system at about 20% of the data, demonstrating that data subset selection is a suitable technique for such mixed-domain translation tasks. The following rows show results for the various submodular functions shown in Table 4. Out of these, SM-5 corresponds to the best approach in (Biçici and Yuret, 2011). SM-6 is our own best-performing function, beating the cross-entropy method by a statistically significant margin ($p \leq 0.05$) under all conditions.[5] SM-6 is also significantly better than SM-5 in two cases. Finally, it surpasses the performance of the all-data system at only 10% of the training data; possibly, even smaller training data sets could be used but this option was not investigated. While the bilingual submodular functions SM-2 and SM-7) yield an improvement of up to 0.015 BLEU points on the dev set (not shown in the table), they do not consistently outperform the monolingual functions on the test set. Since test set target features cannot be used in our scenario, bilingual features are only helpful to the extent that the development set closely matches the test set. However, target features should be quite helpful when selecting data from an out-of-domain set to match an in-domain training set (as in e.g. (Axelrod et al., 2011)). We found no gain from the length reward $\beta^{|u|}$.

The Europarl results (Table 3) show a similar pattern. Although the differences in BLEU scores are smaller overall (as expected on an in-domain translation task), data subset selection improves over the all-data baseline system in this case as well. The cross-entropy method again outperforms random data selection. On this task we only tested our submodular function that worked best on the NIST task; again we find that it outperforms the cross-entropy method. In two conditions (10% and 30%) these differences are statistically significant. 10% of the training data suffices to outperform the all-data system, and up to a full BLEU point can be gained on this task using 20-30% of the data and a submodular data selection method.

[5] Statistical significance was measured using the paired bootstrap resampling test of (Koehn, 2004), applied to the systems with the median BLEU scores.

7 Conclusions

We have introduced submodularity to SMT data subset selection, generalizing previous approaches to this problem. Our method has theoretical performance guarantees, comes with scalable algorithms, and significantly improves over current, widely-used data selection methods on two different translation tasks. There are many possible extensions to this work. One strategy would be to extend the feature set U with features representing different types of linguistic information - e.g., when using a syntax-based system it might be advantageous to select training data that covers the set of syntactic structures seen in the test data. Secondly, the selected data was test data specific. In some contexts, it is not possible to train test data specific systems dynamically; in that case, different submodular functions could be designed to select a representative "summary" of the training data. Finally, the use of submodular functions for subset selection is applicable to other data sets that can be represented as features or as a pairwise similarity graph. Submodularity thus can be applied to a wide range of problems in NLP beyond machine translation.

Acknowledgments

This material is based on research sponsored by Intelligence Advanced Research Projects Activity (IARPA) under agreement number FA8650-12-2-7263, and is also supported by the National Science Foundation under Grant No. (IIS-1162606), and by a Google, a Microsoft, and an Intel research award. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Intelligence Advanced Research Projects Activity (IARPA) or the U.S. Government.

References

[Ambati et al.2010] V. Ambati, S. Vogel, and J. Carbonell. 2010. Active learning and crowd-sourcing for machine translation. In *Proceedings of LREC*, pages 2169–2174, Valletta, Malta.

[Axelrod et al.2011] A. Axelrod, X. He, and J. Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of EMNLP*, pages 355–362, Edinburgh, Scotland.

[Biçici and Yuret2011] E. Biçici and D. Yuret. 2011. Instance selection for machine translation using feature decay algorithms. In *Proceedings of the 6th Workshop on Statistical Machine Translation*, pages 272–283.

[Biçici2013] E. Biçici. 2013. Feature decay algorithms for fast deployment of accurate statistical machine translation systems. In *Proceedings of the 8th Workshop on Statistical Machine Translation*, pages 78–84.

[Biçici2011] E. Biçici. 2011. *The Regression Model of Machine Translation*. Ph.D. thesis, KOÇ University.

[Bloodgood and Callison-Burch2010] M. Bloodgood and C. Callison-Burch. 2010. Bucking the trend: large-scale cost-focused active learning for statistical machine translation. In *Proceedings of ACL*, pages 854–864.

[Callison-Burch et al.2005] Chris Callison-Burch, Colin Bannard, and Josh Schroeder. 2005. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 255–262. Association for Computational Linguistics.

[Cao and Khudanpur2012] Y. Cao and S. Khudanpur. 2012. Sample selection for large-scale MT discriminative training. In *Proceedings of AMTA*.

[Conforti and Cornuejols1984] M. Conforti and G. Cornuejols. 1984. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete Applied Mathematics*, 7(3):251–274.

[Dasgupta and Hsu2008] S. Dasgupta and D. Hsu. 2008. Hierarchical sampling for active learning. In *Proceedings of ICML*.

[Eck et al.2005] M. Eck, S. Vogel, and A. Waibel. 2005. Low cost portability for statistical machine translation based on n-gram frequency and tf-idf. In *Proceedings of the 10th Machine Translation Summit X*, pages 227–234.

[Edmonds1970] J. Edmonds, 1970. *Combinatorial Structures and their Applications*, chapter Submodular functions, matroids and certain polyhedra, pages 69–87. Gordon and Breach.

[Feige1998] U. Feige. 1998. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652.

[Fujishige2005] S. Fujishige. 2005. *Submodular functions and optimization. Annals of Discrete Mathematics*, volume 58. Elsevier Science.

[Galley and Manning2008] M. Galley and C. D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of EMNLP*, pages 847–855.

[Habash et al.2009] N. Habash, O. Rambow, and R. Roth. 2009. A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In *Proceedings of the MEDAR conference*, pages 102–109.

[Haffari et al.2009] G. Haffari, M. Roy, and A. Sarkar. 2009. Active learning for statistical machine translation. In *Proceedings of HLT*, pages 415–423.

[Hildebrand et al.2005] A. Hildebrand, M. Eck, S. Vogel, and A. Waibel. 2005. Adaptation of the translation model for statistical machine translation based on information retrieval. In *Proceedings of EAMT*, pages 133–142.

[Ittycheriah and Roukos2007] A. Ittycheriah and S. Roukos. 2007. Direct translation model 2. In *Proceedings of HLT/NAACL*, page 5764.

[Iyer and Bilmes2013] R. Iyer and J. Bilmes. 2013. Submodular optimization with submodular cover and submodular knapsack constraints. In *Neural Information Processing Society (NIPS)*, Lake Tahoe, CA, December.

[Jegelka and Bilmes2011] Stefanie Jegelka and Jeff A. Bilmes. 2011. Submodularity beyond submodular energies: coupling edges in graph cuts. In *Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, CO, June.

[Jurafsky and Martin2009] D. Jurafsky and J. H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall, 2nd edition.

[Koehn et al.2007] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL*.

[Koehn2004] P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*.

[Kolmogorov and Zabih2004] V. Kolmogorov and R. Zabih. 2004. What energy functions can be minimized via graph cuts? *IEEE TPAMI*, 26(2):147–159.

[Krause and Guestrin2011] A. Krause and C. Guestrin. 2011. Submodularity and its applications in optimized information gathering. *ACM Transactions on Intelligent Systems and Technology*, 2(4).

[Krause et al.2008] A. Krause, H.B. McMahan, C. Guestrin, and A. Gupta. 2008. Robust submodular observation selection. *Journal of Machine Learning Research*, 9:2761–2801.

[Leskovec et al.2007] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 420–429.

[Lin and Bilmes2010] H. Lin and J. Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of NAACL-HLT*, pages 2761–2801.

[Lin and Bilmes2011] H. Lin and J. Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of ACL*, pages 510–520.

[Lin and Bilmes2012] H. Lin and J. Bilmes. 2012. Learning mixtures of submodular shells with application to document summarization. In *Uncertainty in Artifical Intelligence (UAI)*, Catalina Island, USA, July. AUAI.

[Lopez2007] A. Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *EMNLP-CoNLL*, pages 976–985.

[Lü et al.2007] Y. Lü, J. Huang, and Q. Liu. 2007. Improving statistical machine translation performance by training data selection and optimization. In *Proceedings of EMNLP*, pages 343–350.

[Mandal et al.2008] A. Mandal, D. Vergyri, W. Wang, J. Zheng, A. Stolcke, D. Hakkani-Tür G. Tür, and N.F. Ayan. 2008. Efficient data selection for machine translation. In *Proceedings of the Spoken Language Technology Workshop*, pages 261–264.

[Minoux1978] M. Minoux. 1978. Accelerated greedy algorithms for maximizing submodular functions. In *Lecture Notes in Control and Information Sciences*, volume 7, pages 234–243.

[Mirzasoleiman et al.2013] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause. 2013. Distributed submodular maximization: Identifying representative elements in massive data. In *Neural Information Processing Systems (NIPS)*.

[Moore and Lewis2010] R. Moore and W. Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the Association for Computational Linguistics*, pages 220–224.

[Narasimhan and Bilmes2004] M. Narasimhan and J. Bilmes. 2004. PAC-learning bounded tree-width graphical models. In *Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference (UAI-2004)*. Morgan Kaufmann Publishers, July.

[Narayanan1997] H. Narayanan. 1997. Submodular functions and electrical networks. *Elsevier*.

[Nemhauser et al.1978] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions i. *Mathematical Programming*, 14:265–294.

[Stobbe and Krause2010] P. Stobbe and A. Krause. 2010. Efficient minimization of decomposable submodular functions. In *NIPS*.

[Sviridenko2004] M. Sviridenko. 2004. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43.

[Turchi et al.2012a] M. Turchi, T. de Bie, C. Goutte, and N. Cristianini. 2012a. Learning to translate: A statistical and computational analysis. *Advances in Artificial Intellligence*, 2012:484580:15 pages.

[Turchi et al.2012b] M. Turchi, C. Goutte, and N. Cristianini. 2012b. Learning machine translation from in-domain and out-of-domain data. In *Proceedings of EAMT*, Trento, Italy.

[Wei et al.2013] K. Wei, Y. Liu, K. Kirchhoff, and J. Bilmes. 2013. Using document summarization techniques for speech data subset selection. In *Proceedings of NAACL*, pages 721–726, Atlanta, Georgia, June.

[Wei et al.2014] K. Wei, R. Iyer, and Jeff Bilmes. 2014. Fast multi-stage submodular maximization. In *Proceedings of ICML*, Beijing, China.

Improve Statistical Machine Translation with Context-Sensitive Bilingual Semantic Embedding Model

Haiyang Wu[1] **Daxiang Dong**[1] **Wei He**[1] **Xiaoguang Hu**[1] **Dianhai Yu**[1]
Hua Wu[1] **Haifeng Wang**[1] **Ting Liu**[2]
[1] Baidu Inc., No. 10, Shangdi 10th Street, Beijing, 100085, China
[2] Harbin Institute of Technology, Harbin, China
wuhaiyang,dongdaxiang,hewei,huxiaoguang,yudianhai,
wu_hua,wanghaifeng@baidu.com
tliu@ir.hit.edu.cn

Abstract

We investigate how to improve *bilingual embedding* which has been successfully used as a feature in phrase-based *statistical machine translation* (SMT). Despite bilingual embedding's success, the *contextual information*, which is of critical importance to translation quality, was ignored in previous work. To employ the contextual information, we propose a simple and memory-efficient model for learning bilingual embedding, taking both the source phrase and context around the phrase into account. Bilingual translation scores generated from our proposed bilingual embedding model are used as features in our SMT system. Experimental results show that the proposed method achieves significant improvements on large-scale Chinese-English translation task.

1 Introduction

In Statistical Machine Translation (SMT) system, it is difficult to determine the translation of some phrases that have ambiguous meanings.For example, the phrase "结果 jieguo" can be translated to either "results", "eventually" or "fruit", depending on the context around it. There are two reasons for the problem: First, the length of phrase pairs is restricted due to the limitation of model size and training data. Another reason is that SMT systems often fail to use contextual information in source sentence, therefore, phrase sense disambiguation highly depends on the language model which is trained only on target corpus.

To solve this problem, we present to learn context-sensitive bilingual semantic embedding. Our methodology is to train a supervised model where labels are automatically generated from phrase-pairs. For each source phrase, the aligned target phrase is marked as the positive label whereas other phrases in our phrase table are treated as negative labels. Different from previous work in bilingual embedding learning(Zou et al., 2013; Gao et al., 2014), our framework is a supervised model that utilizes contextual information in source sentence as features and make use of phrase pairs as weak labels. Bilingual semantic embeddings are trained automatically from our supervised learning task.

Our learned bilingual semantic embedding model is used to measure the similarity of phrase pairs which is treated as a feature in decoding. We integrate our learned model into a phrase-based translation system and experimental results indicate that our system significantly outperform the baseline system. On the NIST08 Chinese-English translation task, we obtained 0.68 BLEU improvement. We also test our proposed method on much larger web dataset and obtain 0.49 BLEU improvement against the baseline.

2 Related Work

Using vectors to represent word meanings is the essence of vector space models (VSM). The representations capture words' semantic and syntactic information which can be used to measure semantic similarities by computing distance between the vectors. Although most VSMs represent one word with only one vector, they fail to capture homonymy and polysemy of word. Huang et al. (2012) introduced global document context and multiple word prototypes which distinguishes and uses both local and global context via a joint training objective. Much of the research focus on the task of inducing representations for single languages. Recently, a lot of progress has

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 142–146,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

been made at representation learning for bilingual words. Bilingual word representations have been presented by Peirsman and Padó (2010) and Sumita (2000). Also unsupervised algorithms such as LDA and LSA were used by Boyd-Graber and Resnik (2010), Tam et al. (2007) and Zhao and Xing (2006). Zou et al. (2013) learn bilingual embeddings utilizes word alignments and monolingual embeddings result, Le et al. (2012) and Gao et al. (2014) used continuous vector to represent the source language or target language of each phrase, and then computed translation probability using vector distance. Vulić and Moens (2013) learned bilingual vector spaces from non-parallel data induced by using a seed lexicon. However, none of these work considered the word sense disambiguation problem which Carpuat and Wu (2007) proved it is useful for SMT. In this paper, we learn bilingual semantic embeddings for source content and target phrase, and incorporate it into a phrase-based SMT system to improve translation quality.

3 Context-Sensitive Bilingual Semantic Embedding Model

We propose a simple and memory-efficient model which embeds both contextual information of source phrases and aligned phrases in target corpus into low dimension. Our assumption is that high frequent words are likely to have multiple word senses; therefore, top frequent words are selected in source corpus. We denote our selected words as focused phrase. Our goal is to learn a bilingual embedding model that can capture discriminative contextual information for each focused phrase. To learn an effective context sensitive bilingual embedding, we extract context features nearby a focused phrase that will discriminate focused phrase's target translation from other possible candidates. Our task can be viewed as a classification problem that each target phrase is treated as a class. Since target phrases are usually in very high dimensional space, traditional linear classification model is not suitable for our problem. Therefore, we treat our problem as a ranking problem that can handle large number of classes and optimize the objectives with scalable optimizer stochastic gradient descent.

3.1 Bilingual Word Embedding

We apply a linear embedding model for bilingual embedding learning. Cosine similarity between bilingual embedding representation is considered as score function. The score function should be discriminative between target phrases and other candidate phrases. Our score function is in the form:

$$f(\mathbf{x}, \mathbf{y}; \mathbf{W}, \mathbf{U}) = cos(\mathbf{W}^T\mathbf{x}, \mathbf{U}^T\mathbf{y}) \quad (1)$$

where $\mathbf{x}$ is contextual feature vector in source sentence, and $\mathbf{y}$ is the representation of target phrase, $\mathbf{W} \in R^{|\mathbf{X}|\times k}, \mathbf{U} \in R^{|\mathbf{Y}|\times k}$ are low rank matrix. In our model, we allow $\mathbf{y}$ to be bag-of-words representation. Our embedding model is memory-efficient in that dimensionality of $\mathbf{x}$ and $\mathbf{y}$ can be very large in practical setting. We use $|\mathbf{X}|$ and $|\mathbf{Y}|$ means dimensionality of random variable $\mathbf{x}$ and $\mathbf{y}$, then traditional linear model such as max-entropy model requires memory space of $O(|\mathbf{X}||\mathbf{Y}|)$. Our embedding model only requires $O(k(|\mathbf{X}| + |\mathbf{Y}|))$ memory space that can handle large scale vocabulary setting. To score a focused phrase and target phrase pair with $f(\mathbf{x}, \mathbf{y})$, context features are extracted from nearby window of the focused phrase. Target words are selected from phrase pairs. Given a source sentence, embedding of a focused phrase is estimated from $\mathbf{W}^T\mathbf{x}$ and target phrase embedding can be obtained through $\mathbf{U}^T\mathbf{y}$.

3.2 Context Sensitive Features

Context of a focused phrase is extracted from nearby window, and in our experiment we choose window size of 6 as a focused phrase's context. Features are then extracted from the focused phrase's context. We demonstrate our feature extraction and label generation process from the Chinese-to-English example in figure 1. Window size in this example is three. Position features and Part-Of-Speech Tagging features are extracted from the focused phrase's context. The word *fruit*

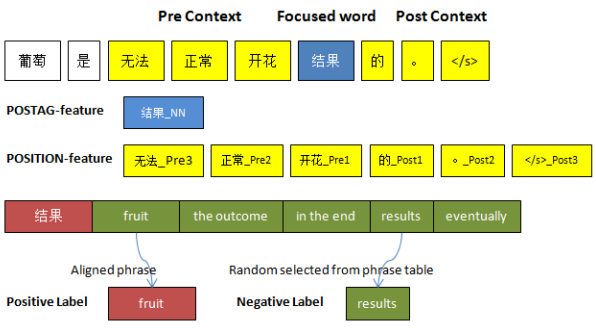


Figure 1: Feature extraction and label generation

is the aligned phrase of our focused phrase and is treated as positive label. The phrase *results* is a randomly selected phrase from phrase table results of 结果. Note that feature window is not well defined near the beginning or the end of a sentence. To conquer this problem, we add special padding word to the beginning and the end of a sentence to augment sentence.

3.3 Parameter Learning

To learn model parameter $\mathbf{W}$ and $\mathbf{U}$, we apply a ranking scheme on candidates selected from phrase table results of each focused phrase. In particular, given a focus phrase w, aligned phrase is treated as positive label whereas phrases extracted from other candidates in phrase table are treated as negative label. A max-margin loss is applied in this ranking setting.

$$I(\Theta) = \frac{1}{m}\sum_{i=1}^{m}(\delta - f(x_i, y_i; \Theta) - f(x_i, y_i'; \Theta))+ \quad (2)$$

Where $f(\mathbf{x_i}, \mathbf{y_i})$ is previously defined, $\Theta = \{W, U\}$ and + means max-margin hinge loss. In our implementation, a margin of $\delta = 0.15$ is used during training. Objectives are minimized through stochastic gradient descent algorithm. For each randomly selected training example, parameters are updated through the following form:

$$\mathbf{\Theta} := \mathbf{\Theta} - \alpha\frac{\partial l(\mathbf{\Theta})}{\partial \mathbf{\Theta}} \quad (3)$$

where $\Theta = \{\mathbf{W}, \mathbf{U}\}$. Given an instance with positive and negative label pair $\{\mathbf{x}, \mathbf{y}, \mathbf{y}'\}$, gradients of parameter $\mathbf{W}$ and $\mathbf{U}$ are as follows:

$$\frac{\partial l(\mathbf{W}, \mathbf{U})}{\partial \mathbf{W}} = qs\mathbf{x}(\mathbf{W}^T\mathbf{x})^T - pqs^3\mathbf{x}(\mathbf{U}^T\mathbf{y}) \quad (4)$$

$$\frac{\partial l(\mathbf{W}, \mathbf{U})}{\partial \mathbf{U}} = qs\mathbf{y}(\mathbf{U}^T\mathbf{y})^T - pqs^3\mathbf{y}(\mathbf{W}^T\mathbf{x}) \quad (5)$$

Where we set $p = (\mathbf{W}^T\mathbf{x})^T(\mathbf{U}^T\mathbf{y})$, $q = \frac{1}{||\mathbf{W}^T\mathbf{x}||_2}$ and $s = \frac{1}{||\mathbf{U}^T\mathbf{y}||_2}$. To initialize our model parameters with strong semantic and syntactic information, word vectors are pre-trained independently on source and target corpus through word2vec (Mikolov et al., 2013). And the pre-trained word vectors are treated as initial parameters of our model. The learned scoring function $f(\mathbf{x}, \mathbf{y})$ will be used during decoding phase as a feature in log-linear model which we will describe in detail later.

4 Integrating Bilingual Semantic Embedding into Phrase-Based SMT Architectures

To incorporate the context-sensitive bilingual embedding model into the state-of-the-art Phrase-Based Translation model, we modify the decoding so that context information is available on every source phrase. For every phrase in a source sentence, the following tasks are done at every node in our decoder:

- Get the focused phrase as well as its context in the source sentence.
- Extract features from the focused phrase's context.
- Get translation candidate extracted from phrase pairs of the focused phrase.
- Compute scores for any pair of the focused phrase and a candidate phrase.

We get the target sub-phrase using word alignment of phrase, and we treat NULL as a common target word if there is no alignment for the focused phrase. Finally we compute the matching score for source content and target word using bilingual semantic embedding model. If there are more than one word in the focus phrase, then we add all score together. A penalty value will be given if target is not in translation candidate list. For each phrase in a given SMT input sentence, the Bilingual Semantic score can be used as an additional feature in log-linear translation model, in combination with other typical context-independent SMT bilexicon probabilities.

5 Experiment

Our experiments are performed using an in-house phrase-based system with a log-linear framework. Our system includes a phrase translation model, an n-gram language model, a lexicalized reordering model, a word penalty model and a phrase penalty model, which is similar to Moses (Koehn et al., 2007). The evaluation metric is BLEU (Papineni et al., 2002).

5.1 Data set

We test our approach on LDC corpus first. We just use a subset of the data available for NIST OpenMT08 task[1] . The parallel training corpus

[1]LDC2002E18, LDC2002L27, LDC2002T01, LDC2003E07, LDC2003E14, LDC2004T07, LDC2005E83, LDC2005T06, LDC2005T10, LDC2005T34, LDC2006E24, LDC2006E26, LDC2006E34, LDC2006E86, LDC2006E92, LDC2006E93, LDC2004T08(HK_News, HK_Hansards)

Method	OpenMT08	WebData
	BLEU	BLEU
Our Baseline	26.24	29.32
LOC	26.78**	29.62*
LOC+POS	26.82**	29.81*

Table 1: Results of lowercase BLEU on NIST08 task. LOC is the location feature and POS is the Part-of-Speech feature * or ** equals to significantly better than our baseline($\rho < 0.05$ or $\rho < 0.01$, respectively)

contains 1.5M sentence pairs after we filter with some simple heuristic rules, such as sentence being too long or containing messy codes. As monolingual corpus, we use the XinHua portion of the English GigaWord. In monolingual corpus we filter sentence if it contain more than 100 words or contain messy codes, Finally, we get monolingual corpus containing 369M words. In order to test our approach on a more realistic scenario, we train our models with web data. Sentence pairs obtained from bilingual website and comparable webpage. Monolingual corpus is gained from some large website such as WiKi. There are 50M sentence pairs and 10B words monolingual corpus.

5.2 Results and Analysis

For word alignment, we align all of the training data with GIZA++ (Och and Ney, 2003), using the grow-diag-final heuristic to improve recall. For language model, we train a 5-gram modified Kneser-Ney language model and use Minimum Error Rate Training (Och, 2003) to tune the SMT. For both OpenMT08 task and WebData task, we use NIST06 as the tuning set, and use NIST08 as the testing set. Our baseline system is a standard phrase-based SMT system, and a language model is trained with the target side of bilingual corpus. Results on Chinese-English translation task are reported in Table 1. Word position features and part-of-speech tagging features are both useful for our bilingual semantic embedding learning. Based on our trained bilingual embedding model, we can easily compute a translation score between any bilingual phrase pair. We list some cases in table 2 to show that our bilingual embedding is context sensitive.

Contextual features extracted from source sentence are strong enough to discriminate different word senses. And we also observe from the word “结果 jieguo” that Part-Of-Speech Tagging features are effective in discriminating target phrases.

Source Sentence	4 Nearest Neighbor from bilingual embedding
只有稳定的社会环境，投资者**才能**踏踏实实地做生意。(Investors can only get down to business in a stable social environment)	will be, can only, will, can
在比赛与交往中，中国残疾人显示了非凡的体育**才能**。(In competitions, the Chinese Disabled have shown extraordinary athletic abilities)	skills, ability, abilities, talent
在哥国的自然环境下，葡萄是无法正常开花**结果**的。(In the natural environment of Costa Rica, grapes do not normally yield fruit.)	fruit, outcome of, the outcome, result
结果，东区区议会通过一项议案。(As a result, Eastern District Council passed a proposal)	in the end, eventually, as a result, results

Table 2: Top ranked focused phrases based on bilingual semantic embedding

6 Conlusion

In this paper, we proposed a context-sensitive bilingual semantic embedding model to improve statistical machine translation. Contextual information is used in our model for bilingual word sense disambiguation. We integrated the bilingual semantic model into the phrase-based SMT system. Experimental results show that our method achieves significant improvements over the baseline on large scale Chinese-English translation task. Our model is memory-efficient and practical for industrial usage that training can be done on large scale data set with large number of classes. Prediction time is also negligible with regard to SMT decoding phase. In the future, we will explore more features to refine the model and try to utilize contextual information in target sentences.

Acknowledgments

We thank the three anonymous reviewers for their valuable comments, and Niu Gang and Wu Xianchao for discussions. This paper is supported by 973 program No. 2014CB340505.

References

Jordan Boyd-Graber and Philip Resnik. 2010. Holistic sentiment analysis across languages: Multilingual supervised latent dirichlet allocation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 45–55, Cambridge, MA, October. Association for Computational Linguistics.

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 61–72, Prague, Czech Republic, June. Association for Computational Linguistics.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proc. ACL*.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea, July. Association for Computational Linguistics.

Hai-Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 39–48, Montréal, Canada, June. Association for Computational Linguistics.

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

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. In *Computational Linguistics, Volume 29, Number 1, March 2003*. Computational Linguistics, March.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

Yves Peirsman and Sebastian Padó. 2010. Cross-lingual induction of selectional preferences with bilingual vector spaces. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 921–929, Los Angeles, California, June. Association for Computational Linguistics.

Eiichiro Sumita. 2000. Lexical transfer using a vector-space model. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, August.

Yik-Cheung Tam, Ian Lane, and Tanja Schultz. 2007. Bilingual-lsa based lm adaptation for spoken language translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 520–527, Prague, Czech Republic, June. Association for Computational Linguistics.

Ivan Vulić and Marie-Francine Moens. 2013. Cross-lingual semantic similarity of words as the similarity of their semantic word responses. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 106–116, Atlanta, Georgia, June. Association for Computational Linguistics.

Bing Zhao and Eric P. Xing. 2006. Bitam: Bilingual topic admixture models for word alignment. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 969–976, Sydney, Australia, July. Association for Computational Linguistics.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398, Seattle, Washington, USA, October. Association for Computational Linguistics.

Transformation from Discontinuous to Continuous Word Alignment Improves Translation Quality

Zhongjun He[1] **Hua Wu**[1] **Haifeng Wang**[1] **Ting Liu**[2]
[1] Baidu Inc., No. 10, Shangdi 10th Street, Beijing, 100085, China
[2] Harbin Institute of Technology, Harbin, China
{hezhongjun,wu_hua,wanghaifeng}@baidu.com
tliu@ir.hit.edu.cn

Abstract

We present a novel approach to improve word alignment for statistical machine translation (SMT). Conventional word alignment methods allow discontinuous alignment, meaning that a source (or target) word links to several target (or source) words whose positions are discontinuous. However, we cannot extract phrase pairs from this kind of alignments as they break the alignment consistency constraint. In this paper, we use a weighted vote method to transform discontinuous word alignment to continuous alignment, which enables SMT systems extract more phrase pairs. We carry out experiments on large scale Chinese-to-English and German-to-English translation tasks. Experimental results show statistically significant improvements of BLEU score in both cases over the baseline systems. Our method produces a gain of +1.68 BLEU on NIST OpenMT04 for the phrase-based system, and a gain of +1.28 BLEU on NIST OpenMT06 for the hierarchical phrase-based system.

1 Introduction

Word alignment, indicating the correspondence between the source and target words in bilingual sentences, plays an important role in statistical machine translation (SMT). Almost all of the SMT models, not only phrase-based (Koehn et al., 2003), but also syntax-based (Chiang, 2005; Liu et al., 2006; Huang et al., 2006), derive translation knowledge from large amount bilingual text annotated with word alignment. Therefore, the quality of the word alignment has big impact on the quality of translation output.

Word alignments are usually automatically obtained from a large amount of bilingual training corpus. The most widely used toolkit for word alignment in SMT community is GIZA++ (Och and Ney, 2004), which implements the well known IBM models (Brown et al., 1993) and the HMM model (Vogel and Ney, 1996). Koehn et al. (2003) proposed some heuristic methods (e.g. the "*grow-diag-final"* method) to refine word alignments trained by GIZA++. Another group of word alignment methods (Liu et al., 2005; Moore et al., 2006; Riesa and Marcu, 2010) define feature functions to describe word alignment. They need manually aligned bilingual texts to train the model. However, the manually annotated data is too expensive to be available for all languages. Although these models reported high accuracy, the GIZA++ and "grow-diag-final" method are dominant in practice.

However, automatic word alignments are usually very noisy. The example in Figure 1 shows a Chinese and English sentence pair, with word alignment automatically trained by GIZA++ and the "grow-diag-final" method. We find many errors (dashed links) are caused by discontinuous alignment (formal definition is described in Section 2), a source (or target) word linking to several discontinuous target (or source) words. This kind of errors will result in the loss of many useful phrase pairs that are learned based on bilingual word alignment. Actually, according to the definition of phrases in a standard phrase-based model, we cannot extract phrases from the discontinuous alignment. The reason is that this kind of alignment break the alignment consistency constraint (Koehn et al., 2003). For example, the Chi-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 147–152,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

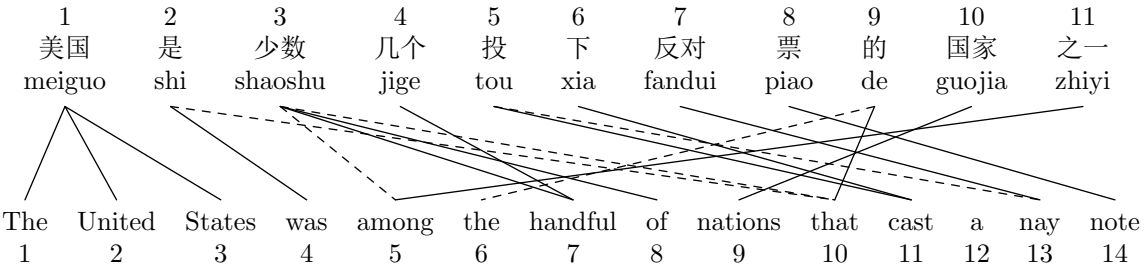


Figure 1: An example of word alignment between a Chinese and English sentence pair. The dashed links are incorrect alignments.

nese word "*shi*$_2$"[1] is aligned to the English words "*was*$_4$" and "*that*$_{10}$". However, these two English words are discontinuous, and we cannot extract the phrase pair "(*shi, was*)".

In this paper, we propose a simple *weighed vote* method to deal with the discontinuous word alignment. Firstly, we split the discontinuous alignment into several continuous alignment groups, and consider each continuous alignment group as a bucket. Secondly, we vote for each bucket with alignment score measured by word translation probabilities. Finally, we select the bucket with the highest score as the final alignment. The strength of our method is that we refine word alignment without using any external knowledge, as the word translation probabilities can be estimated from the bilingual corpus with the original word alignment.

We notice that the discontinuous alignment is helpful for hierarchical phrase-based model, as the model allows discontinuous phrases. Thus, for the hierarchical phrase-based model, our method may lost some discontinuous phrases. To solve the problem, we keep the original discontinuous alignment in the training corpus.

We carry out experiment with the state-of-the-art phrase-based and hierarchical phrase-based (Chiang, 2005) SMT systems implemented in Moses (Koehn et al., 2007). Experiments on large scale Chinese-to-English and German-to-English translation tasks demonstrate significant improvements in both cases over the baseline systems.

2 The Weighted Vote Method

To refine the discontinuous alignment, we propose a weighted vote method to transform discontinuous alignment to continuous alignment by discarding noisy links. We split discontinuous alignment into several continuous groups, and select the best group with the highest score computed by word translation probabilities as the final alignment.

For further understanding, we first describe some definitions. Given a word-aligned sentence pair (F_1^I, E_1^J, A), an **alignment set** $A_{set}(i)$ is the set of target word positions that aligned to the source word F_i^i:

$$A_{set}(i) = \{j|(i,j) \in A\} \quad (1)$$

For example, in Figure 1, the alignment set for the Chinese word "shaoshu$_3$" is $A_{set}(3) = \{5,7,8,10\}$. We define an **alignment span** $A_{span}(i)$ as $[min(A_{set}(i)), max(A_{set}(i))]$. Thus, the alignment span for the Chinese word "shaoshu$_3$" is $A_{span}(3) = [5,10]$.

The alignment for F_i^i is discontinuous if there exist some target words in $A_{span}(i)$ linking to another source word, i.e. $\exists(i',j') \in A$, where $i' \neq i$, $j' \in A_{span}(i)$. Otherwise, the alignment is continuous. According to the definition, the alignment for "shaoshu$_3$" is discontinuous. Because the target words "the$_6$" and "nations$_9$" in the alignment span link to another Chinese words "de$_9$" and "guojia$_{10}$", respectively. For a target word E_j^j, the definition is similar.

If the alignment for F_i^i is discontinuous, we can split the alignment span $A_{span}(i) = [j_1, j_2]$ into m continuous spans $\{[j_p^k, j_q^k]\}$, where $k = 1, 2, ..., m$, and $j_p^k, j_q^k \in [j_1, j_2]$. Our goal is to select the best continuous span for the word F_i^i. To do this, we score each continuous span with word translation probabilities:

$$S([j_p^k, j_q^k]) = \sum_{t=p}^{q} (Pr(E_{j_t^k}|F_i) + Pr(F_i|E_{j_t^k})) \quad (2)$$

where,

$$Pr(f|e) = \frac{count(f,e)}{\sum_{f'} count(f',e)} \quad (3)$$

[1]The subscript denotes the word position.

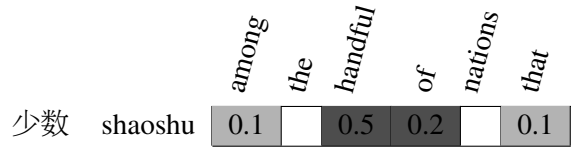


Figure 2: An example of weighted voted method for selecting the best continuous alignment from the discontinuous alignment. The heavy shading area is selected as the final alignment.

$$Pr(e|f) = \frac{count(e, f)}{\sum_{e'} count(f, e')} \quad (4)$$

The word translation probabilities can be computed from the bilingual corpus with the initial word alignment. Finally, we select the span with the highest score as the final alignment, and discard all other alignments.

We illustrate our method in Figure 2, which shows the source word "shaoshu" and its alignment in Figure 1. We split the alignments into three continuous alignment spans and compute score for each span. Finally, the span with highest score (heavy shading area) is selected as the final alignment.

We conduct the procedure for each source and target word, the improved alignment (solid links) is shown in Figure 1.

3 Experiment

To demonstrate the effect of the proposed method, we use the state-of-the-art phrase-based system and hierarchical phrase-based system implemented in Moses (Koehn et al., 2007). The phrase-based system uses continuous phrase pair as the main translation knowledge. While the hierarchical phrase-based system uses both continuous and discontinuous phrase pairs, which has an ability to capture long distance phrase reordering.

we carried out experiments on two translation tasks: the Chinese-to-English task comes from the NIST Open MT Evaluation, and the German-to-English task comes from the Workshop on Machine Translation (WMT) shared task.

3.1 Training

The training data we used are listed in Table 1. For the Chinese-English task, the bilingual data are selected from LDC. We used NIST MT03 as the development set and tested our system on NIST MT evaluation sets from 2004 to 2008. For the German-English task, the bilingual data are from the shared translation task 2013. We used WMT08 as the development set and tested our system on WMT test sets from 2010 to 2013.

Task	Src. Words	Tgt. Words
Chinese-to-English	75M	78M
German-to- English	107M	113M

Table 1: Bilingual data for our experiments.

System	N04	N05	N06	N08
Baseline	34.53	33.02	30.43	23.29
Refined	36.21	33.99	31.59	24.36

Table 2: Chinese-to-English translation quality of the phrase-based system.

System	W10	W11	W12	W13
Baseline	20.71	20.26	20.52	23.26
Refined	21.46	20.95	21.11	23.77

Table 3: German-to-English translation quality of the phrase-based system.

The baseline systems are trained on the training corpus with *initial* word alignment, which was obtained via GIZA++ and "grow-diag-final" method. Based on the initial word alignment, we computed word translation probabilities and used the proposed method to obtain a *refined* word alignment. Then we used the refined word alignment to train our SMT systems.

The translation results are evaluated by case-insensitive BLEU-4 (Papineni et al., 2002). The feature weights of the translation system are tuned with the standard minimum-error-rate-training (Och, 2003) to maximize the systems BLEU score on the development set.

3.2 Results

3.2.1 Phrase-based System

Table 2 shows Chinese-to-English translation quality of the phrase-based system. We observed that our refined method significantly outperformed the baseline word alignment on all test sets. The improvements are ranged from 0.97 to 1.68 BLEU%.

Table 3 shows German-to-English translation quality of the phrase-based system. The improvements are ranged from 0.51 to 0.75 BLEU%.

These results demonstrate that the proposed method improves the translation quality for

System	N04	N05	N06	N08
Baseline	37.33	34.81	32.20	25.33
Refined	37.91	35.36	32.75	25.40
Combined	38.13	35.63	33.48	25.66

Table 4: Chinese-to-English translation quality of the hierarchical phrase-based system.

System	W10	W11	W12	W13
Baseline	21.22	19.77	20.53	23.51
Refined	21.34	20.64	20.88	23.82
Combined	21.65	20.87	21.16	24.04

Table 5: German-to-English translation quality of the hierarchical phrase-based system.

phrase-based system. The reason is that by discarding noisy word alignments from the discontinuous alignments, the phrase pairs constrained by the noisy alignments can be extracted. Thus the system utilized more phrase pairs than the baseline did.

3.2.2 Hierarchical Phrase-based System

The hierarchical phrase-based system utilizes discontinuous phrase pairs for long distance phrase reordering. Some of the discontinuous phrase pairs are extracted from the discontinuous alignments. By transforming the discontinuous alignments to continuous alignments, on the one hand, we may lost some discontinuous phrase pairs. On the other hand, we may extract additional continuous and discontinuous phrase pairs as the alignment restriction is loose.

See Figure 3 for illustration. From the initial alignment, we can extract a hierarchical phrase pair *"(dang X_1 shi, when X_1)"* from the discontinuous alignment of the English word "*when*". However, the hierarchical phrase pair cannot be extracted from our refined alignment, because our method discards the link between the Chinese word "*dang*" and the English word "*when*". Instead, we can extract another hierarchical phrase pair *"(X_1 shi, when X_1)"*.

Does our method still obtain improvements on the hierarchical phrase-based system? Table 4 and Table 5 shows Chinese-to-English and German-to-English translation quality of the hierarchical phrase-based system, respectively. For Chinese-to-English translation, the refined alignment obtained improvements ranged from 0.07 to 0.58

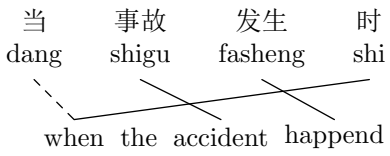


Figure 3: Example of word alignment between a Chinese and English sentence pair. The dashed initial link is discarded by our method.

BLEU% on the test set (the row "Refined"). While for German-to-English translation, the improvements ranged from 0.12 to 0.59 BLEU% on the test set (the row "Refined").

We find that the improvements are less than that of the phrase-based system. As discussed above, our method may lost some hierarchical phrase pairs that extracted from the discontinuous alignments. To solve the problem, we combine [2] the initial alignments and the refined alignments to train the SMT system. The results are shown in the row "Combined" in Table 4 and Table 5. For Chinese-to-English translation, we obtained an improvements of 1.28 BLEU% on NIST06 over the baseline. While for German-to-English translation, the greatest improvements is 1.10 BLEU% on WMT11.

4 Analyses

In order to further study the performance of the proposed method, we analyze the word alignment and the phrase table for Chinese-to-English translation. We find that our method improves the quality of word alignment. And as a result, more useful phrase pairs are extracted from the refined word alignment.

4.1 Word Alignment

The Chinese-to-English training corpus contains 4.5M sentence pairs. By applying GIZA++ and the "grow-diag-final" method, we obtained initial alignments. We find that 4.0M (accounting for 89%) sentence pairs contain discontinuous alignments. We then used the proposed method to discard noisy links. By doing this, the total links between words in the training corpus are reduced from 99.6M to 78.9M, indicating that 21% links are discarded.

[2] We do not perform combination for phrase-based system, because the phrase table extracted from the initial alignment is a subset of that extracted from the refined alignment.

Alignment	Precision	Recall	AER
Initial	62.94	89.55	26.07
Refined	73.43	87.82	20.01

Table 6: Precision, Recall and AER on Chinese-to-English alignment.

Alignment	StandPhr	HierPhr
Initial	29M	86M
Refined	104M	436M

Table 7: The phrase number extracted from the initial and refined alignment for the hierarchical phrase-based system on Chinese-to-English translation. StandPhr is standard phrase, HierPhr is hierarchical phrase.

We evaluated the alignment quality on 200 sentence pairs. Results are shown in Table 6. It is observed that our method improves the precision and decreases the AER, while keeping a high recall. This means that our method effectively discards noisy links in the initial word alignments.

4.2 Phrase Table

According to the standard definition of phrase in SMT, phrase pairs cannot be extracted from the discontinuous alignments. By transforming discontinuous alignments into continuous alignment, we can extract more phrase pairs. Table 7 shows the number of standard phrases and hierarchical phrases extracted from the initial and refined word alignments. We find that the number of both phrases and hierarchical phrases grows heavily. This is because that the word alignment constraint for phrase extraction is loosed by removing noisy links. Although the phrase table becomes larger, fortunately, there are some methods (Johnson et al., 2007; He et al., 2009) to prune phrase table without hurting translation quality.

For further illustration, we compare the phrase pairs extracted from the initial alignment and refined alignment in Figure 1. From the initial alignments, we extracted only 3 standard phrase pairs and no hierarchical phrase pairs (Table 8). After discarding noisy alignments (dashed links) by using the proposed method, we extracted 21 standard phrase pairs and 36 hierarchical phrases. Table 9 and Table 10 show selected phrase pairs and hierarchical phrase pairs, respectively.

Chinese	**English**
meiguo	The United States
guojia	nations
piao	note

Table 8: Phrase pairs extracted from the initial alignment of Figure 1.

Chinese	**English**
shi	was
fandui piao	a nay note
shaoshu jige	the handful of

Table 9: Selected phrase pairs extracted from the refined alignment of Figure 1.

Chinese	**English**
X_1 zhiyi	among X_1
X_1 de guojia	nations that X_1
X_1 fandui piao X_2	X_2 X_1 a nay note

Table 10: Selected hierarchical phrase pairs extracted from the refined alignment of Figure 1.

5 Conclusion and Future Work

In this paper, we proposed a novel method to improve word alignment for SMT. The method refines initial word alignments by transforming discontinuous alignment to continuous alignment. As a result, more useful phrase pairs are extracted from the refined word alignment. Our method is simple and efficient, since it uses only the word translation probabilities obtained from the initial alignments to discard noisy links. Our method is independent of languages and can be applied to most SMT models. Experimental results show significantly improvements for the state-of-the-art phrase-based and hierarchical phrase-based systems on all Chinese-to-English and German-to-English translation tasks.

In the future, we will refine the method by considering neighbor words and alignments when discarding noisy links.

Acknowlegement

This paper is supported by the 973 program No. 2014CB340505. We would like to thank Xuan Liu and the anonymous reviewers for their insightful comments.

References

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

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 263–270.

Zhongjun He, Yao Meng, and Hao Yu. 2009. Discarding monotone composed rule for hierarchical phrase-based statistical machine translation. In *Proceedings of the 3rd International Universal Communication Symposium*, pages 25–29.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas*.

Howard Johnson, Joel Martin, George Foster, and Roland Kuhn. 2007. Improving translation quality by discarding most of the phrasetable. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 967–975, Prague, Czech Republic, June.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL 2007 demonstration session*.

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Loglinear models for word alignment. In *Proceedings of of ACL 2005*, pages 459–466, Ann Arbor,Michigan, June.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616.

Robert C. Moore, Wen tau Yih, and Andreas Bode. 2006. Improved discriminative bilingual word alignment. In *In Proceedings of COLING/ACL 2006*, pages 513–520, Sydney, Australia, July.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. 30:417–449.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Jason Riesa and Daniel Marcu. 2010. Hierarchical search forword alignment. In *In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 157–166, Uppsala, Sweden, July.

Stephan Vogel and Hermann Ney. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of COLING 1996*, pages 836–841, Copenhagen, Danmark, August.

Unsupervised Word Alignment Using Frequency Constraint in Posterior Regularized EM

Hidetaka Kamigaito[1,2], **Taro Watanabe**[2], **Hiroya Takamura**[1], **Manabu Okumura**[1]
[1]Tokyo Institute of Technology, Precision and Intelligence Laboratory
4259 Nagatsuta-cho Midori-ku Yokohama, Japan
[2]National Institute of Information and Communication Technology
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan

Abstract

Generative word alignment models, such as IBM Models, are restricted to one-to-many alignment, and cannot explicitly represent many-to-many relationships in a bilingual text. The problem is partially solved either by introducing heuristics or by agreement constraints such that two directional word alignments agree with each other. In this paper, we focus on the posterior regularization framework (Ganchev et al., 2010) that can force two directional word alignment models to agree with each other during training, and propose new constraints that can take into account the difference between function words and content words. Experimental results on French-to-English and Japanese-to-English alignment tasks show statistically significant gains over the previous posterior regularization baseline. We also observed gains in Japanese-to-English translation tasks, which prove the effectiveness of our methods under grammatically different language pairs.

1 Introduction

Word alignment is an important component in statistical machine translation (SMT). For instance phrase-based SMT (Koehn et al., 2003) is based on the concept of phrase pairs that are automatically extracted from bilingual data and rely on word alignment annotation. Similarly, the model for hierarchical phrase-based SMT is built from exhaustively extracted phrases that are, in turn, heavily reliant on word alignment.

The Generative word alignment models, such as the IBM Models (Brown et al., 1993) and HMM (Vogel et al., 1996), are popular methods for automatically aligning bilingual texts, but are restricted to represent one-to-many correspondence of each word. To resolve this weakness, various symmetrization methods are proposed. Och and Ney (2003) and Koehn et al. (2003) propose various heuristic methods to combine two directional models to represent many-to-many relationships. As an alternative to heuristic methods, filtering methods employ a threshold to control the trade-off between precision and recall based on a score estimated from the posterior probabilities from two directional models. Matusov et al. (2004) proposed arithmetic means of two models as a score for the filtering, whereas Liang et al. (2006) reported better results using geometric means. The joint training method (Liang et al., 2006) enforces agreement between two directional models. Posterior regularization (Ganchev et al., 2010) is an alternative agreement method which directly encodes agreement during training. DeNero and Macherey (2011) and Chang et al. (2014) also enforce agreement during decoding.

However, these agreement models do not take into account the difference in language pairs, which is crucial for linguistically different language pairs, such as Japanese and English: although content words may be aligned with each other by introducing some agreement constraints, function words are difficult to align.

We focus on the posterior regularization framework and improve upon the previous work by proposing new constraint functions that take into account the difference in languages in terms of content words and function words. In particular, we differentiate between content words and function words by frequency in bilingual data, following Setiawan et al. (2007).

Experimental results show that the proposed methods achieved better alignment qualities on the French-English Hansard data and the Japanese-English Kyoto free translation task (KFTT) measured by AER and F-measure. In translation evaluations, we achieved statistically significant gains

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 153–158,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

in BLEU scores in the NTCIR10.

2 Statistical word alignment with posterior regularization framework

Given a bilingual sentence $\boldsymbol{x} = (\boldsymbol{x}^s, \boldsymbol{x}^t)$ where $\boldsymbol{x}^s$ and $\boldsymbol{x}^t$ denote a source and target sentence, respectively, the bilingual sentence is aligned by a many-to-many alignment of $\boldsymbol{y}$. We represent posterior probabilities from two directional word alignment models as $\overrightarrow{p}_\theta(\overrightarrow{\boldsymbol{y}}|\boldsymbol{x})$ and $\overleftarrow{p}_\theta(\overleftarrow{\boldsymbol{y}}|\boldsymbol{x})$ with each arrow indicating a particular direction, and use θ to denote the parameters of the models. For instance, $\overrightarrow{y}$ is a subset of $\boldsymbol{y}$ for the alignment from $\boldsymbol{x}^s$ to $\boldsymbol{x}^t$ under the model of $p(\boldsymbol{x}^t, \overrightarrow{\boldsymbol{y}}|\boldsymbol{x}^s)$. In the case of IBM Model 1, the model is represented as follows:

$$p(\boldsymbol{x}^t, \overrightarrow{\boldsymbol{y}}|\boldsymbol{x}^s) = \prod_i \frac{1}{|\boldsymbol{x}^s|+1} p_t(x_i^t|\boldsymbol{x}^s_{\overrightarrow{y}_i}). \quad (1)$$

where we define the index of $\boldsymbol{x}^t$, $\boldsymbol{x}^s$ as $i, j (1 \leq i \leq |\boldsymbol{x}^t|, 1 \leq j \leq |\boldsymbol{x}^s|)$ and the posterior probability for the word pair (x_i^t, x_j^s) is defined as follows:

$$\overrightarrow{p}(i, j|\boldsymbol{x}) = \frac{p_t(x_i^t|x_j^s)}{\sum_{j'} p_t(x_i^t|x_{j'}^s)}. \quad (2)$$

Herein, we assume that the posterior probability for wrong directional alignment is zero (i.e., $\overrightarrow{p}(\overleftarrow{\boldsymbol{y}}|\boldsymbol{x}) = 0$).[1] Given the two directional models, Ganchev et al. defined a symmetric feature for each target/source position pair, i, j as follows:

$$\phi_{i,j}(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} +1 & (\overrightarrow{\boldsymbol{y}} \subset \boldsymbol{y}) \cap (\overrightarrow{y}_i = j), \\ -1 & (\overleftarrow{\boldsymbol{y}} \subset \boldsymbol{y}) \cap (\overleftarrow{y}_j = i), \\ 0 & otherwise. \end{cases} \quad (3)$$

The feature assigns 1 for the subset of word alignment for $\overrightarrow{\boldsymbol{y}}$, but assigns -1 for $\overleftarrow{\boldsymbol{y}}$. As a result, if a word pair i, j is aligned with equal posterior probabilities in two directions, the expectation of the feature value will be zero. Ganchev et al. defined a joint model that combines two directional models using arithmetic means:

$$p_\theta(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{2}\overrightarrow{p}_\theta(\boldsymbol{y}|\boldsymbol{x}) + \frac{1}{2}\overleftarrow{p}_\theta(\boldsymbol{y}|\boldsymbol{x}). \quad (4)$$

Under the posterior regularization framework, we instead use q that is derived by maximizing the following posterior probability parametrized by $\boldsymbol{\lambda}$ for each bilingual data $\boldsymbol{x}$ as follows (Ganchev et al., 2010):

$$q_{\boldsymbol{\lambda}}(\boldsymbol{y}|\boldsymbol{x}) = \frac{\overrightarrow{p}_\theta(\overrightarrow{\boldsymbol{y}}|\boldsymbol{x}) + \overleftarrow{p}_\theta(\overleftarrow{\boldsymbol{y}}|\boldsymbol{x})}{2} \cdot \frac{exp\{-\boldsymbol{\lambda}\cdot\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})\}}{Z} \quad (5)$$

$$= \frac{\overrightarrow{q}(\overrightarrow{\boldsymbol{y}}|\boldsymbol{x})\frac{Z_{\overrightarrow{q}}}{\overrightarrow{p}_\theta(\boldsymbol{x})} + \overleftarrow{q}(\overleftarrow{\boldsymbol{y}}|\boldsymbol{x})\frac{Z_{\overleftarrow{q}}}{\overleftarrow{p}_\theta(\boldsymbol{x})}}{2Z},$$

$$Z = \frac{1}{2}\left(\frac{Z_{\overrightarrow{q}}}{\overrightarrow{p}_\theta} + \frac{Z_{\overleftarrow{q}}}{\overleftarrow{p_\theta}}\right),$$

$$\overrightarrow{q}(\overrightarrow{\boldsymbol{y}}|\boldsymbol{x}) = \frac{1}{Z_{\overrightarrow{q}}}\overrightarrow{p}_\theta(\overrightarrow{\boldsymbol{y}}, \boldsymbol{x}) exp\{-\boldsymbol{\lambda}\cdot\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})\},$$

$$Z_{\overrightarrow{q}} = \sum_{\overrightarrow{y}} \overrightarrow{p}_\theta(\overrightarrow{\boldsymbol{y}}, \boldsymbol{x}) exp\{-\boldsymbol{\lambda}\cdot\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})\},$$

$$\overleftarrow{q}(\overleftarrow{\boldsymbol{y}}|\boldsymbol{x}) = \frac{1}{Z_{\overleftarrow{q}}}\overleftarrow{p}_\theta(\overleftarrow{\boldsymbol{y}}, \boldsymbol{x}) exp\{-\boldsymbol{\lambda}\cdot\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})\},$$

$$Z_{\overleftarrow{q}} = \sum_{\overleftarrow{y}} \overleftarrow{p}_\theta(\overleftarrow{\boldsymbol{y}}, \boldsymbol{x}) exp\{-\boldsymbol{\lambda}\cdot\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})\},$$

such that $\mathbb{E}_{q_{\boldsymbol{\lambda}}}[\phi_{i,j}(\boldsymbol{x}, \boldsymbol{y})] = 0$. In the E-step of EM-algorithm, we employ $q_{\boldsymbol{\lambda}}$ instead of p_θ to accumulate fractional counts for its use in the M-step. $\boldsymbol{\lambda}$ is efficiently estimated by the gradient ascent for each bilingual sentence $\boldsymbol{x}$. Note that posterior regularization is performed during parameter estimation, and not during testing.

3 Posterior Regularization with Frequency Constraint

The symmetric constraint method represented in Equation (3) assumes a strong one-to-one relation for any word, and does not take into account the divergence in language pairs. For linguistically different language pairs, such as Japanese-English, content words may be easily aligned one-to-one, but function words are not always aligned together. In addition, Japanese is a pro-drop language which can easily violate the symmetric constraint when proper nouns in the English side have to be aligned with a "null" word. In addition, low frequency words may cause unreliable estimates for adjusting the weighing parameters $\boldsymbol{\lambda}$.

In order to solve the problem, we improve Ganchev's symmetric constraint so that it can consider the difference between content words and function words in each language. In particular, we follow the frequency-based idea of Setiawan et al. (2007) that discriminates content words and function words by their frequencies. We propose constraint features that take into account the difference between content words and function words, determined by a frequency threshold.

3.1 Mismatching constraint

First, we propose a mismatching constraint that penalizes word alignment between content words and function words by decreasing the corresponding posterior probabilities.

[1] No alignment is represented by alignment into a special token "null".

The constraint is represented as *f2c* (*function to content*) constraint:

$$\phi^{\text{f2c}}_{i,j}(\boldsymbol{x},\boldsymbol{y}) = \tag{6}$$
$$\begin{cases} +1 & (\overrightarrow{\boldsymbol{y}} \subset \boldsymbol{y}) \cap (\overrightarrow{y}_i = j) \cap ((x^t_i \in \mathcal{C}^t \cap x^s_j \in \mathcal{F}^s) \\ & \cup (x^t_i \in \mathcal{F}^t \cap x^s_j \in \mathcal{C}^s)) \cap (\delta_{i,j}(\boldsymbol{x},\boldsymbol{y}) > 0), \\ 0 & (\overleftarrow{\boldsymbol{y}} \subset \boldsymbol{y}) \cap (\overleftarrow{y}_j = i) \cap ((x^t_i \in \mathcal{C}^t \cap x^s_j \in \mathcal{F}^s) \\ & \cup (x^t_i \in \mathcal{F}^t \cap x^s_j \in \mathcal{C}^s)) \cap (\delta_{i,j}(\boldsymbol{x},\boldsymbol{y}) > 0), \\ 0 & (\overrightarrow{\boldsymbol{y}} \subset \boldsymbol{y}) \cap (\overrightarrow{y}_i = j) \cap ((x^t_i \in \mathcal{C}^t \cap x^s_j \in \mathcal{F}^s) \\ & \cup (x^t_i \in \mathcal{F}^t \cap x^s_j \in \mathcal{C}^s)) \cap (\delta_{i,j}(\boldsymbol{x},\boldsymbol{y}) < 0), \\ -1 & (\overleftarrow{\boldsymbol{y}} \subset \boldsymbol{y}) \cap (\overleftarrow{y}_j = i) \cap ((x^t_i \in \mathcal{C}^t \cap x^s_j \in \mathcal{F}^s) \\ & \cup (x^t_i \in \mathcal{F}^t \cap x^s_j \in \mathcal{C}^s)) \cap (\delta_{i,j}(\boldsymbol{x},\boldsymbol{y}) < 0). \end{cases}$$

where $\delta_{i,j}(\boldsymbol{x},\boldsymbol{y}) = \overrightarrow{p}_\theta(i,j|\boldsymbol{x}) - \overleftarrow{p}_\theta(i,j|\boldsymbol{x})$ is the difference in the posterior probabilities between the source-to-target and the target-to-source alignment. $\mathcal{C}^s$ and $\mathcal{C}^t$ represent content words in the source sentence and target sentence, respectively. Similarly, $\mathcal{F}^s$ and $\mathcal{F}^t$ are function words in the source and target sentence, respectively. Intuitively, when there exists a mismatch in content word and function word for a word pair (i,j), the constraint function returns a non-zero value for the model with the highest posterior probability. When coupled with the constraint such that the expectation of the feature value is zero, the constraint function decreases the posterior probability of the highest direction and discourages agreement with each other.

Note that when this constraint is not fired, we fall back to the constraint function in Equation (3) for each word pair.

3.2 Matching constraint

In contrast to the mismatching constraint, our second constraint function rewards alignment for *function to function* word matching, namely *f2f*. The f2f constraint function is defined as follows:

$$\phi^{\text{f2f}}_{i,j}(\boldsymbol{x},\boldsymbol{y}) = \tag{7}$$
$$\begin{cases} +1 & (\overrightarrow{\boldsymbol{y}} \subset \boldsymbol{y}) \cap (\overrightarrow{y}_i = j) \cap \\ & (x^t_i \in \mathcal{F}^t \cap x^s_j \in \mathcal{F}^s) \cap (\delta_{i,j}(\boldsymbol{x},\boldsymbol{y}) > 0), \\ 0 & (\overleftarrow{\boldsymbol{y}} \subset \boldsymbol{y}) \cap (\overleftarrow{y}_j = i) \cap \\ & (x^t_i \in \mathcal{F}^t \cap x^s_j \in \mathcal{F}^s) \cap (\delta_{i,j}(\boldsymbol{x},\boldsymbol{y}) > 0), \\ 0 & (\overrightarrow{\boldsymbol{y}} \subset \boldsymbol{y}) \cap (\overrightarrow{y}_i = j) \cap \\ & (x^t_i \in \mathcal{F}^t \cap x^s_j \in \mathcal{F}^s) \cap (\delta_{i,j}(\boldsymbol{x},\boldsymbol{y}) < 0), \\ -1 & (\overleftarrow{\boldsymbol{y}} \subset \boldsymbol{y}) \cap (\overleftarrow{y}_j = i) \cap \\ & (x^t_i \in \mathcal{F}^t \cap x^s_j \in \mathcal{F}^s) \cap (\delta_{i,j}(\boldsymbol{x},\boldsymbol{y}) < 0). \end{cases}$$

This constraint function returns a non-zero value for a word pair (i,j) when they are function words. As a result, the pair of function words are encouraged to agree with each other, but not other pairs. The *content to content* word matching function *c2c* can be defined similarly by replacing F^s and F^t by C^s and C^t, respectively. Likewise, the *function to content* word matching function *f2c* is defined by considering the matching of content words and function words in two languages. As noted in the mismatch function, when no constraint is fired, we fall back to Eq (3) for each word pair.

4 Experiment

4.1 Experimental Setup

The data sets used in our experiments are the French-English Hansard Corpus, and two data sets for Japanese-English tasks: the Kyoto free translation task (KFTT) and NTCIR10. The Hansard Corpus consists of parallel texts drawn from official records of the proceedings of the Canadian Parliament. The KFTT (Neubig, 2011) is derived from Japanese Wikipedia articles related to Kyoto, which is professionally translated into English. NTCIR10 comes from patent data employed in a machine translation shared task (Goto et al., 2013). The statistics of these data are presented in Table 1.

Sentences of over 40 words on both source and target sides are removed for training alignment models. We used a word alignment toolkit cicada [2] for training the IBM Model 4 with our proposed methods. Training is bootstrapped from IBM Model 1, followed by HMM and IBM Model 4. When generating the final bidirectional word alignment, we use a grow-diag-final heuristic for the Japanese-English tasks and an intersection heuristic in the French-English task, judged by preliminary studies.

Following Bisazza and Federico (2012), we automatically decide the threshold for word frequency to discriminate between content words and function words. Specifically, the threshold is determined by the ratio of highly frequent words. The threshold th is the maximum frequency that satisfies the following equation:

$$\frac{\sum_{w \in (freq(w) > th)} freq(w)}{\sum_{w \in all} freq(w)} > r. \tag{8}$$

Here, we empirically set $r = 0.5$ by preliminary studies. This method is based on the intuition that content words and function words exist in a document at a constant rate.

4.2 Word alignment evaluation

We measure the impact of our proposed methods on the quality of word alignment measured

[2] https://github.com/tarowatanabe/cicada

Table 1: The statistics of the data sets

			hansard		kftt		NTCIR10	
			French	English	Japanese	English	Japanese	English
train		sentence	1.13M		329.88K		2.02M	
		word	23.3M	19.8M	6.08M	5.91M	53.4M	49.4M
		vocabulary	78.1K	57.3K	114K	138K	114K	183K
dev		sentence			1.17K		2K	
		word			26.8K	24.3K	73K	67.3K
		vocabulary			4.51K	4.78K	4.38K	5.04K
test	WA	sentence	447		582			
		word	7.76K	7.02K	14.4K	12.6K		
		vocabulary	1,92K	1.69K	2.57K	2.65K		
	TR	sentence			1.16K		8.6K	
		word			28.5K	26.7K	334K	310K
		vocabulary			4.91K	4.57K	10.4K	12.7K

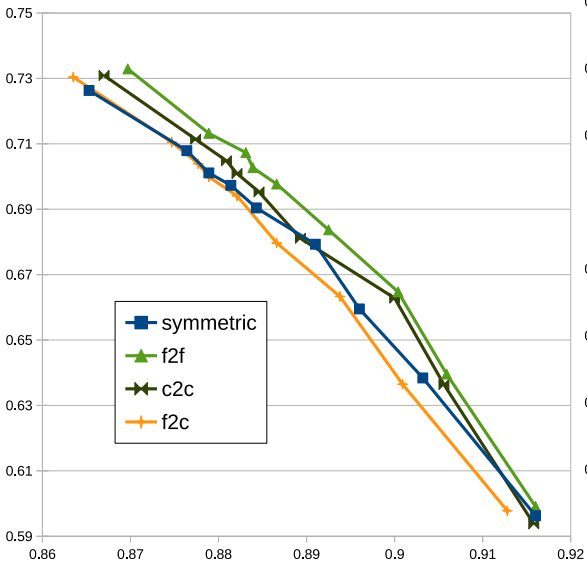


Figure 1: Precision Recall graph in Hansard French-English

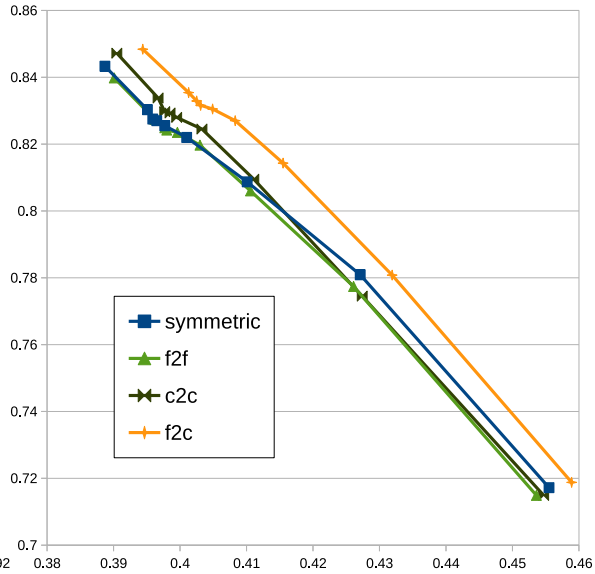


Figure 2: Precision Recall graph in KFTT

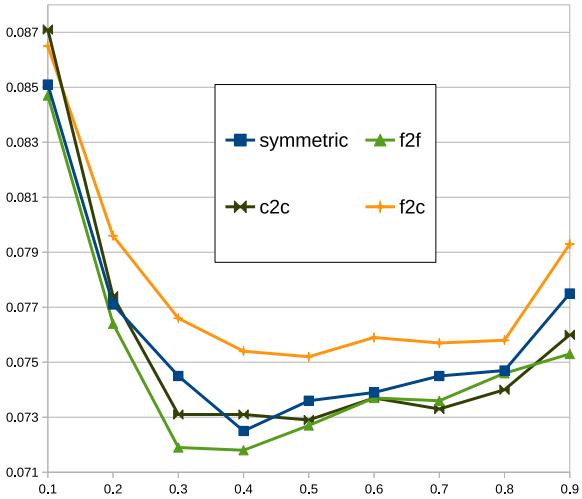


Figure 3: AER in Hansard French-English

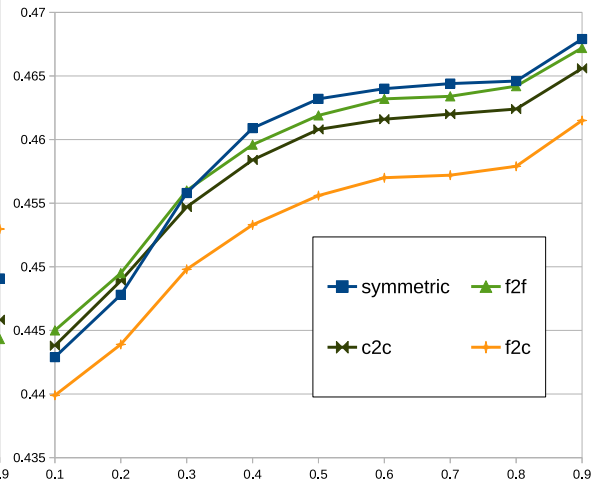


Figure 4: AER in KFTT

Table 2: Results of word alignment evaluation with the heuristics-based method (GDF)

	KFTT				Hansard (French-English)			
method	precision	recall	AER	F	precision	recall	AER	F
symmetric	0.4595	0.5942	48.18	0.5182	0.7029	0.8816	*7.29*	0.7822
f2f	**0.4633**	*0.5997*	**47.73**	**0.5227**	**0.7042**	*0.8851*	*7.29*	*0.7844*
c2c	0.4606	0.5964	48.02	0.5198	0.7001	0.8816	7.34	0.7804
f2c	*0.4630*	**0.5998**	*47.74*	*0.5226*	*0.7037*	**0.8871**	**7.10**	**0.7848**

by AER and F-measure (Och and Ney, 2003). Since there exists no distinction for sure-possible alignments in the KFTT data, we use only sure alignment for our evaluation, both for the French-English and the Japanese-English tasks. Table 2 summarizes our results.

The baseline method is symmetric constraint (Ganchev et al., 2010) shown in Table 2. The numbers in bold and in italics indicate the best score and the second best score, respectively. The differences between f2f,f2c and baseline in KFTT are statistically significant at $p < 0.05$ using the sign-test, but in hansard corpus, there exist no significant differences between the baseline and the proposed methods. In terms of F-measure, it is clear that the f2f method is the most effective method in KFTT, and both f2f and f2c methods exceed the original posterior regularized model of Ganchev et al. (2010).

We also compared these methods with filtering methods (Liang et al., 2006), in addition to heuristic methods. We plot precision/recall curves and AER by varying the threshold between 0.1 and 0.9 with 0.1 increments. From Figures, it can be seen that our proposed methods are superior to the baseline in terms of both precision-recall and AER.

4.3 Translation evaluation

Next, we performed a translation evaluation, measured by BLEU (Papineni et al., 2002). We compared the grow-diag-final and filtering method (Liang et al., 2006) for creating phrase tables. The threshold for the filtering factor was set to 0.1 which was the best setting in the word alignment experiment in section 4.2 under KFTT. From the English side of the training data, we trained a word using the 5-gram model with SRILM (Stolcke and others, 2002). "Moses" toolkit was used as a decoder (Koehn et al., 2007) and the model parameters were tuned by k-best MIRA (Cherry and Foster, 2012). In order to avoid tuning instability, we evaluated the average of five runs (Hopkins and May, 2011). The results are summarized in Table 3. Our proposed methods achieved large gains in NTCIR10 task with the filtered method, but observed no gain in the KFTT with the filtered method. In NTCIR10 task with GDF, the gain in BLEU was smaller than that of KFTT. We calculate p-values and the difference between symmetric and c2c (the most effective proposed constraint) are lower than 0.05 in kftt with GDF and NTCIR10 with filtered method. There seems to be no clear tendency in the improved alignment qualities and the translation qualities, as shown in numerous previous studies (Ganchev et al., 2008).

Table 3: Results of translation evaluation

	KFTT		NTCIR10	
	GDF	Filtered	GDF	Filtered
symmetric	19.06	**19.28**	*28.3*	29.71
f2f	*19.15*	19.17	**28.36**	*29.74*
c2c	**19.26**	19.02	**28.36**	**29.92**
f2c	18.91	*19.20*	**28.36**	29.67

5 Conclusion

In this paper, we proposed new constraint functions under the posterior regularization framework. Our constraint functions introduce a fine-grained agreement constraint considering the frequency of words, a assuming that the high frequency words correspond to function words whereas the less frequent words may be treated as content words, based on the previous work of Setiawan et al. (2007). Experiments on word alignment tasks showed better alignment qualities measured by F-measure and AER on both the Hansard task and KFTT. We also observed large gain in BLEU, 0.2 on average, when compared with the previous posterior regularization method under NTCIR10 task.

As our future work, we will investigate more precise methods for deciding function words and content words for better alignment and translation qualities.

References

Arianna Bisazza and Marcello Federico. 2012. Cutting the long tail: Hybrid language models for translation style adaptation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 439–448. Association for Computational Linguistics.

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

Yin-Wen Chang, Alexander M. Rush, John DeNero, and Michael Collins. 2014. A constrained viterbi relaxation for bidirectional word alignment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1481–1490, Baltimore, Maryland, June. Association for Computational Linguistics.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436. Association for Computational Linguistics.

John DeNero and Klaus Macherey. 2011. Model-based aligner combination using dual decomposition. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 420–429, Portland, Oregon, USA, June. Association for Computational Linguistics.

Kuzman Ganchev, João V. Graça, and Ben Taskar. 2008. Better alignments = better translations? In *Proceedings of ACL-08: HLT*, pages 986–993, Columbus, Ohio, June. Association for Computational Linguistics.

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 99:2001–2049.

Isao Goto, Ka Po Chow, Bin Lu, Eiichiro Sumita, and Benjamin K Tsou. 2013. Overview of the patent machine translation task at the ntcir-10 workshop. In *Proceedings of the 10th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access, NTCIR-10*.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 104–111, New York City, USA, June. Association for Computational Linguistics.

E. Matusov, R. Zens, and H. Ney. 2004. Symmetric Word Alignments for Statistical Machine Translation. In *Proceedings of COLING 2004*, pages 219–225, Geneva, Switzerland, August 23–27.

Graham Neubig. 2011. The Kyoto free translation task. http://www.phontron.com/kftt.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Hendra Setiawan, Min-Yen Kan, and Haizhou Li. 2007. Ordering phrases with function words. In *Proceedings of the 45th annual meeting on association for computational linguistics*, pages 712–719. Association for Computational Linguistics.

Andreas Stolcke et al. 2002. Srilm-an extensible language modeling toolkit. In *INTERSPEECH*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics.

Asymmetric Features of Human Generated Translation

Sauleh Eetemadi
Michigan State University, East Lansing, MI
Microsoft Research, Redmond, WA
saulehe@microsoft.com

Kristina Toutanova
Microsoft Research
Redmond, WA
kristout@microsoft.com

Abstract

Distinct properties of translated text have been the subject of research in linguistics for many year (Baker, 1993). In recent years computational methods have been developed to empirically verify the linguistic theories about translated text (Baroni and Bernardini, 2006). While many characteristics of translated text are more apparent in comparison to the original text, most of the prior research has focused on monolingual features of translated and original text. The contribution of this work is introducing bilingual features that are capable of explaining differences in translation direction using localized linguistic phenomena at the phrase or sentence level, rather than using monolingual statistics at the document level. We show that these bilingual features outperform the monolingual features used in prior work (Kurokawa et al., 2009) for the task of classifying translation direction.

1 Introduction

It has been known for many years in linguistics that translated text has distinct patterns compared to original or authored text (Baker, 1993). The term "Translationese" is often used to refer to the characteristics of translated text. Patterns of Translationese can be categorized as follows (Volansky et al., 2013):

1. **Simplification**: The process of translation is often coupled with a simplification process at several levels. For example, there tends to be less lexical variety in translated text and rare words are often avoided.
2. **Explicitation**: Translators often have to be more explicit in their translations due to lack of the cultural context that speakers of the source language have. Another manifestation of this pattern is making arguments more explicit which can be observed in the heavy use of cohesive markers like "therefore" and "moreover" in translated text (Koppel and Ordan, 2011).
3. **Normalization**: Translated text often contains more formal and repeating language.
4. **Interference**: A translator is likely to produce a translation that is structurally and grammatically closer to the source text or their native language.

In Figure 1 the size of a word in the "Translated" section is proportional to the difference between the frequency of the word in original and in the translated text (Fellows, 2013). For example, it is apparent that the word "the" is over-represented in translated English as noted by other research (Volansky et al., 2013). In addition, cohesive markers are clearly more common in translated text.

In the past few years there has been work on machine learning techniques for identifying Translationese. Standard machine learning algorithms like SVMs (Baroni and Bernardini, 2006) and Bayesian Logistic Regression (Koppel and Ordan, 2011) have been employed to train classifiers for one of the following tasks:

i. Given a chunk of text in a specific language, classify it as "Original" or "Translated".
ii. Given a chunk of translated text, predict the source language of the translation.
iii. Given a text chunk pair and their languages, predict the direction of translation.

There are two stated motivations for the tasks above: first, empirical validation of linguistic theories about Translationese (Volansky et al., 2013), and second, improving statistical machine translation by leveraging the knowledge of the translation direction in training and test data (Lember-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 159–164,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

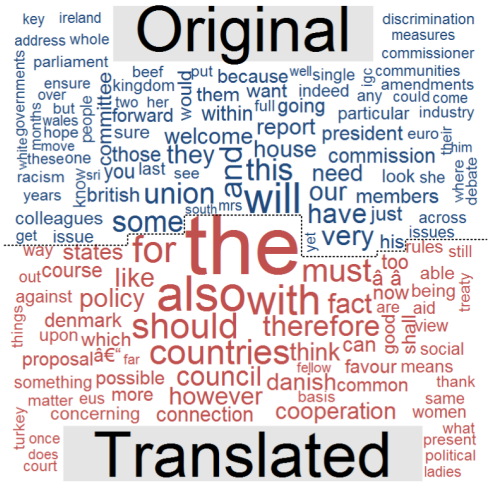


Figure 1: EuroParl Word Cloud Data Visualization (Translated vs Original) [1]

sky et al., 2012a; Lembersky et al., 2013; Lembersky et al., 2012b). Few parallel corpora including a customized version of EuroParl (Islam and Mehler, 2012) and a processed version of Hansard (Kurokawa et al., 2009) are labeled for translated versus original text. Using these limited resources, it has been shown that taking the translation direction into account when training a statistical machine translation system can improve translation quality (Lembersky et al., 2013). However, improving statistical machine translation using translation direction information has been limited by several factors.

1. **Limited Labeled Data**: The amount of labeled data is limited by language and domain and therefore by itself is not enough to make a significant improvement in statistical machine translation.
2. **Cross-Domain Scalability**: Current methods of Translationese detection do not scale across different corpora. For example, a classifier trained on EuroParl corpus (Koehn, 2005) had in-domain accuracy of 92.7% but out-of-domain accuracy of 64.8% (Koppel and Ordan, 2011).
3. **Text Chunk Size**: The reported high accuracy of Translationese detection is based on relatively large (approximately 1500 tokens) text chunks (Koppel and Ordan, 2011). When similar tasks are performed at the sentence level the accuracy drops by 15 percentage points or more (Kurokawa et al., 2009). Figure 2 shows how detection accuracy drops with the reduction of the input text chunk size. Since parallel data are often available at the sentence level or small chunks of text, existing detection methods aren't suitable for this type of data.

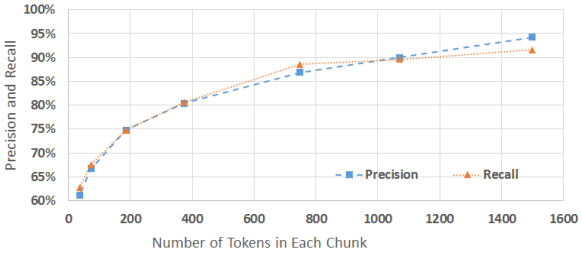


Figure 2: Effects of Chunk Size on Translationese Detection Accuracy[2]

Motivated by these limitations, in this work we focus on improving sentence-level classification accuracy by using non-domain-specific bilingual features at the sentence level. In addition to improving accuracy, these fine-grained features may be better able to confirm existing theories or discover new linguistic phenomena that occur in the translation process. We use a fast linear classifier trained with online learning, Vowpal Wabbit (Langford et al., 2007). The Hansard French-English dataset (Kurokawa et al., 2009) is used for training and test data in all experiments.

2 Related Work

While distinct patterns of Translationese have been studied widely in the past, the work of Baroni and Bernardini (2006) is the first to introduce a computational method for detecting Translationese with high accuracy. Prior work has shown in-domain accuracy can be very high at the chunk-level if fully lexicalized features are used (Volansky et al., 2013), but then the phenomena learned are clearly not generalizable across domains. For example, in Figure 1, it can be observed that content words like "commission", "council" or "union" can be used effectively for classification while they do not capture any general linguistic phenomena and are unlikely to scale

[1]This word cloud was created using the *wordcloud* and *tm* **R** packages (Fellows, 2013) from EuroParl parallel data annotated for translation direction (Islam and Mehler, 2012) obtained from http://www.hucompute.org/ressourcen/corpora/56.

[2]This is a reproduction of the results of Koppel and Ordan (2011) using function word frequencies as features for a logistic regression classifier. Based on the description of how text chunks were created, the results of the paper (92.7% accuracy) are based on text chunk sizes of approximately 1500 tokens.

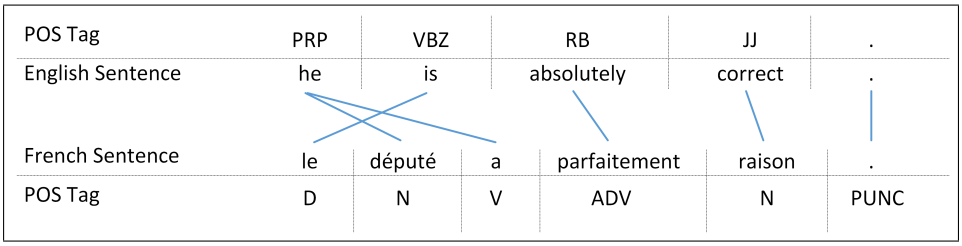


Figure 3: POS Tagged Aligned Sentence Pairs

to other corpora. This is also confirmed by an average human performance of 72.7% precision with 82.1% recall on a similar task where the test subjects were not familiar with the domain and were not able to use domain-specific lexical features (Baroni and Bernardini, 2006). A more general feature set still with high in-domain accuracy is POS tags with lexicalization of function words (Baroni and Bernardini, 2006; Kurokawa et al., 2009). We build on this feature set and explore bilingual features.

The only work to consider features of the two parallel chunks (one original, one translated) is the work of Kurokawa et al. (2009). They simply used the union of the n-gram mixed-POS[3] features of the two sides; these are monolingual features of the original and translated text and do not look at translation phenomena directly. Their work is also the only work to look at sentence level detection accuracy and report 15 percentage points drop in accuracy when going from chunk level to sentence level classification.

3 Bilingual Features for Translation Direction Classification

We are interested in learning common localized linguistic phenomena that occur during the translation process when translating in one direction but not the other.

3.1 POS Tag MTUs

Minimal translation units (MTUs) for a sentence pair are defined as pairs of source and target word sets that satisfy the following conditions (Quirk and Menezes, 2006).

1. No alignment links between distinct MTUs.
2. MTUs are not decomposable into smaller MTUs without violating the previous rule.

We use POS tags to capture linguistic structures and MTUs to map linguistic structures of the two languages. To obtain POS MTUs from a parallel corpus, first, the parallel corpus is word aligned. Next, the source and target side of the corpus are tagged independently. Finally, words are replaced with their corresponding POS tag in word-aligned sentence pairs. MTUs were extracted from the POS tagged word-aligned sentence pairs from left to right and listed in source order. Unigram, bi-gram, and higher order n-gram features were built over this sequence of POS MTUs. For example, for the sentence pair in Figure 3, the following POS MTUs will be extracted: `VBZ⇒D, PRP⇒(N,V), RB⇒ADV, JJ⇒N, .⇒PUNC`.

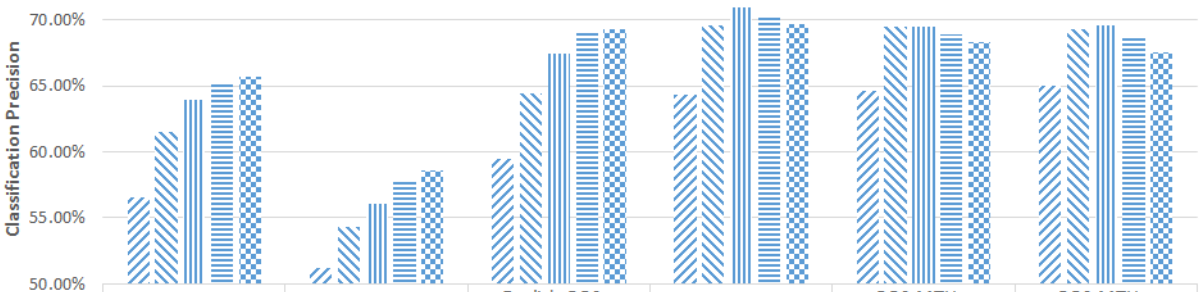
3.2 Distortion

In addition to the mapping of linguistic structures, another interesting phenomenon is the reordering of linguistic structures during translation. One hypothesis is that when translating from a fixed-order to a free-order language, the order of the target will be very influenced by the source (almost monotone translation), but when translating into a fixed order language, more re-ordering is required to ensure grammaticality of the target. To capture this pattern we add distortion to POS Tag MTU features. We experiment with absolute distortion (word position difference between source and target of a link) as well as HMM distortion (word position difference between the target of a link and the target of the previous link). We bin the distortions into three bins: "$= 0$", "> 0" and "< 0", to reduce sparsity.

4 Experimental Setup

For the translation direction detection task explained in section 1, we use a fast linear classifier trained with online learning, Vowpal Wabbit (Langford et al., 2007). Training data and classification features are explained in section 4.1 and 4.2.

[3]Only replacing content words with their POS tags while leaving function words as is.



	English-POS	French-POS	English-POS + French-POS	POS-MTU	POS-MTU + Absolute Distortion	POS-MTU + HMM Distorition
1-gram	56.59%	51.20%	59.46%	64.37%	64.58%	65.01%
2-gram	61.48%	54.27%	64.43%	69.60%	69.47%	69.29%
3-gram	63.93%	56.04%	67.43%	70.95%	69.45%	69.57%
4-gram	65.28%	57.72%	69.01%	70.29%	68.91%	68.58%
5-gram	65.71%	58.61%	69.32%	69.63%	68.36%	67.55%

Figure 4: Sentence level translation direction detection precision using different features with n-gram lengths of 1 through 5.

4.1 Data

For this task we require a parallel corpus with sentence pairs available in both directions (sentences authored in language A and then translated to language B and vice versa). While the customized version of EuroParl (Islam and Mehler, 2012) contains sentence pairs for many language pairs, none of the language pairs have sentence pairs available in both directions (e.g., it does contain sentences authored in English and translated into French but not vice versa). The Canadian Hansard corpus on the other hand fits the requirement as it has 742,408 sentence pairs translated from French to English and 2,203,504 sentences pairs that were translated from English to French (Kurokawa et al., 2009). We use the Hansard data for training classifiers. For training the HMM word alignment model used to define features, we use a larger set of ten billion words of parallel text from the WMT English-French corpus.

4.2 Preprocessing and Feature Extraction

We used a language filter[4], deduplication filter[5] and length ratio filter to clean the data. After filtering we were left with 1,890,603 English-French sentence pairs and 640,117 French-English sentence pairs. The Stanford POS tagger (Toutanova and Manning, 2000) was used to tag the English and the French sides of the corpus. The HMM alignment model (Vogel et al., 1996) trained on WMT data was used to word-align the Hansard corpus while replacing words with their corresponding POS tags. Due to differences in word breaking between the POS tagger tool and our word alignment tool there were some mismatches. For simplicity we dropped the entire sentence pair whenever a token mismatch occurred. This left us with 401,569 POS tag aligned sentence pairs in the French to English direction and 1,184,702 pairs in the other direction. We chose to create a balanced dataset and reduced the number of English-French sentences to 401,679 with 20,000 sentence pairs held out for testing in each direction.

5 Results

The results of our experiments on the translation direction detection task are listed in Table 4. We would like to point out several results from the table. First, when using only unigram features, the highest accuracy is achieved by the "POS-MTU + HMM Distortion" feature, which uses POS minimal translation units together with distortion. The highest accuracy overall if obtained by a "POS-MTU" trigram model, showing the advantage of bilingual features over prior work using only a union of monolingual features (reproduced by the "English-POS + French-POS" configuration). While higher order features generally show better in-domain accuracy, the advantage of low-order bilingual features might be even higher in cross-domain classification.

[4]A character n-gram language model is used to detect the language of source and target side text and filter them out if they do not match their annotated language.

[5]Duplicate sentences pairs are filtered out.

[6]For description of English POS tags see (Marcus et al., 1993) and (Abeillé et al., 2003) for French

	POS MTU (E⇒F)	FE#	EF#	Example
1	NNPS⇒(N,C)	336	12	quebecers(NNPS) ⇒ québécoises(N) et(C) des québécois
2	IN⇒(CL,V)	69	1027	a few days ago(IN) ⇒ il y(CL) a(V) quelques
3	PRP⇒(N,V)	18	663	he(PRP) is ⇒ le député(N) à(V)
4	(NNP,POS)⇒A	155	28	quebec(NNP) 's(POS) history ⇒ histoire québécoises(A)
5	(FW,FW)⇒ADV	7	195	pro(FW) bono(FW) work ⇒ bénévolement(ADV) travailler
6	(RB,MD)⇒V	2	112	money alone(RB) could(MD) solve ⇒ argent suffirait(V) à résoudre

Table 1: POS MTU features with highest weight. FE# indicates the number of times this feature appeared when translating from French to English.[6]

6 Analysis

An interesting aspect of this work is that it is able to extract features that can be linguistically interpreted. Although linguistic analysis of these features is outside the scope of this work, we list POS MTU features with highest positive or negative weights in Table 1. Although the top feature, NNPS⇒(N,C)[7], in this context is originating from a common phrase used by French speaking members of the Canadian Parliament, québécoises et des québécois, it does highlight an underlying linguistic phenomenon that is not specific to the Canadian Parliament. When translating a plural noun from English to French it is likely that only the masculine form of the noun appears, while if it was authored in French with both forms of the nouns, a single plural noun would appear in English as English doesn't have masculine and feminine forms of the word. A more complete form of this feature would have been NNPS⇒(N,C,N), but since word alignment models, in general, discourage one-to-many alignments, the extracted MTU only covers the first noun and conjunction.

7 Conclusion and Future Work

In this work we introduce new features for translation direction detection that leverage word alignment, source POS and target POS in the form of POS MTUs. POS MTUs are a powerful tool for capturing linguistic interactions between languages during the translation process. Since POS MTUs are not lexical features they are more likely to scale across corpora and domains compared to lexicalized features. Although most of the high weight POS MTU features used in classification (Table 1) are not corpus specific, unfortunately, due to lack of training data in multiple domains, experiments were not run to validate this claim. In future work, we intend to obtain training data from multiple domains that enables us to verify cross-domain scalability of POS-MTUs. In addition, observing linguistic phenomena that occur in one translation direction but not the other can be very informative in improving statistical machine translation quality. Another future direction for this work is leveraging sentence level translation direction detection to improve statistical machine translation output quality. Finally, further investigation of the linguistic interpretation of individual feature that are most discriminating between opposite translation directions can lead to discovery of new linguistic phenomena that occur during the translation process.

[7] NNPS: Plural Noun, N: Noun, C:Conjunction

Acknowledgement

The authors would like to thank Lee Schwartz for analyzing classification features and providing linguistic insight for them. We would like to also acknowledge the thoughtful comments and detailed feedback of the reviewers which helped us improve the paper.

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for french. In Anne Abeillé, editor, *Treebanks*, volume 20 of *Text, Speech and Language Technology*, pages 165–187. Springer Netherlands.

Mona Baker. 1993. Corpus linguistics and translation studies: Implications and applications. *Text and technology: in honour of John Sinclair*, 233:250.

Marco Baroni and Silvia Bernardini. 2006. A new approach to the study of translationese: Machine-learning the difference between original and translated text. *Literary and Linguistic Computing*, 21(3):259–274.

Ian Fellows, 2013. *wordcloud: Word Clouds*. R package version 2.4.

Zahurul Islam and Alexander Mehler. 2012. Customization of the europarl corpus for translation studies. In *LREC*, page 2505–2510.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Moshe Koppel and Noam Ordan. 2011. Translationese and its dialects. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, page 1318–1326. Association for Computational Linguistics.

David Kurokawa, Cyril Goutte, and Pierre Isabelle. 2009. Automatic detection of translated text and its impact on machine translation. *Proceedings. MT Summit XII, The twelfth Machine Translation Summit International Association for Machine Translation hosted by the Association for Machine Translation in the Americas*.

J Langford, L Li, and A Strehl, 2007. *Vowpal wabbit online learning project*.

Gennadi Lembersky, Noam Ordan, and Shuly Wintner. 2012a. Adapting translation models to translationese improves SMT. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, page 255–265. Association for Computational Linguistics.

Gennadi Lembersky, Noam Ordan, and Shuly Wintner. 2012b. Language models for machine translation: Original vs. translated texts. *Computational Linguistics*, 38(4):799–825.

Gennadi Lembersky, Noam Ordan, and Shuly Wintner. 2013. Improving statistical machine translation by adapting translation models to translationese.

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

Chris Quirk and Arul Menezes. 2006. Do we need phrases?: Challenging the conventional wisdom in statistical machine translation. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 9–16, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, EMNLP '00, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics.

Vered Volansky, Noam Ordan, and Shuly Wintner. 2013. On the features of translationese. *Literary and Linguistic Computing*, page fqt031.

Syntax-Augmented Machine Translation using Syntax-Label Clustering

Hideya Mino, Taro Watanabe and **Eiichiro Sumita**
National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, JAPAN
{hideya.mino, taro.watanabe, eiichiro.sumita}@nict.go.jp

Abstract

Recently, syntactic information has helped significantly to improve statistical machine translation. However, the use of syntactic information may have a negative impact on the speed of translation because of the large number of rules, especially when syntax labels are projected from a parser in syntax-augmented machine translation. In this paper, we propose a syntax-label clustering method that uses an exchange algorithm in which syntax labels are clustered together to reduce the number of rules. The proposed method achieves clustering by directly maximizing the likelihood of synchronous rules, whereas previous work considered only the similarity of probabilistic distributions of labels. We tested the proposed method on Japanese-English and Chinese-English translation tasks and found order-of-magnitude higher clustering speeds for reducing labels and gains in translation quality compared with previous clustering method.

1 Introduction

In recent years, statistical machine translation (SMT) models that use syntactic information have received significant research attention. These models use syntactic information on the source side (Liu et al., 2006; Mylonakis and Sima'an, 2011), the target side (Galley et al., 2006; Huang and Knight, 2006) or both sides (Chiang, 2010; Hanneman and Lavie, 2013) produce syntactically correct translations. Zollmann and Venugopal (2006) proposed syntax-augmented MT (SAMT), which is a MT system that uses syntax labels of a parser. The SAMT grammar directly encodes syntactic information into the synchronous context-free grammar (SCFG) of Hiero (Chiang, 2007), which relies on two nonterminal labels. One problem in adding syntax labels to Hiero-style rules is that only partial phrases are assigned labels. It is common practice to extend labels by using the idea of combinatory categorial grammar (CCG) (Steedman, 2000) on the problem. Although this extended syntactical information may improve the coverage of rules and syntactic correctness in translation, the increased grammar size causes serious speed and data-sparseness problems. To address these problems, Hanneman and Lavie (2013) coarsen syntactic labels using the similarity of the probabilistic distributions of labels in synchronous rules and showed that performance improved.

In the present work, we follow the idea of label-set coarsening and propose a new method to group syntax labels. First, as an optimization criterion, we use the logarithm of the likelihood of synchronous rules instead of the similarity of probabilistic distributions of syntax labels. Second, we use exchange clustering (Uszkoreit and Brants, 2008), which is faster than the agglomerative-clustering algorithm used in the previous work. We tested our proposed method on Japanese-English and Chinese-English translation tasks and observed gains comparable to those of previous work with similar reductions in grammar size.

2 Syntax-Augmented Machine Translation

SAMT is an instance of SCFG $\mathcal{G}$, which can be formally defined as

$$\mathcal{G} = (\mathcal{N}, S, \mathcal{T}_\sigma, \mathcal{T}_\tau, \mathcal{R})$$

where $\mathcal{N}$ is a set of nonterminals, $S \in \mathcal{N}$ is a start label, $\mathcal{T}_\sigma$ and $\mathcal{T}_\tau$ are the source- and target-side terminals, and $\mathcal{R}$ is a set of synchronous rules. Each synchronous rule in $\mathcal{R}$ takes the form

$$X \rightarrow \langle \alpha, \beta, \sim \rangle$$

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 165–171,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

where $X \in \mathcal{N}$ is a nonterminal, $\alpha \in (\mathcal{N} \cup \mathcal{T}_\sigma)*$ is a sequence of nonterminals or source-side terminals, and $\beta \in (\mathcal{N} \cup \mathcal{T}_\tau)*$ is a sequence of nonterminals or target-side terminals. The number $\#NT(\alpha)$ of nonterminals in α is equal to the number $\#NT(\beta)$ of nonterminals in β, and $\sim$: $\{1, ..., \#NT(\alpha)\} \rightarrow \{1, ..., \#NT(\beta)\}$ is a one-to-one mapping from nonterminals in α to nonterminals in β. For each synchronous rule, a nonnegative real-value weight $w(X \rightarrow \langle \alpha, \beta, \sim \rangle)$ is assigned and the sum of the weights of all rules sharing the same left-hand side in a grammar is unity.

Hierarchical phrase-based SMT (Hiero) (Chiang, 2007) translates by using synchronous rules that only have two nonterminal labels X and S but have no linguistic information. SAMT augments the Hiero-style rules with syntax labels from a parser and extends these labels based on CCG. Although the use of extended syntax labels may increase the coverage of rules and improve the potential for syntactically correct translations, the growth of the nonterminal symbols significantly affects the speed of decoding and causes a serious data-sparseness problem.

To address these problems, Hanneman and Lavie (2013) proposed a label-collapsing algorithm, in which syntax labels are clustered by using the similarity of the probabilistic distributions of clustered labels in synchronous rules. First, Hanneman and Lavie defined the label-alignment distribution as

$$P(s|t) = \frac{\#(s,t)}{\#(t)} \tag{1}$$

where $\mathcal{N}_\sigma$ and $\mathcal{N}_\tau$ are the source- and target-side nonterminals in synchronous rules, $s \in \mathcal{N}_\sigma$ and $t \in \mathcal{N}_\tau$ are syntax labels from the source and target sides, $\#(s,t)$ denotes the number of left-hand-side label pairs, and $\#(t)$ denotes the number of target-side labels. Second, for each target-side label pair (t_i, t_j), we calculate the total distance d of the absolute differences in the likelihood of labels that are aligned to a source-side label s:

$$d(t_i, t_j) = \sum_{s \in \mathcal{N}_\sigma} |P(s|t_i) - P(s|t_j)| \tag{2}$$

Next, the closest syntax-label pair of $\hat{t}$ and $\hat{t}'$ is combined into a new single label. The agglomerative clustering is applied iteratively until the number of the syntax labels reaches a given value.

The clustering of Hanneman and Lavie proved successful in decreasing the grammar size and providing a statistically significant improvement in translation quality. However, their method relies on an agglomerative clustering with a worst-case time complexity of $O(|\mathcal{N}|^2 \log |\mathcal{N}|)$. Also, clustering based on label distributions does not always imply higher-quality rules, because it does not consider the interactions of the nonterminals on the left-hand side and the right-hand side in each synchronous rule.

3 Syntax-Label Clustering

As an alternative to using the similarity of probabilistic distributions as a criterion for syntax-label clustering, we propose a clustering method based on the maximum likelihood of the synchronous rules in a training data $\mathcal{D}$. We uses the idea of maximizing the Bayesian posterior probability $P(M|\mathcal{D})$ of the overall model structure M given data $\mathcal{D}$ (Stolcke and Omohundro, 1994). While their goal is to maximize the posterior

$$P(M|\mathcal{D}) \propto P(M)P(\mathcal{D}|M) \tag{3}$$

we omit the prior term $P(M)$ and directly maximize the $P(\mathcal{D}|M)$. A model M is a clustering structure[1] . The synchronous rule in the data $\mathcal{D}$ for SAMT with target-side syntax labels is represented as

$$X \rightarrow \langle a_1 Y^{(1)} a_2 Z^{(2)} a_3, b_1 Y^{(1)} b_2 Z^{(2)} b_3 \rangle \tag{4}$$

where a_1, a_2, a_3 and b_1, b_2, b_3 are the source- and target-side terminals, respectively X, Y, Z are nonterminal syntax labels, and the superscript number indicates alignment between the source- and target-side nonterminals. Using Equation (4) we maximize the posterior probability $P(\mathcal{D}|M)$ which we define as the probability of right-hand side given the syntax label X of the left-hand side rule in the training data as follows:

$$\sum_{X \rightarrow \langle \alpha, \beta, \sim \rangle \in \mathcal{D}} \log Pr(\langle \alpha, \beta, \sim \rangle | X) \tag{5}$$

For the sake of simplicity, we assume that the generative probability for each rule does not depend on the existence of terminal symbols and that the reordering in the target side may be ignored. Therefore, Equation (5) simplifies to

$$\sum_{X \rightarrow \langle a_1 Y^{(1)} a_2 Z^{(2)} a_3, b_1 Y^{(1)} b_2 Z^{(2)} b_3 \rangle} \log p(Y, Z|X) \tag{6}$$

[1] $P(M)$ is reflected by the number of clusters.

3.1 Optimization Criterion

The generative probability in each rule of the form of Equation (6) can be approximated by clustering nonterminal symbols as follows:

$$p(Y,Z|X) \approx p(Y|c(Y)) \cdot p(Z|c(Z)) \cdot p(c(Y),c(Z)|c(X)) \quad (7)$$

where we map a syntax label X to its equivalence cluster $c(X)$. This can be regarded as the clustering criterion usually used in a class-based n-gram language model (Brown et al., 1992). If each label on the right-hand side of a synchronous rule (4) is independent of each other, we can factor the joint model as follows:

$$p(Y,Z|X) \approx p(Y|c(Y)) \cdot p(Z|c(Z)) \cdot p(c(Y)|c(X)) p(c(Z)|c(X)) \quad (8)$$

We introduce the predictive idea of Uszkoreit and Brants (2008) to Equation (8), which doesn't condition on the clustered label $c(X)$, but directly on the syntax label X:

$$p(Y,Z|X) \approx p(Y|c(Y)) \cdot p(Z|c(Z)) \cdot p(c(Y)|X) \cdot p(c(Z)|X) \quad (9)$$

The objective in Equation (9) is represented using the frequency in the training data as

$$\frac{N(Y)}{N(c(Y))} \cdot \frac{N(X,c(Y))}{N(X)} \cdot \frac{N(Z)}{N(c(Z))} \cdot \frac{N(X,c(Z))}{N(X)} \quad (10)$$

where $N(X)$ and $N(c(X))$ denote the frequency[2] of X and $c(X)$, and $N(X,K)$ denotes the frequency of cluster K in the right-hand side of a synchronous rule whose left-hand side syntax label is X. By replacing the rule probabilities in Equation (9) with Equation (10) and plugging the result into Equation (6), our objective becomes

$$\begin{aligned} F(\mathcal{C}) &= \sum_{Y \in \mathcal{N}} N(Y) \cdot \log \frac{N(Y)}{N(c(Y))} + \sum_{X \in \mathcal{N}, K \in \mathcal{C}} N(X,K) \cdot \log \frac{N(X,K)}{N(X)} \\ &= \sum_{Y \in \mathcal{N}} N(Y) \cdot \log N(Y) - \sum_{Y \in \mathcal{N}} N(Y) \cdot \log N(c(Y)) + \sum_{X \in \mathcal{N}, K \in \mathcal{C}} N(X,K) \cdot \log N(X,K) - \sum_{X \in \mathcal{N}, K \in \mathcal{C}} N(X,K) \cdot \log N(X) \end{aligned} \quad (11)$$

[2] We use a fractional count (Chiang, 2007) which adds up to one as a frequency.

start with the initial mapping (label $X \rightarrow c(X)$)
compute objective function $F(\mathcal{C})$
for each label X do
remove label X from $c(X)$
for each cluster K do
move label X tentatively to cluster K
compute $F(\mathcal{C})$ for this exchange
move label X to cluster with maximum $F(\mathcal{C})$
do until the cluster mapping does not change

Table 1: Outline of syntax-label clustering method

where $\mathcal{C}$ denotes all clusters and $\mathcal{N}$ denotes all syntax labels. For Equation (11), the last summation is equivalent to the sum of the occurrences of all syntax labels, and canceled out by the first summation. K in the third summation considers clusters in a synchronous rule whose left-hand side label is X, and we let $ch(X)$ denote a set of those clusters. The second summation equals $\sum_{K \in C} N(K) \cdot \log N(K)$. As a result, Equation (11) simplifies to

$$F(\mathcal{C}) = \sum_{X \in \mathcal{N}, K \in ch(X)} N(X,K) \cdot \log N(X,K) - \sum_{K \in \mathcal{C}} N(K) \cdot \log N(K) \quad (12)$$

3.2 Exchange Clustering

We used an exchange clustering algorithm (Uszkoreit and Brants, 2008) which was proven to be very efficient in word clustering with a vocabulary of over 1 million words. The exchange clustering for words begins with the initial clustering of words and greedily exchanges words from one cluster to another such that an optimization criterion is maximized after the move. While agglomerative clustering requires recalculation for all pair-wise distances between words, exchange clustering only demands computing the difference of the objective for the word pair involved in a particular movement. We applied this exchange clustering to syntax-label clustering. Table 1 shows the outline. For initial clustering, we partitioned all the syntax labels into clusters according to the frequency of syntax labels in synchronous rules. If `remove` and `move` are as computationally intensive as computing the change in $F(\mathcal{C})$ in Equation (12), then the time complexity of `remove` and `move` is $O(K)$ (Martin et al., 1998), where K is the number of clusters. Since the `remove` procedure is called once for each label and, for a given label, the `move` procedure is called $K-1$ times

Data	Lang	Training			Development		Test	
		sent	src-tokens	tgt-tokens	sent	tgt-tokens	sent	tgt-tokens
IWSLT07	J to E	40 K	483 K	369 K	500	7.4 K	489	3.7 K
FBIS	C to E	302 K	2.7 M	3.4 M	1,664	47 K	919	30 K
NIST08		1 M	15 M	17 M				

Table 2: Data sets: The "sent" column indicates the number of sentences. The "src-tokens" and "tgt-tokens" columns indicate the number of words in the source- and the target-side sentences.

to find the maximum $F(\mathcal{C})$, the worst-time complexity for one iteration of the syntax-label clustering is $O(|\mathcal{N}|K^2)$. The exchange procedure is continued until the cluster mapping is stable or the number of iterations reaches a threshold value of 100.

4 Experiments

4.1 Data

We conducted experiments on Japanese-English (ja-en) and Chinese-English (zh-en) translation tasks. The ja-en data comes from IWSLT07 (Fordyce, 2007) in a spoken travel domain. The tuning set has seven English references and the test set has six English references. For zh-en data we prepared two kind of data. The one is extracted from FBIS[3], which is a collection of news articles. The other is 1 M sentences extracted rondomly from NIST Open MT 2008 task (NIST08). We use the NIST Open MT 2006 for tuning and the MT 2003 for testing. The tuning and test sets have four English references. Table 2 shows the details for each corpus. Each corpus is tokenized, put in lower-case, and sentences with over 40 tokens on either side are removed from the training data. We use KyTea (Neubig et al., 2011) to tokenize the Japanese data and Stanford Word Segmenter (Tseng et al., 2005) to tokenize the Chinese data. We parse the English data with the Berkeley parser (Petrov and Klein, 2007).

4.2 Experiment design

We did experiments with the SAMT (Zollmann and Venugopal, 2006) model with the Moses (Koehn et al., 2007). For the SAMT model, we conducted experiments with two label sets. One is extracted from the phrase structure parses and the other is extended with CCG[4]. We applied the proposed method (+*clustering*) and the baseline method (+*coarsening*), which uses the Hanneman label-collapsing algorithm described in Section 2, for syntax-label clustering to the SAMT models with *CCG*. The number of clusters for each clustering was set to 80. The language models were built using SRILM Toolkits (Stolcke, 2002). The language model with the IWSLT07 is a 5-gram model trained on the training data, and the language model with the FBIS and NIST08 is a 5-gram model trained on the Xinhua portion of English GigaWord. For word alignments, we used MGIZA++ (Gao and Vogel, 2008). To tune the weights for BLEU (Papineni et al., 2002), we used the n-best batch MIRA (Cherry and Foster, 2012).

[3]LDC2003E14

[4]Using the relax-parse with option SAMT 4 for IWSLT07 and FBIS and SAMT 2 for NIST08 in the Moses

Label set	Label	Rule	$\mathbf{F}(\mathcal{C})$	SD
parse	63	0.3 K	-	-
CCG	3,147	4.2 M	-	-
+ *coarsening*	80	2.4 M	-3.8 e+08	249
+ *clustering*	80	3.8 M	-7.2 e+07	73

Table 3: SAMT grammars on ja-en experiments

Label set	Label	Rule	$\mathbf{F}(\mathcal{C})$	SD
FBIS				
parse	70	2.1 M	-	-
CCG	5,460	60 M	-	-
+ *coarsening*	80	32 M	-1.5 e+10	526
+ *clustering*	80	38 M	-7.9 e+09	154
NIST08				
parse	70	12 M	-	-
CCG	7,328	120 M	-	-
+ *clustering*	80	100 M	-2.6 e+10	218

Table 4: SAMT grammars on zh-en experiments

5 Results and analysis

Tables 3 and 4 present the details of SAMT grammars with each label set learned by the experiments using the IWSLT07 (ja-en), FBIS and NIST08 (zh-en), which include the number of syntax labels and synchronous rules, the values of the objective ($F(\mathcal{C})$), and the standard deviation (SD) of the number of labels assigned to each cluster. For NIST08 we applied only the + *clustering* because the + *coarsening* needs a huge amount of computation time. Table 5 shows the differences between the BLEU score and the rule number for

each cluster number when using the IWSLT07 dataset.

Since the +*clustering* maximizes the likelihood of synchronous rules, it can introduce appropriate rules adapted to training data given a fixed number of clusters. For each experiment, SAMT grammars with the +*clustering* have a greater number of rules than with the +*coarsening* and, as shown in Table 5, the number of synchronous rules with +*clustering* increase with the number of clusters. For +*clustering* with eight clusters and +*coarsening* with 80 clusters, which have almost $2.4M$ rules, the BLEU score of +*clustering* with eight clusters is higher. Also, the SD of the number of labels, which indicates the balance of the number of labels among clusters, with +*clustering* is smaller than with +*coarsening*. These results suggest that +*clustering* maintain a large-scale variation of synchronous rules for high performance by balancing the number of labels in each cluster.

The number of synchronous rules grows as you progress from +*coarsening* to +*clustering* and finally to raw label with *CCG*. To confirm the effect of the number of rules, we measured the decoding time per sentence for translating the test set by taking the average of ten runs with FBIS corpus. +*coarsening* takes 0.14 s and +*clustering* takes 0.16 s while raw label with CCG takes 0.37s. Thus the increase in the number of synchronous rules adversely affects the decoding speed.

Table 6 presents the results for the experiments[5] using ja-en and zh-en with the BLEU metric. SAMT with *parse* have the lowest BLEU scores. It appears that the linguistic information of the raw syntax labels of the phrase structure parses is not enough to improve the translation performance. Hiero has the higher BLEU score than SAMT with *CCG* on zh-en. This is likely due to the low accuracy of the parses, on which SAMT relies while Hiero doesn't. SAMT with + *clustering* have the higher BLEU score than raw label with *CCG*. For SAMT with *CCG* using IWSLT07 and FBIS, though the statistical significance tests were not significant when $p < 0.05$, +*clustering* have the higher BLEU scores than +*coarsening*. For these results, the performance of +*clustering* is comparable to that of +*coarsening*. For the complexity of both clustering algorithm, though it is difficult to evaluate directly because the speed

[5]As another baseline, we also used Phrase-based SMT (Koehn et al., 2003) and Hiero (Chiang, 2007).

	+*clustering*				+*coarsening*
Cluster	**80**	**40**	**8**	**4**	**80**
BLEU	50.21	49.49	49.96	50.25	49.54
Rule	3.8 M	3.5 M	2.4 M	2.2 M	2.4 M

Table 5: BLEU score and rule number for each cluster number using IWSLT07

	ja-en		**zh-en**			
Model	***parse***	***CCG***	***parse***	***CCG***	***parse***	***CCG***
SAMT	42.58	48.77	23.66	26.97	24.67	27.28
+*coarsening*	-	49.54	-	27.12	-	-
+*clustering*	-	**50.21**	-	**27.47**	-	27.29
Hiero	48.91		**28.31**		**27.62**	
PB-SMT	49.14		26.88		26.71	

Table 6: BLEU scores on each experiments

depends on how each algorithm is implemented, +*clustering* is an order of magnitude faster than +*coarsening*. For the clustering experiment that groups 5460 raw labels with *CCG* into 80 clusters using FBIS corpus, +*coarsening* takes about 1 week whereas +*clustering* takes about 10 minutes.

6 Conclusion

In this paper, we propose syntax-label clustering for SAMT, which uses syntax-label information to generate syntactically correct translations. One of the problems of SAMT is the large grammar size when a CCG-style extended label set is used in the grammar, which make decoding slower. We cluster syntax labels with a very fast exchange algorithm in which the generative probabilities of synchronous rules are maximized. We demonstrate the effectiveness of the proposed method by using it to translate Japanese-English and Chinese-English tasks and measuring the decoding speed, the accuracy and the clustering speed. Future work involves improving the optimization criterion. We expect to make a new objective that includes the terminal symbols and the reordering of nonterminal symbols that were ignored in this work. Another interesting direction is to determine the appropriate number of clusters for each corpus and the initialization method for clustering.

Acknowledgments

We thank the anonymous reviewers for their suggestions and helpful comments on the early version of this paper.

References

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Montréal, Canada, June. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, pages 201–228, June.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452, Uppsala, Sweden, July. Association for Computational Linguistics.

Cameron Shaw Fordyce. 2007. Overview of the 4th international workshop on spoken language translation iwslt 2007 evaluation campaign. In *In Proceedings of IWSLT 2007*, pages 1–12, Trento, Italy, October.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968, Sydney, Australia, July. Association for Computational Linguistics.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57, Columbus, Ohio, June. Association for Computational Linguistics.

Greg Hanneman and Alon Lavie. 2013. Improving syntax-augmented machine translation by coarsening the label set. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 288–297, Atlanta, Georgia, June. Association for Computational Linguistics.

Bryant Huang and Kevin Knight. 2006. Relabeling syntax trees to improve syntax-based machine translation quality. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 240–247, New York City, USA, June. Association for Computational Linguistics.

Phillip Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *In Proceedings of HLT-NAACL*, pages 48–54, Edmonton, Canada, May/July.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616, Sydney, Australia, July. Association for Computational Linguistics.

Sven Martin, Jorg Liermann, and Hermann Ney. 1998. Algorithms for bigram and trigram word clustering. In *Speech Communication*, pages 19–37.

Markos Mylonakis and Khalil Sima'an. 2011. Learning hierarchical translation structure with linguistic annotations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 642–652, Portland, Oregon, USA, June. Association for Computational Linguistics.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 529–533, Portland, Oregon, USA, June. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

Mark Steedman. 2000. *The syntactic process*, volume 27. MIT Press.

Andreas Stolcke and Stephen Omohundro. 1994. Inducing probabilistic grammars by bayesian model

merging. In R. C. Carrasco and J. Oncina, editors, *Grammatical Inference and Applications (ICGI-94)*, pages 106–118. Berlin, Heidelberg.

Andreas Stolcke. 2002. Srilm an extensible language modeling toolkit. In *In Proceedings of the Seventh International Conference on Spoken Language Processing*, pages 901–904.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter. *In Fourth SIGHAN Workshop on Chinese Language Processing*, pages 168–171. Jeju Island, Korea.

Jakob Uszkoreit and Thorsten Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *Proceedings of ACL-08: HLT*, pages 755–762, Columbus, Ohio, June. Association for Computational Linguistics.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 138–141, New York City, June. Association for Computational Linguistics.

Testing for Significance of Increased Correlation with Human Judgment

Yvette Graham **Timothy Baldwin**
Department of Computing and Information Systems
The University of Melbourne
graham.yvette@gmail.com, tb@ldwin.net

Abstract

Automatic metrics are widely used in machine translation as a substitute for human assessment. With the introduction of any new metric comes the question of just how well that metric mimics human assessment of translation quality. This is often measured by correlation with human judgment. Significance tests are generally not used to establish whether improvements over existing methods such as BLEU are statistically significant or have occurred simply by chance, however. In this paper, we introduce a significance test for comparing correlations of two metrics, along with an open-source implementation of the test. When applied to a range of metrics across seven language pairs, tests show that for a high proportion of metrics, there is insufficient evidence to conclude significant improvement over BLEU.

1 Introduction

Within machine translation (MT), efforts are ongoing to improve evaluation metrics and find better ways to automatically assess translation quality. The process of validating a new metric involves demonstration that it correlates better with human judgment than a standard metric such as BLEU (Papineni et al., 2001). However, although it is standard practice in MT evaluation to measure increases in automatic metric scores with significance tests (Germann, 2003; Och, 2003; Kumar and Byrne, 2004; Koehn, 2004; Riezler and Maxwell, 2005; Graham et al., 2014), this has not been the case in papers proposing new metrics. Thus it is possible that some reported improvements in correlation with human judgment are attributable to chance rather than a systematic improvement.

In this paper, we motivate and introduce a novel significance test to assess the statistical significance of differences in correlation with human judgment for pairs of automatic metrics. We apply tests to the WMT-12 shared metrics task to compare each of the participating methods, and find that for a high proportion of metrics, there is not enough evidence to conclude that they significantly outperform BLEU.

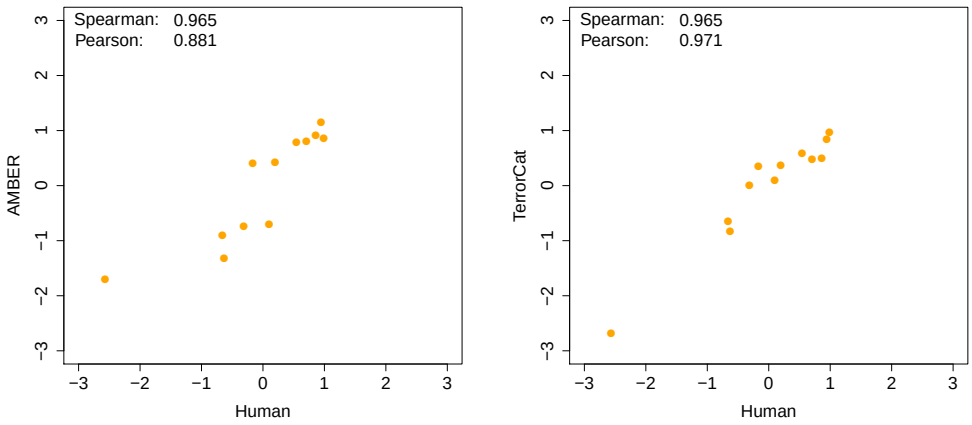
2 Correlation with Human Judgment

A common means of assessing automatic MT evaluation metrics is Spearman's rank correlation with human judgments (Melamed et al., 2003), which measures the relative degree of monotonicity between the metric and human scores in the range $[-1, 1]$. The standard justification for calculating correlations over ranks rather than raw scores is to: (a) reduce anomalies due to absolute score differences; and (b) focus evaluation on what is generally the primary area of interest, namely the ranking of systems/translations.

An alternative means of evaluation is Pearson's correlation, which measures the linear correlation between a metric and human scores (Leusch et al., 2003). Debate on the relative merits of Spearman's and Pearson's correlation for the evaluation of automatic metrics is ongoing, but there is an increasing trend towards Pearson's correlation, e.g. in the recent WMT-14 shared metrics task.

Figure 1 presents the system-level results for two evaluation metrics – AMBER (Chen et al., 2012) and TERRORCAT (Fishel et al., 2012) – over the WMT-12 Spanish-to-English metrics task. These two metrics achieved the joint-highest rank correlation ($\rho = 0.965$) for the task, but differ greatly in terms of Pearson's correlation ($r = 0.881$ vs. 0.971, resp.). The largest contributor to this artifact is the system with the lowest human score, represented by the leftmost point in both plots.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 172–176,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics



(a) AMBER (b) TERRORCAT

Figure 1: Scatter plot of human and automatic scores of WMT-12 Spanish-to-English systems for two MT evaluation metrics (AMBER and TERRORCAT)

Consistent with the WMT-14 metrics shared task, we argue that Pearson's correlation is more sensitive than Spearman's correlation. There is still the question, however, of whether an observed difference in Pearson's r is statistically significant, which we address in the next section.

3 Significance Testing

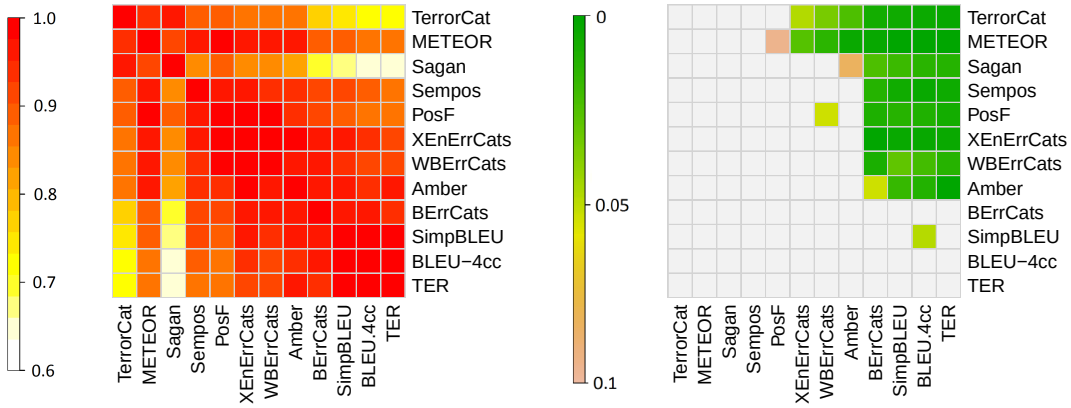
Evaluation of a new automatic metric, M_{new}, commonly takes the form of quantifying the correlation between the new metric and human judgment, $r(M_{new}, H)$, and contrasting it with the correlation for some baseline metric, $r(M_{base}, H)$. It is very rare in the MT literature for significance testing to be performed in such cases, however. We introduce a statistical test which can be used for this purpose, and apply the test to the evaluation of metrics participating in the WMT-12 metric evaluation task.

At first gloss, it might seem reasonable to perform significance testing in the following manner when an increase in correlation with human assessment is observed: apply a significance test separately to the correlation of each metric with human judgment, with the hope that the newly proposed metric will achieve a significant correlation where the baseline metric does not. However, besides the fact that the correlation between almost any document-level metric and human judgment will generally be significantly greater than zero, the logic here is flawed: the fact that one correlation is significantly higher than zero ($r(M_{new}, H)$) and that of another is not, does not necessarily mean that the *difference* between the two correlations is significant. Instead, a specific test should be applied to the difference in correlations on the data. For this same reason, confidence intervals for individual correlations with human judgment are also not particularly meaningful.

In psychological studies, it is often the case that samples that data are drawn from are independent, and differences in correlations are computed on independent data sets. In such cases, the Fisher r to z transformation is applied to test for significant differences in correlations. In the case of automatic metric evaluation, however, the data sets used are almost never independent. This means that if $r(M_{base}, H)$ and $r(M_{new}, H)$ are both > 0, the correlation between the metric scores themselves, $r(M_{base}, M_{new})$, must also be > 0. The strength of this correlation, directly between pairs of metrics, should be taken into account using a significance test of the difference in correlation between $r(M_{base}, H)$ and $r(M_{new}, H)$.

3.1 Correlated Correlations

Correlations computed for two separate automatic metrics on the same data set are not independent, and for this reason in order to test the difference in correlation between them, the degree to which the pair of metrics correlate with each other should be taken into account. The Williams test (Williams,



(a) Pearson's correlation (b) Statistical significance

Figure 2: (a) Pearson's correlation between pairs of automatic metrics; and (b) p-value of Williams significance tests, where a colored cell in row i (named on y-axis), col j indicates that metric i (named on x-axis) correlates significantly higher with human judgment than metric j; all results are based on the WMT-12 Spanish-to-English data set.

1959)[1] evaluates significance in a difference in dependent correlations (Steiger, 1980). It is formulated as follows, as a test of whether the population correlation between X_1 and X_3 equals the population correlation between X_2 and X_3:

$$t(n-3) = \frac{(r_{13} - r_{23})\sqrt{(n-1)(1+r_{12})}}{\sqrt{2K\frac{(n-1)}{(n-3)} + \frac{(r_{23}+r_{13})^2}{4}(1-r_{12})^3}},$$

where r_{ij} is the Pearson correlation between X_i and X_j, n is the size of the population, and:

$$K = 1 - {r_{12}}^2 - {r_{13}}^2 - {r_{23}}^2 + 2r_{12}r_{13}r_{23}$$

The Williams test is more powerful than the equivalent for independent samples (Fisher r to z), as it takes the correlations between X_1 and X_2 (metric scores) into account. All else being equal, the higher the correlation between the metric scores, the greater the statistical power of the test.

4 Evaluation and Discussion

Figure 2a is a heatmap of the degree to which automatic metrics correlate with one another when computed on the same data set, in the form of the Pearson's correlation between each pair of metrics that participated in the WMT-12 metrics task for Spanish-to-English evaluation. Metrics are ordered in all tables from highest to lowest *correlation with human assessment*. In addition, for the purposes of significance testing, we take the absolute value of all correlations, in order to compare error-based metrics with non-error based ones.

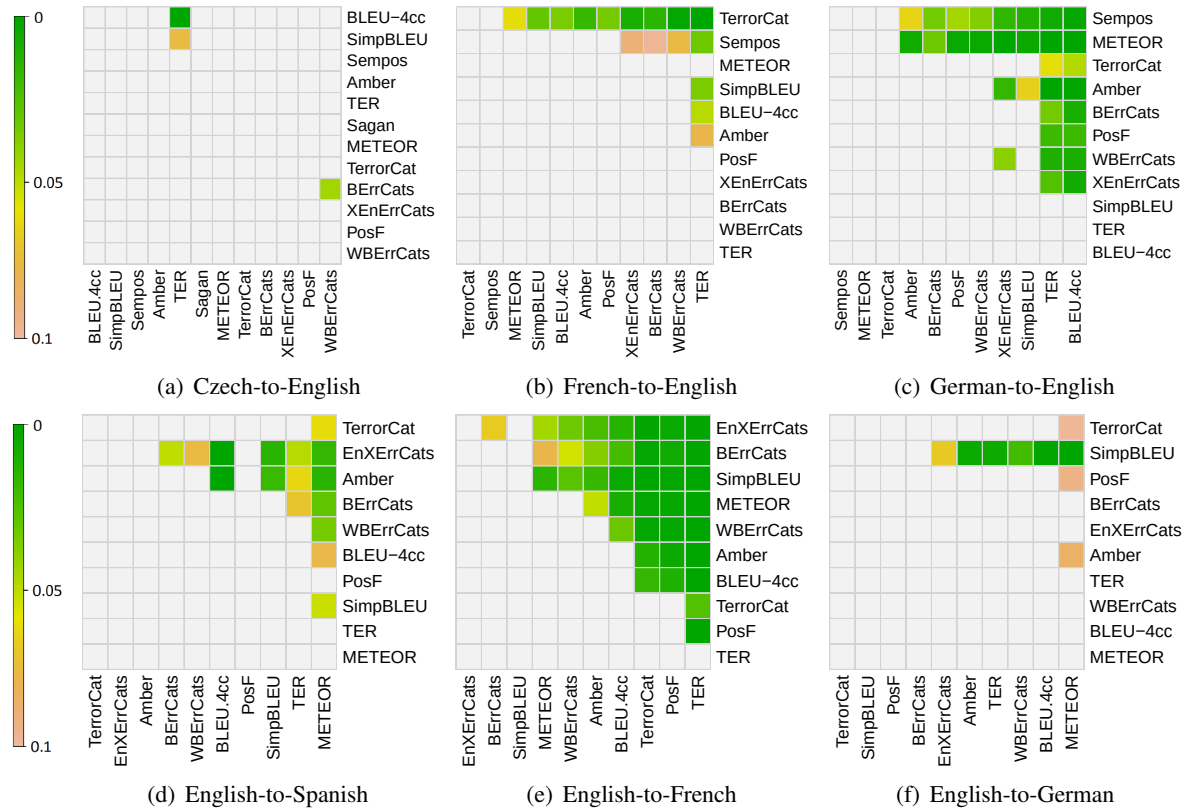
In general, the correlation is high amongst all pairs of metrics, with a high proportion of paired metrics achieving a correlation in excess of $r = 0.9$. Two exceptions to this are TERRORCAT (Fishel et al., 2012) and SAGAN (Castillo and Estrella, 2012), as seen in the regions of yellow and white.

Figure 2b shows the results of Williams significance tests for all pairs of metrics. Since we are interested in not only identifying significant differences in correlations, but ultimately ranking competing metrics, we use a one-sided test. Here again, the metrics are ordered from highest to lowest (absolute) correlation with human judgment.

For the Spanish-to-English systems, approximately 60% of WMT-12 metric pairs show a significant difference in correlation with human judgment at $p < 0.05$ (for one of the two metric directions).[2] As expected, the higher the correlation with human judgment, the more metrics a given method is superior to at a level of statistical significance. Although TERRORCAT (Fishel et al., 2012) achieves the highest absolute correlation with human judgment, it is not significantly better ($p \geq 0.05$) than the four next-best metrics (METEOR (Denkowski and Lavie, 2011), SAGAN (Castillo and Estrella, 2012), SEMPOS (Macháček and Bo-

[1] Also sometimes referred to as the Hotelling–Williams test.

[2] Correlation matrices (red) are maximally filled, in contrast to one-sided significance test matrices (green), where, at a maximum, fewer than half of the cells can be filled.



(a) Czech-to-English (b) French-to-English (c) German-to-English

(d) English-to-Spanish (e) English-to-French (f) English-to-German

Figure 3: Significance results for pairs of automatic metrics for each WMT-12 language pair.

jar, 2011) and POSF (Popovic, 2012)). There is not enough evidence to conclude, therefore, that this metric is any better at evaluating Spanish-to-English MT system quality than the next four metrics.

Figure 3 shows the results of significance tests for the six other language pairs used in the WMT-12 metrics shared task.[3] For no language pair is there an outright winner amongst the metrics, with proportions of significant differences between metrics for a given language pair ranging from 3% for Czech-to-English to 82% for English-to-French ($p < 0.05$). The number of metrics that significantly outperform BLEU for a given language pair is only 34% ($p < 0.05$), and no method significantly outperforms BLEU over all language pairs – indeed, even the best methods achieve statistical significance over BLEU for only a small minority of language pairs. This underlines the dangers of assessing metrics based solely on correlation numbers, and emphasizes the importance of statistical testing.

It is important to note that the number of competing metrics a metric significantly outperforms should not be used as the criterion for ranking competing metrics. This is due to the fact that the power of the Williams test to identify significant differences between correlations changes depending on the degree to which the pair of metrics correlate with each other. Therefore, a metric that happens to correlate strongly with many other metrics would be at an unfair advantage, were numbers of significant wins to be used to rank metrics. For this reason, it is best to interpret pairwise metric tests in isolation.

As part of this research, we have made available an open-source implementation of statistical tests tailored to the assessment of MT metrics available at `https://github.com/ygraham/significance-williams`.

5 Conclusions

We have provided an analysis of current methodologies for evaluating automatic metrics in machine translation, and identified an issue with respect to the lack of significance testing. We introduced the Williams test as a means of calculating the statistical significance of differences

[3] We omit English-to-Czech due to some metric scores being omitted from the WMT-12 data set.

in correlations for dependent samples. Analysis of statistical significance in the WMT-12 metrics shared task showed there is currently insufficient evidence for a high proportion of metrics to conclude that they outperform BLEU.

Acknowledgments

We wish to thank the anonymous reviewers for their valuable comments. This research was supported by funding from the Australian Research Council.

References

Julio Castillo and Paula Estrella. 2012. Semantic textual similarity for MT evaluation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 52–58, Montréal, Canada.

Boxing Chen, Roland Kuhn, and George Foster. 2012. Improving AMBER, an MT evaluation metric. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 59–63, Montréal, Canada.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91, Edinburgh, UK.

Mark Fishel, Rico Sennrich, Maja Popović, and Ondřej Bojar. 2012. TerrorCat: a translation error categorization-based MT quality metric. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 64–70, Montréal, Canada.

Ulrich Germann. 2003. Greedy decoding for statistical machine translation in almost linear time. In *Proceedings of the 2003 Conference of the North American Chapter of the Assoc. Computational Linguistics on Human Language Technology-Volume 1*, pages 1–8, Edmonton, Canada.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2014. Randomized significance tests in machine translation. In *Proceedings of the ACL 2014 Ninth Workshop on Statistical Machine Translation*, pages 266–274, Baltimore, USA.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of Empirical Methods in Natural Language Processing 2004 (EMNLP 2004)*, pages 388–395, Barcelona, Spain.

Shankar Kumar and William Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the 4th International Conference on Human Language Technology Research and 5th Annual Meeting of the NAACL (HLT-NAACL 2004)*, pages 169–176, Boston, USA.

Gregor Leusch, Nicola Ueffing, and Hermann Ney. 2003. A novel string-to-string distance measure with applications to machine translation evaluation. In *Proceedings 9th Machine Translation Summit (MT Summit IX)*, pages 240–247, New Orleans, USA.

Matouš Macháček and Ondřej Bojar. 2011. Approximating a deep-syntactic metric for MT evaluation and tuning. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 92–98, Edinburgh, UK.

Dan Melamed, Ryan Green, and Joseph Turian. 2003. Precision and recall of machine translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL 2003) — Short Papers*, pages 61–63, Edmonton, Canada.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: A method for automatic evaluation of machine translation. Technical Report RC22176 (W0109-022), IBM Research, Thomas J. Watson Research Center.

Maja Popovic. 2012. Class error rates for evaluation of machine translation output. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 71–75, Montréal, Canada.

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

James H. Steiger. 1980. Tests for comparing elements of a correlation matrix. *Psychological Bulletin*, 87(2):245.

Evan J. Williams. 1959. *Regression Analysis*, volume 14. Wiley, New York, USA.

Syntactic SMT Using a Discriminative Text Generation Model

Yue Zhang
SUTD, Singapore
yue_zhang@sutd.edu.sg

Kai Song*
NEU, China
songkai.sk@alibaba-inc.com

Linfeng Song*
ICT/CAS, China
songlinfeng@ict.ac.cn

Jingbo Zhu
NEU, China
zhujingbo@mail.neu.edu.cn

Qun Liu
CNGL, Ireland and ICT/CAS, China
qliu@computing.dcu.ie

Abstract

We study a novel architecture for syntactic SMT. In contrast to the dominant approach in the literature, the system does not rely on translation rules, but treat translation as an unconstrained target sentence generation task, using soft features to capture lexical and syntactic correspondences between the source and target languages. Target syntax features and bilingual translation features are trained consistently in a discriminative model. Experiments using the IWSLT 2010 dataset show that the system achieves BLEU comparable to the state-of-the-art syntactic SMT systems.

1 Introduction

Translation rules have been central to hierarchical phrase-based and syntactic statistical machine translation (SMT) (Galley et al., 2004; Chiang, 2005; Liu et al., 2006; Quirk et al., 2005; Marcu et al., 2006; Shen and Joshi, 2008; Xie et al., 2011). They are attractive by capturing the recursiveness of languages and syntactic correspondences between them. One important advantage of translation rules is that they allow efficient decoding by treating MT as a statistical **parsing** task, transforming a source sentence to its translation via recursive rule application.

The efficiency takes root in the fact that target word orders are encoded in translation rules. This fact, however, also leads to rule explosion, noise and coverage problems (Auli et al., 2009), which can hurt translation quality. Flexibility of function word usage, rich morphology and paraphrasing all add to the difficulty of rule extraction. In addition, restricting target word orders by hard translation rules can also hurt output fluency.

* Work done while visiting Singapore University of Technology and Design (SUTD)

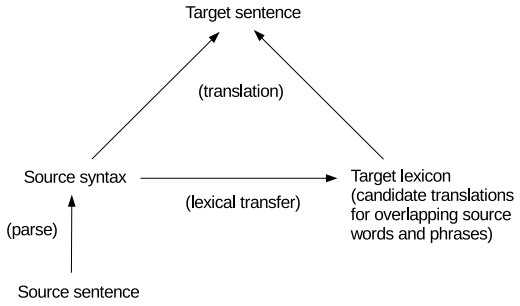


Figure 1: Overall system architecture.

A potential solution to the problems above is to treat translation as a **generation** task, representing syntactic correspondences using *soft* features. Both adequacy and fluency can potentially be improved by giving full flexibility to target synthesis, and leaving all options to the statistical model. The main challenge to this method is a significant increase in the search space (Knight, 1999). To this end, recent advances in tackling complex search tasks for text generation offer some solutions (White and Rajkumar, 2009; Zhang and Clark, 2011).

In this short paper, we present a preliminary investigation on the possibility of building a syntactic SMT system that does not use hard translation rules, by utilizing recent advances in statistical natural language generation (NLG). The overall architecture is shown in Figure 1. Translation is performed by first parsing the source sentence, then transferring source words and phrases to their target equivalences, and finally synthesizing the target output.

We choose dependency grammar for both the source and the target syntax, and adapt the syntactic text synthesis system of Zhang (2013), which performs dependency-based linearization. The linearization task for MT is different from the monolingual task in that not all translation options are used to build the output, and that bilingual correspondences need to be taken into account dur-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 177–182,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

ing synthesis. The algorithms of Zhang (2013) are modified to perform word selection as well as ordering, using two sets of features to control translation adequacy and fluency, respectively.

Preliminary experiments on the IWSLT[1] 2010 data show that the system gives BLEU comparable to traditional tree-to-string and string-to-tree translation systems. It demonstrates the feasibility of leveraging statistical NLG techniques for SMT, and the possibility of building a statistical transfer-based MT system.

2 Approach

The main goal being proof of concept, we keep the system simple by utilizing existing methods for the main components, minimizing engineering efforts. Shown in Figure 1, the end-to-end system consists of two main components: *lexical transfer* and *synthesis*. The former provides candidate translations for (overlapping) source words and phrases. Although lexicons and rules can be used for this step, we take a simple statistical alignment-based approach. The latter searches for a target translation by constructing dependency trees bottom-up. The process can be viewed as a syntax-based generation process from a bag of overlapping translation options.

2.1 Lexical transfer

We perform word alignment using IBM model 4 (Brown et al., 1993), and then extract phrase pairs according to the alignment and automatically-annotated target syntax. In particular, consistent (Och et al., 1999) and cohesive (Fox, 2002) phrase pairs are extracted from intersected alignments in both directions: the target side must form a projective span, with a single root, and the source side must be contiguous. A resulting phrase pair consists of the source phrase, its target translation, as well as the head position and head part-of-speech (POS) of the target span, which are useful for target synthesis. We further restrict that neither the source nor the target side of a valid phrase pair contains over s words.

Given an input source sentence, the lexical transfer unit finds all valid target translation options for overlapping source phrases up to size s, and feeds them as inputs to the target synthesis decoder. The translation options with a probability below $\lambda \cdot P_{max}$ are filtered out, where P_{max} is the probability of the most probable translation. Here the probability of a target translation is calculated as the count of the translation divided by the count of all translations of the source phrase.

2.2 Synthesis

The synthesis module is based on the monolingual text synthesis algorithm of Zhang (2013), which constructs an ordered dependency tree given a bag of words. In the bilingual setting, inputs to the algorithm are translation options, which can be overlapping and mutually exclusive, and not necessarily all of which are included in the output. As a result, the decoder needs to perform word selection in addition to word ordering. Another difference between the bilingual and monolingual settings is that the former requires translation adequacy in addition to output fluency.

We largely rely on the monolingual system for MT decoding. To deal with overlapping translation options, a source coverage vector is used to impose mutual exclusiveness on input words and phrases. Each element in the coverage vector is a binary value that indicates whether a particular source word has been translated in the corresponding target hypothesis. For translation adequacy, we use a set of bilingual features on top of the set of monolingual features for text synthesis.

2.2.1 Search

The search algorithm is the best-first algorithm of Zhang (2013). Each search hypothesis is a partial or full target-language dependency tree, and hypotheses are constructed bottom-up from leaf nodes, which are translation options. An *agenda* is used to maintain a list of search hypothesis to be expanded, and a *chart* is used to record a set of accepted hypotheses. Initially empty, the chart is a beam of size $k \cdot n$, where n is the number of source words and k is a positive integer. The agenda is a priority queue, initialized with all leaf hypotheses (i.e. translation options). At each step, the highest-scored hypothesis e is popped off the agenda, and expanded by combination with all hypotheses on the chart in all possible ways, with the set of newly generated hypotheses $e_1, e_2, ...e_N$ being put onto the agenda, and e being put onto the chart. When two hypotheses are combined, they can be put in two different orders, and in each case different dependencies can be constructed between their head words, leading to different new

[1]International Workshop on Spoken Language Translation, *http://iwslt2010.fbk.eu*

dependency syntax
WORD(h) · POS(h) · NORM($size$) ,
WORD(h) · NORM($size$), POS(h) · NORM($size$)
POS(h) · POS(m) · POS(b) · dir
POS(h) · POS(h_l) · POS(m) · POS(m_r) · dir ($h > m$),
POS(h) · POS(h_r) · POS(m) · POS(m_l) · dir ($h < m$)
WORD(h) · POS(m) · POS(m_l) · dir,
WORD(h) · POS(m) · POS(m_r) · dir
POS(h) · POS(m) · POS(m_1) · dir,
POS(h) · POS(m_1) · dir, POS(m) · POS(m_1) · dir
WORD(h) · POS(m) · POS(m_1) · POS(m_2) · dir,
POS(h) · POS(m) · POS(m_1) · POS(m_2) · dir ,
...

dependency syntax for completed words
WORD(h) · POS(h) · WORD(h_l) · POS(h_l),
POS(h) · POS(h_l),
WORD(h) · POS(h) · POS(h_l),
POS(h) · WORD(h_l) · POS(h_l) ,
WORD(h) · POS(h) · WORD(h_r) · POS(h_r),
POS(h) · POS(h_r),
...

surface string patterns (B—bordering index)
WORD($B-1$) · WORD(B), POS($B-1$) · POS(B),
WORD($B-1$) · POS(B), POS($B-1$) · WORD(B),
WORD($B-1$) · WORD(B) · WORD($B+1$),
WORD($B-2$) · WORD($B-1$) · WORD(B),
POS($B-1$) · POS(B) · POS($B+1$),
...

surface string patterns for complete sentences
WORD(0), WORD(0) · WORD(1),
WORD($size-1$),
WORD($size-1$) · WORD($size-2$),
POS(0), POS(0) · POS(1),
POS(0) · POS(1) · POS(2),
...

Table 1: Monolingual feature templates.

hypotheses. The decoder expands a fixed number L hypotheses, and then takes the highest-scored chart hypothesis that contains over $\beta \cdot n$ words as the output, where β is a real number near 1.0.

2.2.2 Model and training

A scaled linear model is used by the decoder to score search hypotheses:

$$Score(e) = \frac{\vec{\theta} \cdot \Phi(e)}{|e|},$$

where $\Phi(e)$ is the global feature vector of the hypothesis e, $\vec{\theta}$ is the parameter vector of the model, and $|e|$ is the number of leaf nodes in e. The scaling factor $|e|$ is necessary because hypotheses with different numbers of words are compared with each other in the search process to capture translation equivalence.

While the monolingual features of Zhang (2013) are applied (example feature templates from the system are shown in Table 1), an additional set of bilingual features is defined, shown

phrase translation features
PHRASE(m) · PHRASE(t), $P(trans)$,

bilingual syntactic features
POS(th) · POS(tm) · dir · LEN($path$),
WORD(th) · POS(tm) · dir · LEN($path$),
POS(th) · WORD(tm) · dir · LEN($path$),
WORD(th) · WORD(tm) · dir · LEN($path$),
WORD(sh) · WORD(sm) · dir · LEN($path$),
WORD(sh) · WORD(th) · dir · LEN($path$),
WORD(sm) · WORD(tm) · dir · LEN($path$),

bilingual syntactic features (LEN($path$) ≤ 3)
POS(th) · POS(tm) · dir · LABELS($path$),
WORD(th) · POS(tm) · dir · LABELS($path$),
POS(th) · WORD(tm) · dir · LABELS($path$),
WORD(th) · WORD(tm) · dir · LABELS($path$),
WORD(sh) · WORD(sm) · dir · LABELS($path$),
WORD(sh) · WORD(th) · dir · LABELS($path$),
WORD(sm) · WORD(tm) · dir · LABELS($path$),
POS(th) · POS(tm) · dir · LABELSPOS($path$),
WORD(th) · POS(tm) · dir · LABELSPOS($path$),
POS(th) · WORD(tm) · dir · LABELSPOS($path$),
WORD(th) · WORD(tm) · dir · LABELSPOS($path$),
WORD(sh) · WORD(sm) · dir · LABELSPOS($path$),
WORD(sh) · WORD(th) · dir · LABELSPOS($path$),
WORD(sm) · WORD(tm) · dir · LABELSPOS($path$),

Table 2: Bilingual feature templates.

in Table 2. In the tables, s and t represent the source and target, respectively; h and m represent the head and modifier in a dependency arc, respectively; h_l and h_r represent the neighboring words on the left and right of h, respectively; m_l and m_r represent the neighboring words on the left and right of m, respectively; m_1 and m_2 represent the closest and second closest sibling of m on the side of h, respectively. dir represents the arc direction (i.e. left or right); PHRASE represents a lexical phrase; P($trans$) represents the source-to-target translation probability from the phrase-table, used as a real-valued feature; $path$ represents the shortest path in the source dependency tree between the two nodes that correspond to the target head and modifier, respectively; LEN($path$) represents the number of arcs on *path*, normalized to bins of [5, 10, 20, 40+]; LABELS($path$) represents the array of dependency arc labels on *path*; LABELSPOS($path$) represents the array of dependency arc labels and source POS on *path*. In addition, a real-valued four-gram language model feature is also used, with four-grams extracted from the surface boundary when two hypothesis are combined.

We apply the discriminative learning algorithm of Zhang (2013) to train the parameters $\vec{\theta}$. The algorithm requires training examples that consist of full target derivations, with leaf nodes being *input translation options*. However, the readily available

training examples are automatically-parsed target derivations, with leaf nodes being *the reference translation*. As a result, we apply a search procedure to find a derivation process, through which the target dependency tree is constructed from a subset of input translation options. The search procedure can be treated as a constrained decoding process, where only the oracle tree and its sub trees can be constructed. In case the set of translation options cannot lead to the oracle tree, we ignore the training instance.[2] Although the ignored training sentence pairs cannot be utilized for training the discriminative synthesizer, they are nevertheless used for building the phrase table and training the language model.

3 Experiments

We perform experiments on the IWSLT 2010 Chinese-English dataset, which consists of training sentence pairs from the dialog task (dialog) and Basic Travel and Expression Corpus (BTEC). The union of dialog and BTEC are taken as our training set, which contains 30,033 sentence pairs. For system tuning, we use the IWSLT 2004 test set (also released as the second development test set of IWSLT 2010), which contains 500 sentences. For final test, we use the IWSLT 2003 test set (also released as the first development test set of IWSLT 2010), which contains 506 sentences.

The Chinese sentences in the datasets are segmented using NiuTrans[3] (Xiao et al., 2012), while POS-tagging of both English and Chinese is performed using ZPar[4] version 0.5 (Zhang and Clark, 2011). We train the English POS-tagger using the WSJ sections of the Penn Treebank (Marcus et al., 1993), turned into lower-case. For syntactic parsing of both English and Chinese, we use the default models of ZPar 0.5.

We choose three baseline systems: a string-to-tree (S2T) system, a tree-to-string (T2S) system and a tree-to-tree (T2T) system (Koehn, 2010). The Moses release 1.0 implementations of all three systems are used, with default parameter settings. IRSTLM[5] release 5.80.03 (Federico et al., 2008) is used to train a four-gram language models over the English training data, which is applied to the baseline systems and our system. Kneser-Ney smoothing is used to train the language model.

[2]This led to the ignoring of over 40% of the training sentence pairs. For future work, we will consider substitute oracles from reachable target derivations by using maximum sentence level BLEU approximation (Nakov et al., 2012) or METEOR (Denkowski and Lavie, 2011) as selection criteria.

[3]*http://www.nlplab.com/NiuPlan/NiuTrans.ch.html*

[4]*http://sourceforge.net/projects/zpar/*

[5]*http://sourceforge.net/apps/mediawiki/irstlm*

System	T2S	S2T	T2T	OURS
BLEU	32.65	36.07	28.46	34.24

Table 3: Final results.

SOURCE: 我 现在 头痛 的 厉害 。
REF: I have a terrible headache .
OURS: now , I have a headache .
SOURCE: 我 要 带 浴缸 的 双人房 。
REF: I 'd like a twin room with a bath please .
OURS: a twin room , I 'll find a room with a bath .
SOURCE: 请 把 日元 兑换 成 美元 。
REF: can you change yen into dollars ?
OURS: please change yen into dollars .
SOURCE: 请 给 我 烤鸡 。
REF: roast chicken , please .
OURS: please have roast chicken .
SOURCE: 请 每 次 饭 后 吃 两 粒 。
REF: take two tablets after every meal .
OURS: please eat after each meal .
SOURCE: 请 结帐 。
REF: check , please .
OURS: I have to check - out , please .
SOURCE: 对 呀 那 是 本店 最 拿手 的 菜 啊 。
REF: yes , well , that 's our specialty .
OURS: ah , the food that 's right .
SOURCE: 空调 坏 了 。
REF: my air conditioner is n't working .
OURS: the air - conditioner does n't work .

Table 4: Sample output sentences.

We use the tuning set to determine the optimal number of training iterations. The translation option filter λ is set to 0.1; the phrase size limit s is set to 5 in order to verify the effectiveness of synthesis; the number of expanded nodes L is set to 200; the chart factor k is set to 16 for a balance between efficiency and accuracy; the goal parameter β is set to 0.8.

The final scores of our system and the baselines are shown in Table 3. Our system gives a BLEU of 34.24, which is comparable to the baseline systems. Some example outputs are shown in Table 4. Manual comparison does not show significant differences in overall translation adequacy or fluency between the outputs of the four systems. However, an observation is that, while our system can produce more fluent outputs, the choice of translation options can be more frequently incorrect. This suggests that while the target synthesis component is effective under the bilingual setting, a stronger lexical selection component may be necessary for better translation quality.

4 Related work

As discussed in the introduction, our work is closely related to previous studies on syntactic MT, with the salient difference that we do not rely on hard translation rules, but allow free target synthesis. The contrast can be summarized as "translation by parsing" vs "translation by generation".

There has been a line of research on generation for translation. Soricut and Marcu (2006) use a form of weighted IDL-expressions (Nederhof and Satta, 2004) for generation. Bangalore et al. (2007) treats MT as a combination of global lexical transfer and word ordering; their generation component does not perform lexical selection, relying on an n-gram language model to order target words. Goto et al. (2012) use a monotonic phrase-based system to perform target word selection, and treats target ordering as a post-processing step. More recently, Chen et al. (2014) translate source dependencies arc-by-arc to generate pseudo target dependencies, and generate the translation by reordering of arcs. In contrast with these systems, our system relies more heavily on a syntax-based synthesis component, in order to study the usefulness of statistical NLG on SMT.

With respect to syntax-based word ordering, Chang and Toutanova (2007) and He et al. (2009) study a simplified word ordering problem by assuming that the un-ordered target dependency tree is given. Wan et al. (2009) and Zhang and Clark (2011) study the ordering of a bag of words, without input syntax. Zhang et al. (2012), Zhang (2013) and Song et al. (2014) further extended this line of research by adding input syntax and allowing joint inflection and ordering. de Gispert et al. (2014) use a phrase-structure grammer for word ordering. Our generation system is based on the work of Zhang (2013), but further allows lexical selection.

Our work is also in line with the work of Liang et al. (2006), Blunsom et al. (2008), Flanigan et al. (2013) and Yu et al. (2013) in that we build a discriminative model for SMT.

5 Conclusion

We investigated a novel system for syntactic machine translation, treating MT as an unconstrained generation task, solved by using a single discriminative model with both monolingual syntax and bilingual translation features. Syntactic correspondence is captured by using soft features rather than hard translation rules, which are used by most syntax-based statistical methods in the literature.

Our results are preliminary in the sense that the experiments were performed using a relatively small dataset, and little engineering effort was made on fine-tuning of parameters for the baseline and proposed models. Our Python implementation gives the same level of BLEU scores compared with baseline syntactic SMT systems, but is an order of magnitude slower than Moses. However, the results demonstrate the feasibility of leveraging text generation techniques for machine translation, directly connecting the two currently rather separated research fields. The system is not strongly dependent on the specific generation algorithm, and one potential of the SMT architecture is that it can directly benefit from advances in statistical NLG technology.

Acknowledgement

The work has been supported by the Singapore Ministration of Education Tier 2 project T2MOE201301 and the startup grant SRG ISTD 2012 038 from SUTD. We thank the anonymous reviewers for their constructive comments.

References

Michael Auli, Adam Lopez, Hieu Hoang, and Philipp Koehn. 2009. A systematic analysis of translation model search spaces. In *Proc. WMT*, pages 224–232.

Srinivas Bangalore, Patrick Haffner, and Stephan Kanthak. 2007. Statistical machine translation through global lexical selection and sentence reconstruction. In *Proc. ACL*, pages 152–159.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proc. ACL*, pages 200–208.

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

Pi-Chuan Chang and Kristina Toutanova. 2007. A discriminative syntactic word order model for machine translation. In *Proc. ACL*, pages 9–16.

Hongshen Chen, Jun Xie, Fandong Meng, Wenbin Jiang, and Qun Liu. 2014. A dependency edge-based transfer model for statistical machine translation. In *Proc. COLING 2014*, pages 1103–1113.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. ACL*, pages 263–270.

Adrià de Gispert, Marcus Tomalin, and Bill Byrne. 2014. Word ordering with phrase-based grammars. In *Proc. EACL*, pages 259–268.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proc. WMT*, pages 85–91.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. IRSTLM: an open source toolkit for handling large scale language models. In *Proc. Interspeech*, pages 1618–1621.

Jeffrey Flanigan, Chris Dyer, and Jaime Carbonell. 2013. Large-scale discriminative training for statistical machine translation using held-out line search. In *Proc. NAACL*, pages 248–258.

Heidi Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proc. EMNLP*, pages 304–311.

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

Isao Goto, Masao Utiyama, and Eiichiro Sumita. 2012. Post-ordering by parsing for Japanese-English statistical machine translation. In *Proc. ACL*, pages 311–316.

Wei He, Haifeng Wang, Yuqing Guo, and Ting Liu. 2009. Dependency based Chinese sentence realization. In *Proc. ACL/AFNLP*, pages 809–816.

Kevin Knight. 1999. Squibs and Discussions: Decoding Complexity in Word-Replacement Translation Models. *Computational Linguistics*, 25(4):607–615.

Phillip Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press.

P. Liang, A. Bouchard-Cote, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. COLING/ACL*, pages 761–768.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. COLING/ACL*, pages 609–616.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. EMNLP*, pages 44–52.

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

Preslav Nakov, Francisco Guzman, and Stephan Vogel. 2012. Optimizing for sentence-level BLEU+1 yields short translations. In *Proc. Coling*, pages 1979–1994.

Mark-Jan Nederhof and Giorgio Satta. 2004. Idl-expressions: a formalism for representing and parsing finite languages in natural language processing. *J. Artif. Intell. Res.(JAIR)*, 21:287–317.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proc. EMNLP*, pages 20–28.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proc. ACL*, pages 271–279.

Libin Shen and Aravind Joshi. 2008. LTAG dependency parsing with bidirectional incremental construction. In *Proc. EMNLP*, pages 495–504.

Linfeng Song, Yue Zhang, Kai Song, and Qun Liu. 2014. Joint morphological generation and syntactic linearization. In *Proc. AAAI*, pages 1522–1528.

Radu Soricut and Daniel Marcu. 2006. Stochastic language generation using widl-expressions and its application in machine translation and summarization. In *Proc. ACL*, pages 1105–1112.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2009. Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model. In *Proc. EACL*, pages 852–860.

Michael White and Rajakrishnan Rajkumar. 2009. Perceptron reranking for CCG realization. In *Proc. the EMNLP*, pages 410–419.

Tong Xiao, Jingbo Zhu, Hao Zhang, and Qiang Li. 2012. NiuTrans: An open source toolkit for phrase-based and syntax-based machine translation. In *Proc. ACL Demos*, pages 19–24.

Jun Xie, Haitao Mi, and Qun Liu. 2011. A novel dependency-to-string model for statistical machine translation. In *Proc. EMNLP*, pages 216–226.

Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. 2013. Max-violation perceptron and forced decoding for scalable MT training. In *Proc. EMNLP*, pages 1112–1123.

Yue Zhang and Stephen Clark. 2011. Syntax-based grammaticality improvement using CCG and guided search. In *Proc. EMNLP*, pages 1147–1157.

Yue Zhang, Graeme Blackwood, and Stephen Clark. 2012. Syntax-based word ordering incorporating a large-scale language model. In *Proc. EACL*, pages 736–746.

Yue Zhang. 2013. Partial-tree linearization: Generalized word ordering for text synthesis. In *Proc. IJCAI*, pages 2232–2238.

Learning Hierarchical Translation Spans

Jingyi Zhang[1,2], **Masao Utiyama**[3], **Eiichro Sumita**[3], **Hai Zhao**[1,2]
[1]Center for Brain-Like Computing and Machine Intelligence, Department of Computer Science and Engineering, Shanghai Jiao Tong Unviersity, Shanghai, 200240, China
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Shanghai Jiao Tong Unviersity, Shanghai, 200240, China
[3]National Institute of Information and Communications Technology
3-5Hikaridai, Keihanna Science City, Kyoto, 619-0289, Japan
zhangjingyizz@gmail.com, mutiyama/eiichiro.sumita@nict.go.jp,
zhaohai@cs.sjtu.edu.cn

Abstract

We propose a simple and effective approach to learn translation spans for the hierarchical phrase-based translation model. Our model evaluates if a source span should be covered by translation rules during decoding, which is integrated into the translation system as soft constraints. Compared to syntactic constraints, our model is directly acquired from an aligned parallel corpus and does not require parsers. Rich source side contextual features and advanced machine learning methods were utilized for this learning task. The proposed approach was evaluated on NTCIR-9 Chinese-English and Japanese-English translation tasks and showed significant improvement over the baseline system.

1 Introduction

The hierarchical phrase-based (HPB) translation model (Chiang, 2005) has been widely adopted in statistical machine translation (SMT) tasks. The HPB translation rules based on the synchronous context free grammar (SCFG) are simple and powerful.

One drawback of the HPB model is the applications of translation rules to the input sentence are highly ambiguous. For example, a rule whose English side is "X1 by X2" can be applied to any word sequence that has "*by*" in them. In Figure 1, this rule can be applied to the whole sentence as well as to "*experiment by tomorrow*".

In order to tackle rule application ambiguities, a few previous works used syntax trees. Chiang (2005) utilized a syntactic feature in the HPB model, which represents if the source span covered by a translation rule is a syntactic constituent. However, the experimental results showed this feature gave no significant improvement. Instead of using the undifferentiated constituency feature, (Marton and Resnik, 2008) defined different soft syntactic features for different constituent types and obtained substantial performance improvement. Later, (Mylonakis and Sima'an, 2011) introduced joint probability synchronous grammars to integrate flexible linguistic information. (Liu et al., 2011) proposed the soft syntactic constraint model based on discriminative classifiers for each constituent type and integrated all of them into the translation model. (Cui et al., 2010) focused on hierarchical rule selection using many features including syntax constituents.

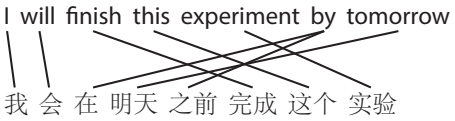


Figure 1: A translation example.

These works have demonstrated the benefits of using syntactic features in the HPB model. However, high quality syntax parsers are not always easily obtained for many languages. Without this problem, word alignment constraints can also be used to guide the application of the rules.

Suppose that we want to translate the English sentence into the Chinese sentence in Figure 1, a translation rule can be applied to the source span "*finish this experiment by tomorrow*". Nonetheless, if a rule is applied to "*experiment by*", then the Chinese translation can not be correctly obtained, because the target span projected from "*ex-*

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 183–188,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

periment by" contains words projected from the source words outside "*experiment by*".

In general, a translation rule projects one continuous source word sequence (source span) into one continuous target word sequence. Meanwhile, the word alignment links between the source and target sentence define the source spans where translation rules are applicable. In this paper, we call a source span that can be covered by a translation rule without violating word alignment links a *translation span*.

Translation spans that have been correctly identified can guide translation rules to function properly, thus (Xiong et al., 2010) attempted to use extra machine learning approaches to determine boundaries of translation spans. They used two separate classifiers to learn the beginning and ending boundaries of translation spans, respectively. A source word is marked as beginning (ending) boundary if it is the first (last) word of a translation span. However, a source span whose first and last words are both boundaries is not always a translation span. In Figure 1, "*I*" is a beginning boundary since it is the first word of translation span "*I will*" and "*experiment*" is an ending boundary since it is the last word of translation span "*finish this experiment*" , but "*I will finish this experiment*" is not a translation span. This happens because the translation spans are nested or hierarchical. Note that (He et al., 2010) also learned phrase boundaries to constrain decoding, but their approach identified boundaries only for monotone translation.

In this paper, taking fully into account that translation spans being nested, we propose an approach to learn hierarchical translation spans directly from an aligned parallel corpus that makes more accurate identification over translation spans.

The rest of the paper is structured as follows: In Section 2, we briefly review the HPB translation model. Section 3 describes our approach. We describe experiments in Section 4 and conclude in Section 5.

2 Hierarchical Phrase-based Translation

Chiang's HPB model is based on a weighted SCFG. A translation rule is like: $X \rightarrow \langle \gamma, \alpha, \sim \rangle$, where X is a nonterminal, γ and α are source and target strings of terminals and nonterminals, and $\sim$ is a one-to-one correspondence between nonterminals in γ and α. The weight of each rule is:

$$w\left(X \rightarrow \langle \gamma, \alpha, \sim \rangle\right) = \prod_t h_t(X \rightarrow \langle \gamma, \alpha, \sim \rangle)^{\lambda_t} \quad (1)$$

where h_t are the features defined on the rules.

Rewriting begins with a pair of linked start symbols and ends when there is no nonterminal left. Let D be a derivation of the grammar, $f\left(D\right)$ and $e\left(D\right)$ be the source and target strings generated by D. D consists of a set of triples $\langle r, i, j \rangle$, each of which stands for applying a rule r on a span $f\left(D\right)_i^j$. The weight of D is calculated as:

$$w\left(D\right) = \prod_{\langle r,i,j \rangle \in D} w\left(r\right) \times P_{lm}(e)^{\lambda_{lm}} \times \exp\left(-\lambda_{wp}\left|e\right|\right) \quad (2)$$

where $w\left(r\right)$ is the weight of rule r, the last two terms represent the language model and word penalty, respectively.

3 Learning Translation Spans

We will describe how to learn translation spans in this section.

3.1 Our Model

We make a series of binary classifiers $\{C_1, C_2, C_3, ...\}$ to learn if a source span $f\left(D\right)_i^j$ should be covered by translation rules during translation. C_k is trained and tested on source spans whose lengths are k, i.e., $k = j - i + 1$.[1]

C_k learns the probability

$$P_k\left(v | f\left(D\right), i, j\right) \quad (3)$$

where $v \in \{0, 1\}$, $v = 1$ represents a rule is applied on $f\left(D\right)_i^j$, otherwise $v = 0$.

Training instances for these classifiers are extracted from an aligned parallel corpus according to Algorithm 1. For example, "*I will*" and "*will finish*" are respectively extracted as positive and negative instances in Figure 1.

Note that our model in Equation 3 only uses the source sentence $f\left(D\right)$ in the condition. This means that the probabilities can be calculated before translation. Therefore, the predicted probabilities can be integrated into the decoder conveniently as soft constraints and no extra time is added during decoding. This enables us to use rich source contextual features and various machine learning methods for this learning task.

[1] We indeed can utilize just one classifier for all source spans. However, it will be difficult to design features for such a classifier unless only boundary word features are adopted. On the contrary, we can fully take advantage of rich information about inside words as we turn to the fixed span length approach.

3.2 Integration into the decoder

It is straightforward to integrate our model into Equation 2. It is extended as

$$w(D) = \prod_{\langle r,i,j\rangle \in D} w(r) \times P_{lm}(e)^{\lambda_{lm}} \times \exp(-\lambda_{wp}|e|) \times P_k(v=1|f(D),i,j)^{\lambda_k} \quad (4)$$

where λ_k is the weight for C_k.

During decoding, the decoder looks up the probabilities P_k calculated and stored before decoding.

Algorithm 1 Extract training instances.

Input: A pair of parallel sentence f_1^n and e_1^m with word alignments A.

Output: Training examples for $\{C_1, C_2, C_3, ...\}$.

```
for i = 1 to n do
  for j = i to n do
    if ∃e_p^q, 1 ≤ p ≤ q ≤ m
    & ∃(k,t) ∈ A, i ≤ k ≤ j, p ≤ t ≤ q
    & ∀(k,t) ∈ A, i ≤ k ≤ j ↔ p ≤ t ≤ q
    then
      f_i^j is a positive instance for C_{j-i+1}
    else
      f_i^j is a negative instance for C_{j-i+1}
    end if
  end for
end for
```

3.3 Classifiers

We compare two machine learning methods for learning a series of binary classifiers.

For the first method, each C_k is individually learned using the maximum entropy (ME) approach (Berger et al., 1996):

$$P_k(v|f(D),i,j) = \frac{\exp\left(\sum_t \mu_t h_t(v, f(D), i, j)\right)}{\sum_{v'} \exp\left(\sum_t \mu_t h_t(v', f(D), i, j)\right)} \quad (5)$$

where h_t is a feature function and μ_t is weight of h_t. We use rich source contextual features: unigram, bigram and trigram of the phrase $[f_{i-3}, ..., f_{j+3}]$.

As the second method, these classification tasks are learned in the continuous space using feed-forward neural networks (NNs). Each C_k has the similar structure with the NN language model (Vaswani et al., 2013). The inputs to the NN are indices of the words: $[f_{i-3}, ..., f_{j+3}]$. Each source word is projected into an N dimensional vector. The output layer has two output neurons, whose values correspond to $P_k(v=0|f(D),i,j)$ and $P_k(v=1|f(D),i,j)$.

For both ME and NN approaches, words that occur only once or never occur in the training corpus are treated as a special word "UNK" (unknown) during classifier training and predicting, which can reduce training time and make the classifier training more smooth.

4 Experiment

We evaluated the effectiveness of the proposed approach for Chinese-to-English (CE) and Japanese-to-English (JE) translation tasks. The datasets officially provided for the patent machine translation task at NTCIR-9 (Goto et al., 2011) were used in our experiments. The detailed training set statistics are given in Table 1. The development and test

		SOURCE	TARGET
CE	#Sents	954k	
	#Words	37.2M	40.4M
	#Vocab	288k	504k
JE	#Sents	3.14M	
	#Words	118M	104M
	#Vocab	150k	273k

Table 1: Data sets.

sets were both provided for CE task while only the test set was provided for JE task. Therefore, we used the sentences from the NTCIR-8 JE test set as the development set. Word segmentation was done by BaseSeg (Zhao et al., 2006; Zhao and Kit, 2008; Zhao et al., 2010; Zhao and Kit, 2011; Zhao et al., 2013) for Chinese and Mecab [2] for Japanese.

To learn the classifiers for each translation task, the training set and development set were put together to obtain symmetric word alignment using GIZA++ (Och and Ney, 2003) and the *grow-diag-final-and* heuristic (Koehn et al., 2003). The source span instances extracted from the aligned training and development sets were used as the training and validation data for the classifiers.

The toolkit Wapiti (Lavergne et al., 2010) was adopted to train ME classifiers using the classical quasi-newton optimization algorithm with limited memory. The NNs are trained by the toolkit NPLM (Vaswani et al., 2013). We chose "rectifier" as the activation function and the logarithmic loss function for NNs. The number of epochs was set to 20. Other parameters were set to default

[2] http://sourceforge.net/projects/mecab/files/

Span length	CE					JE				
		ME		NN			ME		NN	
	Rate	P	N	P	N	Rate	P	N	P	N
1	2.67	0.93	0.63	0.93	0.64	1.08	0.85	0.79	0.86	0.80
2	1.37	0.83	0.70	0.82	0.75	0.73	0.69	0.84	0.71	0.87
3	0.86	0.70	0.80	0.73	0.83	0.52	0.56	0.89	0.63	0.90
4	0.62	0.57	0.81	0.67	0.88	0.36	0.48	0.93	0.54	0.93
5	0.48	0.52	0.90	0.61	0.91	0.26	0.30	0.96	0.47	0.95
6	0.40	0.47	0.91	0.58	0.92	0.20	0.25	0.97	0.41	0.96
7	0.34	0.40	0.93	0.53	0.93	0.16	0.14	0.98	0.33	0.97
8	0.28	0.35	0.94	0.46	0.94	0.13	0	1	0.32	0.97
9	0.22	0.28	0.96	0.37	0.96	0.10	0	1	0.25	0.98
10	0.15	0.21	0.97	0.28	0.97	0.08	0	1	0.23	0.99

Table 2: Classification accuracies. The Rate column represents ratio of positive instances to negative instances; the P and N columns give classification accuracies for positive and negative instances.

values. The training time of one classifier on a 12-core 3.47GHz Xeon X5690 machine was 0.5h (2.5h) using ME (NN) approach for CE task; 1h (4h) using ME (NN) approach for JE task .

The classification results are shown in Table 2. Instead of the undifferentiated classification accuracy, we present separate classification accuracies for positive and negative instances. The big difference between classification accuracies for positive and negative instances was caused by the unbalanced rate of positive and negative instances in the training corpus. For example, if there are more positive training instances, then the classifier will tend to classify new instances as positive and the classification accuracy for positive instances will be higher. In our classification tasks, there are less positive instances for longer span lengths.

Since the word order difference of JE task is much more significant than that of CE task, there are more negative Japanese translation span instances than Chinese. In JE tasks, the ME classifiers C_8, C_9 and C_{10} predicted all new instances to be negative due to the heavily unbalanced instance distribution.

As shown in Table 2, NN outperformed ME approach for our classification tasks. As the span length growing, the advantage of NN became more significant. Since the classification accuracies deceased to be quite low for source spans with more than 10 words, only $\{C_1, ..., C_{10}\}$ were integrated into the HPB translation system.

For each translation task, the recent version of Moses HPB decoder (Koehn et al., 2007) with the training scripts was used as the baseline (Base). We used the default parameters for Moses, and a 5-gram language model was trained on the target side of the training corpus by IRST LM Toolkit [3] with improved Kneser-Ney smoothing. $\{C_1, ..., C_{10}\}$ were integrated into the baseline with different weights, which were tuned by MERT (Och, 2003) together with other feature weights (language model, word penalty,...) under the log-linear framework (Och and Ney, 2002).

	Method	TER	BLEU-n 4	n-gram precisions 1 2 3 4
CE	Base	49.39- -	33.07- -	69.9/40.7/25.8/16.9
	BLM	48.60	33.93	70.0/41.4/26.6/17.6
	ME	49.02-	33.63-	70.0/41.2/26.3/17.4
	NN	48.09++	34.35++	70.1/41.9/27.0/18.0
JE	Base	57.39- -	30.13- -	67.1/38.3/23.0/14.0
	BLM	56.79	30.81	67.7/38.9/23.6/14.5
	ME	56.48	31.01	67.6/39.0/23.8/14.7
	NN	55.96++	31.77++	67.8/39.7/24.6/15.4

Table 3: Translation results. The symbol ++ (- -) represents a significant difference at the $p < 0.01$ level and - represents a significant difference at the $p < 0.05$ level against the BLM.

We compare our method with the baseline and the boundary learning method (BLM) (Xiong et al., 2010) based on Maximum Entropy Markov Models with Markov order 2. Table 3 reports BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) scores. Significance tests are conducted using bootstrap sampling (Koehn, 2004). Our ME classifiers achieve comparable translation improvement with the BLM and NN classifiers enhance translation system significantly compared to others. Table 3 also shows that the relative gain was higher for higher n-grams, which is reasonable since the higher n-grams have higher ambiguities in the translation rule application.

It is true that because of multiple parallel sentences, a source span can be applied with transla-

[3] http://hlt.fbk.eu/en/irstlm

tion rules in one sentence pair but not in another sentence pair. So we used the probability score as a feature in the decoding. That is, we did not use classification results directly but use the probability score for softly constraining the decoding process.

5 Conclusion

We have proposed a simple and effective translation span learning model for HPB translation. Our model is learned from aligned parallel corpora and predicts translation spans for source sentence before translating, which is integrated into the translation system conveniently as soft constraints. We compared ME and NN approaches for this learning task. The results showed that NN classifiers on the continuous space model achieved both higher classification accuracies and better translation performance with acceptable training times.

Acknowledgments

Hai Zhao were partially supported by CSC fund (201304490199), the National Natural Science Foundation of China (Grant No.60903119, Grant No.61170114, and Grant No.61272248), the National Basic Research Program of China (Grant No.2013CB329401), the Science and Technology Commission of Shanghai Municipality (Grant No.13511500200), the European Union Seventh Framework Program (Grant No.247619), and the art and science interdiscipline funds of Shanghai Jiao Tong University, a study on mobilization mechanism and alerting threshold setting for online community, and media image and psychology evaluation: a computational intelligence approach.

References

Adam Berger, Vincent Della Pietra, and Stephen Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270. Association for Computational Linguistics.

Lei Cui, Dongdong Zhang, Mu Li, Ming Zhou, and Tiejun Zhao. 2010. A joint rule selection model for hierarchical phrase-based translation. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 6–11. Association for Computational Linguistics.

Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin K Tsou. 2011. Overview of the patent machine translation task at the ntcir-9 workshop. In *Proceedings of NTCIR*, volume 9, pages 559–578.

Zhongjun He, Yao Meng, and Hao Yu. 2010. Learning phrase boundaries for hierarchical phrase-based translation. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 383–390. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*, pages 388–395.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale crfs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 504–513, Uppsala, Sweden, July. Association for Computational Linguistics.

Lemao Liu, Tiejun Zhao, Chao Wang, and Hailong Cao. 2011. A unified and discriminative soft syntactic constraint model for hierarchical phrase-based translation. In *the Thirteenth Machine Translation Summit*, pages 253–260. Asia-Pacific Association for Machine Translation.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *ACL*, pages 1003–1011.

Markos Mylonakis and Khalil Sima'an. 2011. Learning hierarchical translation structure with linguistic annotations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 642–652. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302. Association for Computational Linguistics.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, pages 223–231.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, Seattle, Washington, USA, October. Association for Computational Linguistics.

Deyi Xiong, Min Zhang, and Haizhou Li. 2010. Learning translation boundaries for phrase-based decoding. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 136–144. Association for Computational Linguistics.

Hai Zhao and Chunyu Kit. 2008. Exploiting unlabeled text with different unsupervised segmentation criteria for chinese word segmentation. *Research in Computing Science*, 33:93–104.

Hai Zhao and Chunyu Kit. 2011. Integrating unsupervised and supervised word segmentation: The role of goodness measures. *Information Sciences*, 181(1):163–183.

Hai Zhao, Chang-Ning Huang, and Mu Li. 2006. An improved chinese word segmentation system with conditional random field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 162–165. Sydney: July.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2010. A unified character-based tagging framework for chinese word segmentation. *ACM Transactions on Asian Language Information Processing (TALIP)*, 9(2):5.

Hai Zhao, Masao Utiyama, Eiichiro Sumita, and Bao-Liang Lu. 2013. An empirical study on word segmentation for chinese machine translation. In *Computational Linguistics and Intelligent Text Processing*, pages 248–263. Springer.

Neural Network Based Bilingual Language Model Growing for Statistical Machine Translation

Rui Wang[1,3,*], **Hai Zhao**[1,3], **Bao-Liang Lu**[1,3], **Masao Utiyama**[2] **and Eiichro Sumita**[2]
[1]Center for Brain-Like Computing and Machine Intelligence,
Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, 200240, China
[2]Multilingual Translation Laboratory, MASTAR Project,
National Institute of Information and Communications Technology,
3-5 Hikaridai, Keihanna Science City, Kyoto, 619-0289, Japan
[3]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
wangrui.nlp@gmail.com, {zhaohai, blu}@cs.sjtu.edu.cn,
{mutiyama, eiichiro.sumita}@nict.go.jp

Abstract

Since larger n-gram Language Model (LM) usually performs better in Statistical Machine Translation (SMT), how to construct efficient large LM is an important topic in SMT. However, most of the existing LM growing methods need an extra monolingual corpus, where additional LM adaption technology is necessary. In this paper, we propose a novel neural network based bilingual LM growing method, only using the bilingual parallel corpus in SMT. The results show that our method can improve both the perplexity score for LM evaluation and BLEU score for SMT, and significantly outperforms the existing LM growing methods without extra corpus.

1 Introduction

'*Language Model (LM) Growing*' refers to adding n-grams outside the corpus together with their probabilities into the original LM. This operation is useful as it can make LM perform better through letting it become larger and larger, by only using a small training corpus.

There are various methods for adding n-grams selected by different criteria from a monolingual corpus (Ristad and Thomas, 1995; Niesler and Woodland, 1996; Siu and Ostendorf, 2000; Siivola et al., 2007). However, all of these approaches need additional corpora. Meanwhile the extra corpora from different domains will not result in better LMs (Clarkson and Robinson, 1997; Iyer et al., 1997; Bellegarda, 2004; Koehn and Schroeder, 2007). In addition, it is very difficult or even impossible to collect an extra large corpus for some special domains such as the TED corpus (Cettolo et al., 2012) or for some rare languages. Therefore, to improve the performance of LMs, without assistance of extra corpus, is one of important research topics in SMT.

Recently, Continues Space Language Model (CSLM), especially Neural Network based Language Model (NNLM) (Bengio et al., 2003; Schwenk, 2007; Mikolov et al., 2010; Le et al., 2011), is being actively used in SMT (Schwenk et al., 2006; Son et al., 2010; Schwenk, 2010; Schwenk et al., 2012; Son et al., 2012; Niehues and Waibel, 2012). One of the main advantages of CSLM is that it can more accurately predict the probabilities of the n-grams, which are not in the training corpus. However, in practice, CSLMs have not been widely used in the current SMT systems, due to their too high computational cost.

Vaswani and colleagues (2013) propose a method for reducing the training cost of CSLM and apply it to SMT decoder. However, they do not show their improvement for decoding speed, and their method is still slower than the n-gram LM. There are several other methods for attempting to implement neural network based LM or translation model for SMT (Devlin et al., 2014; Liu et al., 2014; Auli et al., 2013). However, the decoding speed using n-gram LM is still state-of-the-art one. Some approaches calculate the probabilities of the n-grams n-grams before decoding, and store them in the n-gram format (Wang et al., 2013a; Arsoy et al., 2013; Arsoy et al., 2014). The '*converted CSLM*' can be directly used in SMT. Though more n-grams which are not in the train-

[*]Part of this work was done as Rui Wang visited in NICT.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 189–195,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

ing corpus can be generated by using some of these '*converting*' methods, these methods only consider the monolingual information, and do not take the bilingual information into account.

We observe that the translation output of a phrase-based SMT system is concatenation of phrases from the phrase table, whose probabilities can be calculated by CSLM. Based on this observation, a novel neural network based bilingual LM growing method is proposed using the '*connecting phrases*'. The remainder of this paper is organized as follows: In Section 2, we will review the existing CSLM converting methods. The new neural network based bilingual LM growing method will be proposed in Section 3. In Section 4, the experiments will be conducted and the results will be analyzed. We will conclude our work in Section 5.

2 Existing CSLM Converting Methods

Traditional Backoff N-gram LMs (BNLMs) have been widely used in many NLP tasks (Zhang and Zhao, 2013; Jia and Zhao, 2014; Zhao et al., 2013; Zhang et al., 2012; Xu and Zhao, 2012; Wang et al., 2013b; Jia and Zhao, 2013; Wang et al., 2014). Recently, CSLMs become popular because they can obtain more accurate probability estimation.

2.1 Continues Space Language Model

A CSLM implemented in a multi-layer neural network contains four layers: the input layer projects (first layer) all words in the context h_i onto the projection layer (second layer); the hidden layer (third layer) and the output layer (fourth layer) achieve the non-liner probability estimation and calculate the LM probability $P(w_i|h_i)$ for the given context (Schwenk, 2007).

CSLM is able to calculate the probabilities of all words in the vocabulary of the corpus given the context. However, due to too high computational complexity, CSLM is mainly used to calculate the probabilities of a subset of the whole vocabulary (Schwenk, 2007). This subset is called a *short-list*, which consists of the most frequent words in the vocabulary. CSLM also calculates the sum of the probabilities of all words not included in the short-list by assigning a neuron with the help of BNLM. The probabilities of other words not in the short-list are obtained from an BNLM (Schwenk, 2007; Schwenk, 2010; Wang et al., 2013a).

Let w_i and h_i be the current word and history, respectively. CSLM with a BNLM calculates the probability $P(w_i|h_i)$ of w_i given h_i, as follows:

$$P(w_i|h_i) = \begin{cases} \frac{P_c(w_i|h_i)}{\sum_{w \in V_0} P_c(w|h_i)} P_s(h_i) & \text{if } w_i \in V_0 \\ P_b(w_i|h_i) & \text{otherwise} \end{cases} \quad (1)$$

where V_0 is the short-list, $P_c(\cdot)$ is the probability calculated by CSLM, $\sum_{w \in V_0} P_c(w|h_i)$ is the summary of probabilities of the neuron for all the words in the short-list, $P_b(\cdot)$ is the probability calculated by the BNLM, and

$$P_s(h_i) = \sum_{v \in V_0} P_b(v|h_i). \quad (2)$$

We may regard that CSLM redistributes the probability mass of all words in the short-list, which is calculated by using the n-gram LM.

2.2 Existing Converting Methods

As baseline systems, our approach proposed in (Wang et al., 2013a) only re-writes the probabilities from CSLM into the BNLM, so it can only conduct a convert LM with the same size as the original one. The main difference between our proposed method in this paper and our previous approach is that n-grams outside the corpus are generated firstly and the probabilities using CSLM are calculated by using the same method as our previous approach. That is, the proposed new method is the same as our previous one when no grown n-grams are generated.

The method developed by Arsoy and colleagues (Arsoy et al., 2013; Arsoy et al., 2014) adds all the words in the short-list after the tail word of the i-grams to construct the (i+1)-grams. For example, if the i-gram is "*I want*", then the (i+1)-grams will be "*I want* *", where "*" stands for any word in the short list. Then the probabilities of the (i+1)-grams are calculated using (i+1)-CSLM. So a very large intermediate (i+1)-grams will have to be grown[1], and then be pruned into smaller suitable size using an entropy-based LM pruning method modified from (Stolcke, 1998). The (i+2)-grams are grown using (i+1)-grams, recursively.

[1]In practice, the probabilities of all the target/tail words in the short list for the history i-grams can be calculated by the neurons in the output layer at the same time, which will save some time. According to our experiments, the time cost for Arsoy's growing method is around 4 times more than our proposed method, if the LMs which are 10 times larger than the original one are grown with other settings all the same.

3 Bilingual LM Growing

The translation output of a phrase-based SMT system can be regarded as a concatenation of phrases in the phrase table (except unknown words). This leads to the following procedure:

Step 1. All the n-grams included in the phrase table should be maintained at first.

Step 2. The connecting phrases are defined in the following way.

The w_a^b is a target language phrase starting from the a-th word ending with the b-th word, and $\beta w_a^b \gamma$ is a phrase including w_a^b as a part of it, where β and γ represent any word sequence or none. An i-gram phrase $w_1^k w_{k+1}^i$ $(1 \leq k \leq i-1)$ is a connecting phrase[2], if :

(1) w_1^k is the right (rear) part of one phrase βw_1^k in the phrase table, or

(2) w_{k+1}^i is the left (front) part of one phrase $w_{k+1}^i \gamma$ in the phrase table.

After the probabilities are calculated using C-SLM (Eqs.1 and 2), we combine the n-grams in the phrase table from Step 1 and the connecting phrases from Step 2.

3.1 Ranking the Connecting Phrases

Since the size of connecting phrases is too huge (usually more than one Terabyte), it is necessary to decide the usefulness of connecting phrases for SMT. The more useful connecting phrases can be selected, by ranking the appearing probabilities of the connecting phrases in SMT decoding.

Each line of a phrase table can be simplified (without considering other unrelated scores in the phrase table) as

$$f \;|||\; e \;|||\; P(e|f), \tag{3}$$

where the $P(e|f)$ means the translation probability from $f(source\ phrase)$ to $e(target\ phrase)$, which can be calculated using bilingual parallel training data. In decoding, the probability of a target phrase e appearing in SMT should be

$$P_t(e) = \sum_f P_s(f) \times P(e|f), \tag{4}$$

where the $P_s(f)$ means the appearing probability of a source phrase, which can be calculated using source language part in the bilingual training data.

Using $P_t(e)$[3], we can select the connecting phrases e with high appearing probabilities as the n-grams to be added to the original n-grams. These n-grams are called '*grown n-grams*'. Namely, we build all the connecting phrases at first, and then we use the appearing probabilities of the connecting phrases to decide which connecting phrases should be selected. For an i-gram connecting phrase $w_1^k w_{k+1}^i$, where w_1^k is part of βw_1^k and w_{k+1}^i is part of $w_{k+1}^i \gamma$ (the βw_1^k and $w_{k+1}^i \gamma$ are from the phrase table), the probability of the connecting phrases can be roughly estimated as

$$P_{\text{con}}(w_1^k w_{k+1}^i) = \sum_{k=1}^{i-1} \Big(\sum_{\beta} P_t(\beta w_1^k) \times \sum_{\gamma} P_t(w_{k+1}^i \gamma) \Big). \tag{5}$$

A threshold for $P_{\text{con}}(w_1^k w_{k+1}^i)$ is set, and only the connecting phrases whose appearing probabilities are higher than the threshold will be selected as the grown n-grams.

3.2 Calculating the Probabilities of Grown N-grams Using CSLM

To our bilingual LM growing method, a 5-gram LM and n-gram (n=2,3,4,5) CSLMs are built by using the target language of the parallel corpus, and the phrase table is learned from the parallel corpus.

The probabilities of unigram in the original n-gram LM will be maintained as they are. The n-grams from the bilingual phrase table will be grown by using the '*connecting phrases*' method. As the whole connecting phrases are too huge, we use the ranking method to select the more useful connecting phrases. The distribution of different n-grams (n=2,3,4,5) of the grown LMs are set as the same as the original LM.

The probabilities of the grown n-grams (n=2,3,4,5) are calculated using the 2,3,4,5-CSLM, respectively. If the tail (target) words of the grown n-grams are not in the short-list of C-SLM, the $P_b(\cdot)$ in Eq. 1 will be applied to calculate their probabilities.

[2] We are aware that connecting phrases can be applied to not only two phrases, but also three or more. However the appearing probabilities (which will be discussed in Eq. 5 of next subsection) of connecting phrases are approximately estimated. To estimate and compare probabilities of longer phrases in different lengths will lead to serious bias, and the experiments also showed using more than two connecting phrases did not perform well (not shown for limited space), so only two connecting phrases are applied in this paper.

[3] This $P_t(e)$ hence provides more bilingual information, in comparison with using monolingual target LMs only.

We combine the n-grams (n=1,2,3,4,5) together and re-normalize the probabilities and backoff weights of the grown LM. Finally the original BNLM and the grown LM are interpolated. The entire process is illustrated in Figure 1.

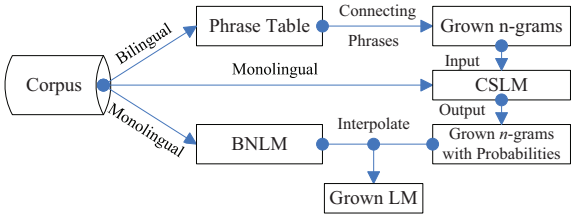


Figure 1: NN based bilingual LM growing.

4 Experiments and Results

4.1 Experiment Setting up

The same setting up of the NTCIR-9 Chinese to English translation baseline system (Goto et al., 2011) was followed, only with various LMs to compare them. The Moses phrase-based SMT system was applied (Koehn et al., 2007), together with GIZA++ (Och and Ney, 2003) for alignment and MERT (Och, 2003) for tuning on the development data. Fourteen standard SMT features were used: five translation model scores, one word penalty score, seven distortion scores, and one LM score. The translation performance was measured by the case-insensitive BLEU on the tokenized test data.

We used the patent data for the Chinese to English patent translation subtask from the NTCIR-9 patent translation task (Goto et al., 2011). The parallel training, development, and test data sets consist of 1 million (M), 2,000, and 2,000 sentences, respectively.

Using SRILM (Stolcke, 2002; Stolcke et al., 2011), we trained a 5-gram LM with the interpolated Kneser-Ney smoothing method using the 1M English training sentences containing 42M words without cutoff. The 2,3,4,5-CSLMs were trained on the same 1M training sentences using CSLM toolkit (Schwenk, 2007; Schwenk, 2010). The settings for CSLMs were: input layer of the same dimension as vocabulary size (456K), projection layer of dimension 256 for each word, hidden layer of dimension 384 and output layer (short-list) of dimension 8192, which were recommended in the CSLM toolkit and (Wang et al., 2013a)[4].

[4]Arsoy used around 55 M words as the corpus, including 84K words as vocabulary, and 20K words as short-list. In this paper, we used the same setting as our previous work, which covers 92.89% of the frequency of words in the training corpus, for all the baselines and our method for fair comparison.

4.2 Results

The experiment results were divided into four groups: the original BNLMs (BN), the CSLM Re-ranking (RE), our previous converting (WA), the Arsoy's growing, and our growing methods. For our bilingual LM growing method, 5 bilingual grown LMs (BI-1 to 5) were conducted in increasing sizes. For the method of Arsoy, 5 grown LMs (AR-1 to 5) with similar size of BI-1 to 5 were also conducted, respectively.

For the CSLM re-ranking, we used CSLM to re-rank the 100-best lists of SMT. Our previous converted LM, Arsoy's grown LMs and bilingual grown LMs were interpolated with the original BNLMs, using default setting of SRILM[5]. To reduce the randomness of MERT, we used two methods for tuning the weights of different SMT features, and two BLEU scores are corresponding to these two methods. The **BLEU-s** indicated that the **same** weights of the BNLM (BN) features were used for all the SMT systems. The **BLEU-i** indicated that the MERT was run **independently** by three times and the average BLEU scores were taken.

We also performed the paired bootstrap resampling test (Koehn, 2004)[6]. Two thousands samples were sampled for each significance test. The marks at the right of the BLEU score indicated whether the LMs were significantly better/worse than the Arsoy's grown LMs with the same IDs for SMT ("++/−−": significantly better/worse at $\alpha = 0.01$, "+/−": $\alpha = 0.05$, no mark: not significantly better/worse at $\alpha = 0.05$).

From the results shown in Table 1, we can get the following observations:

(1) Nearly all the bilingual grown LMs outperformed both BNLM and our previous converted LM on PPL and BLEU. As the size of grown LMs is increased, the PPL always decreased and the BLEU scores trended to increase. These indicated that our proposed method can give better probability estimation for LM and better performance for SMT.

(2) In comparison with the grown LMs in Ar-

[5]In our previous work, we used the development data to tune the weights of interpolation. In this paper, we used the default 0.5 as the interpolation weights for fair comparison.

[6]We used the code available at `http://www.ark.cs.cmu.edu/MT`

Table 1: Performance of the Grown LMs

LMs	n-grams	PPL	BLEU-s	BLEU-i	ALH
BN	73.9M	108.8	32.19	32.19	3.03
RE	N/A	**97.5**	32.34	32.42	N/A
WA	73.9M	104.4	32.60	32.62	3.03
AR-1	217.6M	103.3	32.55	32.75	3.14
AR-2	323.8M	103.1	32.61	32.64	3.18
AR-3	458.5M	103.0	32.39	32.71	3.20
AR-4	565.6M	102.8	32.67	32.51	3.21
AR-5	712.2M	102.5	32.49	32.60	3.22
BI-1	223.5M	101.9	32.81+	33.02+	3.20
BI-2	343.6M	101.0	32.92+	33.11++	3.24
BI-3	464.5M	100.6	33.08++	**33.25**++	3.26
BI-4	571.0M	100.3	**33.15**++	33.12++	3.28
BI-5	705.5M	100.1	33.11++	33.24++	3.31

soy's method, our grown LMs obtained better PPL and significantly better BLEU with the similar size. Furthermore, the improvement of PPL and BLEU of the existing methods became saturated much more quickly than ours did, as the LMs grew.

(3) The last column was the Average Length of the n-grams Hit (ALH) in SMT decoding for different LMs using the following function

$$ALH = \sum_{i=1}^{5} P_{i-gram} \times i, \tag{6}$$

where the P_{i-gram} means the ratio of the i-grams hit in SMT decoding. There were also positive correlations between ALH, PPL and BLEUs. The ALH of bilingual grown LM was longer than that of the Arsoy's grown LM of the similar size. In another word, less back-off was used for our proposed grown LMs in SMT decoding.

4.3 Experiments on TED Corpus

The TED corpus is in special domain as discussed in the introduction, where large extra monolingual corpora are hard to find. In this subsection, we conducted the SMT experiments on TED corpora using our proposed LM growing method, to evaluate whether our method was adaptable to some special domains.

We mainly followed the baselines of the IWSLT 2014 evaluation campaign[7], only with a few modifications such as the LM toolkits and n-gram order for constructing LMs. The Chinese (CN) to English (EN) language pair was chosen, using dev2010 as development data and test2010 as evaluation data. The same LM growing method was applied on TED corpora as on NTCIR corpora. The results were shown in Table 2.

[7] https://wit3.fbk.eu/

Table 2: CN-EN TED Experiments

LMs	n-grams	PPL	BLEU-s
BN	7.8M	87.1	12.41
WA	7.8M	85.3	12.73
BI-1	23.1M	79.2	12.92
BI-2	49.7M	78.3	13.16
BI-3	73.4M	**77.6**	**13.24**

Table 2 indicated that our proposed LM growing method improved both PPL and BLEU in comparison with both BNLM and our previous CSLM converting method, so it was suitable for domain adaptation, which is one of focuses of the current SMT research.

5 Conclusion

In this paper, we have proposed a neural network based bilingual LM growing method by using the bilingual parallel corpus only for SMT. The results show that our proposed method can improve both LM and SMT performance, and outperforms the existing LM growing methods significantly without extra corpus. The connecting phrase-based method can also be applied to LM adaptation.

Acknowledgments

We appreciate the helpful discussion with Dr. Isao Goto and Zhongye Jia, and three anonymous reviewers for valuable comments and suggestions on our paper. Rui Wang, Hai Zhao and Bao-Liang Lu were partially supported by the National Natural Science Foundation of China (No. 60903119, No. 61170114, and No. 61272248), the National Basic Research Program of China (No. 2013CB329401), the Science and Technology Commission of Shanghai Municipality (No. 13511500200), the European Union Seventh Framework Program (No. 247619), the Cai Yuanpei Program (CSC fund 201304490199 and 201304490171), and the art and science interdiscipline funds of Shanghai Jiao Tong University (A study on mobilization mechanism and alerting threshold setting for online community, and media image and psychology evaluation: a computational intelligence approach). The corresponding author of this paper, according to the meaning given to this role by Shanghai Jiao Tong University, is Hai Zhao.

References

Ebru Arsoy, Stanley F. Chen, Bhuvana Ramabhadran, and Abhinav Sethy. 2013. Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition. In *Proceedings of ICASSP-2013*, Vancouver, Canada, May. IEEE.

Ebru Arsoy, Stanley F. Chen, Bhuvana Ramabhadran, and Abhinav Sethy. 2014. Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language*, 22(1):184–192.

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Processings of EMNLP-2013*, pages 1044–1054, Seattle, Washington, USA, October. Association for Computational Linguistics.

Jerome R Bellegarda. 2004. Statistical language model adaptation: review and perspectives. *Speech Communication*, 42(1):93–108. Adaptation Methods for Speech Recognition.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research (JMLR)*, 3:1137–1155, March.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of EAMT-2012*, pages 261–268, Trento, Italy, May.

Philip Clarkson and A.J. Robinson. 1997. Language model adaptation using mixtures and an exponentially decaying cache. In *Proceedings of ICASSP-1997*, volume 2, pages 799–802 vol.2, Munich,Germany.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of ACL-2014*, pages 1370–1380, Baltimore, Maryland, June. Association for Computational Linguistics.

Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin K. Tsou. 2011. Overview of the patent machine translation task at the NTCIR-9 workshop. In *Proceedings of NTCIR-9 Workshop Meeting*, pages 559–578, Tokyo, Japan, December.

Rukmini Iyer, Mari Ostendorf, and Herbert Gish. 1997. Using out-of-domain data to improve in-domain language models. *Signal Processing Letters, IEEE*, 4(8):221–223.

Zhongye Jia and Hai Zhao. 2013. Kyss 1.0: a framework for automatic evaluation of chinese input method engines. In *Proceedings of IJCNLP-2013*, pages 1195–1201, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Zhongye Jia and Hai Zhao. 2014. A joint graph model for pinyin-to-chinese conversion with typo correction. In *Proceedings of ACL-2014*, pages 1512–1523, Baltimore, Maryland, June. Association for Computational Linguistics.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of ACL-2007 Workshop on Statistical Machine Translation*, pages 224–227, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL-2007*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP-2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, J Gauvain, and François Yvon. 2011. Structured output layer neural network language model. In *Proceedings of ICASSP-2011*, pages 5524–5527, Prague, Czech Republic, May. IEEE.

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 2014. A recursive recurrent neural network for statistical machine translation. In *Proceedings of ACL-2014*, pages 1491–1500, Baltimore, Maryland, June. Association for Computational Linguistics.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH-2010*, pages 1045–1048.

Jan Niehues and Alex Waibel. 2012. Continuous space language models using restricted boltzmann machines. In *Proceedings of IWSLT-2012*, pages 311–318, Hong Kong.

Thomas Niesler and Phil Woodland. 1996. A variable-length category-based n-gram language model. In *Proceedings of ICASSP-1996*, volume 1, pages 164–167 vol. 1.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL-2003*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

Eric Sven Ristad and Robert G. Thomas. 1995. New techniques for context modeling. In *Proceedings of ACL-1995*, pages 220–227, Cambridge, Massachusetts. Association for Computational Linguistics.

Holger Schwenk, Daniel Dchelotte, and Jean-Luc Gauvain. 2006. Continuous space language models for statistical machine translation. In *Proceedings of COLING ACL-2006*, pages 723–730, Sydney, Australia, July. Association for Computational Linguistics.

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. 2012. Large, pruned or continuous space language models on a gpu for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, WLM '12, pages 11–19, Montreal, Canada, June. Association for Computational Linguistics.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21(3):492–518.

Holger Schwenk. 2010. Continuous-space language models for statistical machine translation. *The Prague Bulletin of Mathematical Linguistics*, pages 137–146.

Vesa Siivola, Teemu Hirsimki, and Sami Virpioja. 2007. On growing and pruning kneser-ney smoothed n-gram models. *IEEE Transactions on Audio, Speech, and Language*, 15(5):1617–1624.

Manhung Siu and Mari Ostendorf. 2000. Variable n-grams and extensions for conversational speech language modeling. *IEEE Transactions on Speech and Audio*, 8(1):63–75.

Le Hai Son, Alexandre Allauzen, Guillaume Wisniewski, and François Yvon. 2010. Training continuous space language models: some practical issues. In *Proceedings of EMNLP-2010*, pages 778–788, Cambridge, Massachusetts, October. Association for Computational Linguistics.

Le Hai Son, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of NAACL HLT-2012*, pages 39–48, Montreal, Canada, June. Association for Computational Linguistics.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. SRILM at sixteen: Update and outlook. In *Proceedings of INTERSPEECH 2011*, Waikoloa, HI, USA, December.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274, Lansdowne, VA, USA.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Proceedings of INTERSPEECH-2002*, pages 257–286, Seattle, USA, November.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of EMNLP-2013*, pages 1387–1392, Seattle, Washington, USA, October. Association for Computational Linguistics.

Rui Wang, Masao Utiyama, Isao Goto, Eiichro Sumita, Hai Zhao, and Bao-Liang Lu. 2013a. Converting continuous-space language models into n-gram language models for statistical machine translation. In *Proceedings of EMNLP-2013*, pages 845–850, Seattle, Washington, USA, October. Association for Computational Linguistics.

Xiaolin Wang, Hai Zhao, and Bao-Liang Lu. 2013b. Labeled alignment for recognizing textual entailment. In *Proceedings of IJCNLP-2013*, pages 605–613, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Xiao-Lin Wang, Yang-Yang Chen, Hai Zhao, and Bao-Liang Lu. 2014. Parallelized extreme learning machine ensemble based on minmax modular network. *Neurocomputing*, 128(0):31 – 41.

Qiongkai Xu and Hai Zhao. 2012. Using deep linguistic features for finding deceptive opinion spam. In *Proceedings of COLING-2012*, pages 1341–1350, Mumbai, India, December. The COLING 2012 Organizing Committee.

Jingyi Zhang and Hai Zhao. 2013. Improving function word alignment with frequency and syntactic information. In *Proceedings of IJCAI-2013*, pages 2211–2217. AAAI Press.

Xiaotian Zhang, Hai Zhao, and Cong Hui. 2012. A machine learning approach to convert CCGbank to Penn treebank. In *Proceedings of COLING-2012*, pages 535–542, Mumbai, India, December. The COLING 2012 Organizing Committee.

Hai Zhao, Masao Utiyama, Eiichiro Sumita, and Bao-Liang Lu. 2013. An empirical study on word segmentation for chinese machine translation. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 7817 of *Lecture Notes in Computer Science*, pages 248–263. Springer Berlin Heidelberg.

Better Statistical Machine Translation through Linguistic Treatment of Phrasal Verbs

Kostadin Cholakov and Valia Kordoni
Humboldt-Universität zu Berlin, Germany
{kostadin.cholakov,kordonieva}@anglistik.hu-berlin.de

Abstract

This article describes a linguistically informed method for integrating phrasal verbs into statistical machine translation (SMT) systems. In a case study involving English to Bulgarian SMT, we show that our method does not only improve translation quality but also outperforms similar methods previously applied to the same task. We attribute this to the fact that, in contrast to previous work on the subject, we employ detailed linguistic information. We found out that features which describe phrasal verbs as idiomatic or compositional contribute most to the better translation quality achieved by our method.

1 Introduction

Phrasal verbs are a type of multiword expressions (MWEs) and as such, their meaning is not derivable, or is only partially derivable, from the semantics of their lexemes. This, together with the high frequency of MWEs in every day communication (see Jackendoff (1997)), calls for a special treatment of such expressions in natural language processing (NLP) applications. Here, we concentrate on statistical machine translation (SMT) where the word-to-word translation of MWEs often results in wrong translations (Piao et al., 2005).

Previous work has shown that the application of dedicated methods to identify MWEs and then integrate them in some way into the SMT process often improves translation quality. Generally, automatically extracted lexicons of MWEs are employed in the identification step. Further, various integration strategies have been proposed. The so called *static* strategy suggests training the SMT system on corpora in which each MWE is treated as a single unit, e.g. *call_off*. This improves SMT indirectly by improving the alignment between source and target sentences in the training data. Various versions of this strategy are applied in Lambert and Banchs (2005), Carpuat and Diab (2010), and Simova and Kordoni (2013). In all cases there is some improvement in translation quality, caused mainly by the better treatment of separable PVs, such as in *turn the light on*.

Another strategy, which is referred to as *dynamic*, is to modify directly the SMT system. Ren et al. (2009), for example, treat bilingual MWEs pairs as parallel sentences which are then added to training data and subsequently aligned with GIZA++ (Och and Ney, 2003). Other approaches perform feature mining and modify directly the automatically extracted translation table. Ren et al. (2009) and Simova and Kordoni (2013) employ Moses[1] to build and train phrase-based SMT systems and then, in addition to the standard phrasal translational probabilities, they add a binary feature which indicates whether an MWE is present in a given source phrase or not. Carpuat and Diab (2010) employ the same approach but the additional feature indicates the number of MWEs in each phrase. All studies report improvements over a baseline system with no MWE knowledge but these improvements are comparable to those achieved by static methods.

In this article, we further improve the dynamic strategy by adding features which, unlike all previous work, also encode some of the linguistic properties of MWEs. Since it is their peculiar linguistic nature that makes those expressions problematic for SMT, it is our thesis that providing more linguistic information to the translation process will improve it. In particular, we concentrate on a specific type of MWEs, namely phrasal verbs (PVs). We add 4 binary features to the translation table which indicate not only the presence of a PV but also its *transitivity*, *separability*, and *idiomaticity*. We found that PVs are very suitable for this study since we can easily extract the necessary informa-

[1] http://www.statmt.org/moses/

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 196–201,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

tion from various language resources.

To prove our claim, we perform a case study with an English to Bulgarian SMT system. Bulgarian lacks PVs in the same form they appear in English. It is often the case that an English PV is translated to a single Bulgarian verb. Such many-to-one mappings cause the so called *translation asymmetries* which make the translation of PVs very problematic.

We perform automated and manual evaluations with a number of feature combinations which show that the addition of all 4 features proposed above improves translation quality significantly. Moreover, our method outperforms static and dynamic methods previously applied to the same test data. A notable increase in performance is observed for separable PVs where the verb and the particle(s) were not adjacent in the input English sentence as well as for idiomatic PVs. This clearly demonstrates the importance of linguistic information for the proper treatment of PVs in SMT.

We would like to point out that we view the work presented here as a preliminary study towards a more general linguistically informed method for handling similar types of translation asymmetries. The experiments with a single phenomenon, namely PVs, serve as a case study the purpose of which is to demonstrate the validity of our approach and the crucial role of properly integrated linguistic information into SMT. Our work, however, can be immediately extended to other phenomena, such as collocations and noun compounds.

The remainder of the paper is organised as follows. Section 2 describes the asymmetries caused by PVs in English to Bulgarian translation. Section 3 provides details about the resources involved in the experiments. Section 4 describes our method and the experimental setup. Section 5 presents the results and discusses the improvements in translation quality achieved by the method. Sections 6 concludes the paper.

2 Translation Asymmetries

We will first illustrate the main issues which arise when translating English PVs into Bulgarian. For more convenience, the Bulgarian phrases are transcribed with Latin letters.

An English PV is usually mapped to a single Bulgarian verb:

(1) Toj *otmeni* sreshtata.
he cancelled meeting-the
'He *called off* the meeting.'

In the example above the PV *called off* has to be mapped to the single Bulgarian verb *otmeni*, i.e. there is many-to-one mapping. Other cases require a many-to-many type of mapping. One such case is the mapping of an English PV to a 'da'-construction in Bulgarian. Such constructions are very frequent in Bulgarian every day communication since they denote complex verb tenses, modal verb constructions, and subordinating conjunctions:

(2) Toj trjabva da skasa s neja.
he should break off with her
'He should *break off* with her.'

Here, *da skasa* should be mapped to the PV *break off*. Other such cases include Bulgarian reflexive verb constructions.

Note that such many-to-many mappings in the case of Bulgarian pose an additional challenge for the SMT system because, for a good translation, it needs to guess whether to add a 'da' particle or not which further complicates the treatment of PVs. Also, Bulgarian is a language with rich morphology and often translations with very good semantic quality lack the proper morphological inflection. This affects negatively both automated and manual evaluation of translation quality.

3 Language Resources

We employ the data used in the studies reported in Simova and Kordoni (2013). The authors experimented with both static and dynamic methods for handling PVs in an English to Bulgarian SMT system. This allows us to compare the performance of our linguistically informed approach to that of methods which do not make use of the linguistic properties of PVs.

The data for the experiments are derived from the SeTimes news corpus[2] which contains parallel news articles in English and 9 Balkan languages. The training data consist of approximately 151,000 sentences. Another 2,000 sentences are used for the tuning. The test set consists of 800 sentences, 400 of which contain one or more in-

[2] http://www.setimes.com

stances of PVs. There are 138 unique PVs with a total of 403 instances in the test data. Further, a language model for the target language is created based on a 50 million words subset of the Bulgarian National Reference Corpus.[3] All English data are POS tagged and lemmatised using the TreeTagger (Schmid, 1994). For Bulgarian, these tasks were performed with the BTB-LPP tagger (Savkov et al., 2011).

Simova and Kordoni (2013) create automatically a lexicon containing English PVs. It is employed for the identification of such verbs in the data used in the experiments. The lexicon is constructed from a number of resources: the English Phrasal Verbs section of Wiktionary,[4] the Phrasal Verb Demon dictionary,[5] the CELEX Lexical Database (Baayen et al., 1995), WordNet (Fellbaum, 1998), the COMLEX Syntax dictionary (Macleod et al., 1998), and the gold standard data used for the experiments in McCarthy et al. (2003) and Baldwin (2008). English PVs are identified in the data using the jMWE library (Kulkarni and Finlayson, 2011) as well as a post-processing module implemented in the form of a constrained grammar (Karlsson et al., 1995) which filters out spurious PV candidates. For the identification of PVs, Simova and Kordoni (2013) report 91% precision (375 correct instances found) and a recall score of 93% for the 800 test sentences.

The Moses toolkit is employed to build a factored phrase-based translation model which operates on lemmas and POS tags. Given the rich Bulgarian morphology, the use of lemma information instead of surface word forms allows for a better mapping between source and target translation equivalents. The parallel data are aligned with GIZA++. Further, 2 5-gram language models are built using the SRILM toolkit[6] on the monolingual Bulgarian data to model lemma and POS n-gram information. Note that the Bulgarian POS tags are quite complex, so they can account for a variety of morphological phenomena. Automated translation is performed by mapping English lemmas and POS tags to their Bulgarian equivalents and then generating the proper Bulgarian word form by using lemma and POS tag information.

[3] http://webclark.org/

[4] http://en.wiktionary.org/wiki/Category:English_phrasal_verbs

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

[6] http://www-speech.sri.com/projects/srilm/

	1	0
feature 1	PV present	no PV
feature 2	transitive	intransitive
feature 3	separable	inseparable
feature 4	idiomatic	(semi-)comp.

Table 1: Values for the 4 new features.

4 Addition of Linguistic Features

The resources from which the PV lexicon is constructed also contain various types of linguistic information. Wiktionary provides the most details since the entries there contain information about the valency of the verb (transitive vs intransitive) and whether a particle can be separated from the PV in particle verb constructions. Consider *fell off his bike* and **fell his bike off* vs *turn the engine on* and *turn on the engine*.

Further, Wiktionary indicates whether a given PV is compositional or idiomatic in nature. The meaning of (semi-)compositional PVs can be (partially) derived from the meaning of their lexemes, e.g. *carry in*. The degree of compositionality affects the productivity with which verbs and particles combine. Verbs with similar semantics often combine with the same particle, e.g. *bring/carry in*. This is not the case for fully idiomatic PVs, e.g. *get/*obtain over*. Therefore, the notion of compositionality plays a very important role in the treatment of PVs and MWEs in general. The dataset described in McCarthy et al. (2003) also indicates whether a PV is idiomatic or not.

We were able to acquire the PV lexicon and we augmented it with the information obtained from the various resources. Then, once the system is trained, we add 4 binary features to each entry in the automatically created translation table. The values those features take are shown in Table 1. If a given property is not specified for some PV in the lexicon, the value of the corresponding feature is 0. Naturally, if no PV is identified in a source phrase, the value of all 4 features is 0. This is different from previous work where only one feature is added, indicating the presence of a PV. By adding those new features, we want to bias the SMT system towards using phrases that do not "split" PVs during decoding.

	with PVs		no PVs		all	
	bleu	nist	bleu	nist	bleu	nist
baseline	0.244	5.97	0.228	5.73	0.237	6.14
static	0.246	**6.02**	0.230	**5.76**	0.239	**6.18**
dynamic-1	0.250	5.92	0.226	5.54	0.244	6.02
dynamic-4	**0.267**	6.01	**0.232**	5.74	**0.256**	6.16

Table 2: Automatic evaluation of translation quality.

	good	acceptable	incorrect
baseline	0.21	0.41	0.38
static	0.25	0.5	0.25
dynamic-1	0.24	0.51	0.25
dynamic-4	**0.3**	0.5	0.2

Table 3: Manual evaluation of translation quality.

5 Results and Discussion

Automatic Evaluation. Table 2 presents the results from the automatic evaluation, in terms of BLEU (Papineni et al., 2002) and NIST (Doddington, 2002) scores, of 4 system setups. The baseline has no MWE knowledge, while the static and the dynamic-1 system setups are reproduced from the experiments described in Simova and Kordoni (2013). Dynamic-1 includes only a single binary feature which indicates the presence of a PV while our method, dynamic-4, includes the 4 features described in Table 1.

Our method outperforms all other setups in terms of BLEU score, thus proving our point that adding features describing the linguistic properties of PVs improves SMT even further. Also, the results for the 400 sentences without PVs show that the 4 new features do not have a negative impact for PV-free contexts.

In terms of NIST the static strategy consistently performs best, followed closely by our method. NIST is a measure which weights the translated n-grams according to their informativeness. Due to the nature of this measure, less frequent correctly translated n-grams are given more weight in the evaluation process because NIST considers them "more informative". Such less frequent n-grams, or in our case PVs, are likely to be captured better by the static setup. Therefore, this setup achieves the highest NIST scores. This fact also suggests that dynamic and static strategies influence the SMT process in different ways, with our method tending to capture more frequent (and thus less informative) n-grams. Interestingly, the other dynamic method, dynamic-1, has the worst performance of all setups in terms of NIST.

Manual evaluation. To get a better insight on how the different setups deal with the translation of PVs, we also performed a manual evaluation. A native speaker of Bulgarian was asked to judge the translations of PVs for the 375 test sentences in which such verbs were correctly identified during the identification step. The human subject takes into account the target PV and a limited context around it and judges the translation as:

- *good* - correct translation of the PV, correct verb inflection
- *acceptable* - correct translation of the PV but wrong inflection, or wrongly built *da-* or reflexive construction
- *incorrect* - wrong translation which changes the meaning of the sentence

Table 3 shows the results. Our method dynamic-4 produces more good translations and less incorrect ones than all other setups. This illustrates further the benefits of adding linguistic features to the translation model. The results achieved by the static approach are attributed to the better handling of separable PVs in sentences where the particle was not adjacent to the verb. The dynamic-1 approach and the baseline often interpret the particle literally in such cases which leads to almost twice the amount of wrong translations. Our method, on the other hand, performs slightly lower than the static approach in this respect but still much better than the other 2 setups.

Compared to dynamic-1 and the baseline, the static approach also handles better idiomatic PVs but performs slightly worse for sentences with compositional PVs. However, the addition of a specific feature to encode idiomaticity in the translation model enables our method dynamic-4 to achieve the best performance for idiomatic PVs while still handling successfully many compositional PVs. To summarise, the improved results of our method in comparison to previous work are attributed to the better handling of separable PVs which occur in a split form and even more to the improved ability to differentiate between compositional and idiomatic PVs.

Feature combinations. Our method performs best when all 3 linguistic features described above

are taken into account by the SMT system. However, we also experimented with different combinations of those features in order to get some insight of the way each feature influences the translation quality. Adding only the feature denoting verb transitiveness did not lead to any significant improvement compared to the dynamic-1 setup. Also, the combination which leaves out this feature and uses the remaining ones ranks second, achieving only a slightly worse performance than dynamic-4, the setup in which all features are employed. It seems that the transitiveness feature does not contribute much to the task at hand. Adding only the feature denoting separable vs inseparable PVs and adding only the one denoting idiomaticity led to results slightly higher than those of the dynamic-1 and static setups but still, those results were significantly lower than the ones presented in Tables 2 and 3.

6 Conclusion and Outlook

In this article, we showed that the addition of linguistically informative features to a phrase-based SMT model improves the translation quality of a particular type of MWEs, namely phrasal verbs. In a case study involving SMT from English to Bulgarian, we showed that adding features which encode not only the presence of a PV in a given phrase but also its transitiveness, separability, and idiomaticity led to better translation quality compared to previous work which employs both static and dynamic strategies.

In future research, we will extend our method to other language pairs which exhibit the same type of translation asymmetries when it comes to PVs. Such language pairs include, among others, English-Spanish and English-Portuguese.

Further, we will apply our linguistically informed method to other phenomena which cause similar issues for SMT. Immediate candidate phenomena include other types of MWEs, collocations, and noun compounds. When it comes to MWEs, we will pay special attention to the compositionality aspect since it seems to have contributed most to the good performance achieve by our method in the study presented here.

References

R H Baayen, R Piepenbrock, and L. Gulikers. 1995. The CELEX lexical database (CD-ROM).

Timothy Baldwin. 2008. A resource for evaluating the deep lexical acquisition of english verb-particle constructions. In *Proceedings of the LREC 2008 Workshop: Towards a Shared Task for Multiword Expressions*, pages 1–2, Marakesh, Morocco.

Marine Carpuat and Mona Diab. 2010. Task-based evaluation of multiword expressions: a pilot study in statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics.*, HLT '10., pages 242–245, Stroudsburg, PA, USA. Association for Computational Linguistics.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145, San Francisco, CA, USA.

Christiane Fellbaum. 1998. *WordNet: An electronic lexical database*. The MIT press.

Ray Jackendoff. 1997. *The Architecture of the Language Faculty*. MIT Press, Cambridge, MA.

Fred Karlsson, Atro Voutilainen, Juha Heikkila, and Arto Anttila. 1995. Constraint grammar: A language-independent system for parsing unrestricted text. *Natural Language Processing*, 4.

Nidhi Kulkarni and Mark Alan Finlayson. 2011. JMWE – a Java toolkit for detecting multiword expressions. In *Proceedings of the 2011 Workshop on Multiword Expressions*, pages 122–124.

Patrik Lambert and Rafael Banchs. 2005. Data inferred multi-word expressions for statistical machine translation. In *Proceedings of the X Machine Translation Summit*, pages 396–403.

Catherine Macleod, Adam Meyers, and Ralph Grishman, 1998. *COMLEX Syntax Reference Manual*. New York University.

Diana McCarthy, B Keller, and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: analysis, acquisition and treatment*, Sapporo, Japan.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Stroudsburg, PA, USA.

Scott Songlin Piao, Paul Rayson, and and Tony McEnery Dawn Archer. 2005. Comparing and combining a semantic tagger and a statistical tool for MWE extraction. *Comuter Speech and Language*, 19(4):378–397.

Zhixiang Ren, Yajuan Lu, Jie Cao, Qun Liu, and Yun Huang. 2009. Improving statistical machine translation using domain bilingual multiword expressions. In *Proceedings of the ACL Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*, pages 47–54, Singapore.

Aleksandar Savkov, Laska Laskova, Petya Osenova, Kiril Simov, and Stanislava Kancheva. 2011. A web-based morphological tagger for Bulgarian. In *Proceedings of the Sixth International Conference on Natural Language Processing, Multilinguality*, pages 126–137, Bratislava, Slovakia.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK.

Iliana Simova and Valia Kordoni. 2013. Improving English-Bulgarian statistical machine translation by phrasal verb treatment. In *Proceedings of MT Summit XIV Workshop on Multi-word Units in Machine Translation and Translation Technology*, Nice, France.

Fitting Sentence Level Translation Evaluation with Many Dense Features

Miloš Stanojević and **Khalil Sima'an**
Institute for Logic, Language and Computation
University of Amsterdam
Science Park 107, 1098 XG Amsterdam, The Netherlands
{m.stanojevic,k.simaan}@uva.nl

Abstract

Sentence level evaluation in MT has turned out far more difficult than corpus level evaluation. Existing sentence level metrics employ a limited set of features, most of which are rather sparse at the sentence level, and their intricate models are rarely trained for ranking. This paper presents a simple linear model exploiting 33 relatively dense features, some of which are novel while others are known but seldom used, and train it under the *learning-to-rank* framework. We evaluate our metric on the standard WMT12 data showing that it outperforms the strong baseline METEOR. We also analyze the contribution of individual features and the choice of training data, language-pair vs. target-language data, providing new insights into this task.

1 Introduction

Evaluating machine translation (MT) output at the *sentence/ segment level* has turned out far more challenging than corpus/ system level. Yet, sentence level evaluation can be useful because it allows fast, fine-grained analysis of system performance on individual sentences.

It is instructive to contrast two widely used metrics, METEOR (Michael Denkowski and Alon Lavie, 2014) and BLEU (Papineni et al., 2002), on sentence level evaluation. METEOR constantly shows better correlation with human ranking than BLEU (Papineni et al., 2002). Arguably, this shows that sentence level evaluation demands finer grained and trainable models over *less sparse features*. Ngrams, the core of BLEU, are sparse at the sentence level, and a mismatch for longer ngrams implies that BLEU falls back on shorter ngrams. In contrast, METEOR has a trainable model and incorporates a small, yet wider set of features that are less sparse than ngrams. We think that METEOR's features and its training approach *only suggest* that sentence level evaluation should be treated as a modelling challenge. This calls for questions such as what model, what features and what training objective are better suited for modelling sentence level evaluation.

We start out by explicitly formulating sentence level evaluation as the problem of *ranking* a set of competing hypothesis. Given data consisting of human ranked system outputs, the problem then is to formulate an easy to train model for ranking. One particular existing approach (Ye et al., 2007) looks especially attractive because we think it meshes well with a range of effective techniques for *learning-to-rank* (Li, 2011).

We deliberately select a linear modelling approach inspired by *RankSVM* (Herbrich et al., 1999), which is easily trainable for ranking and allows analysis of the individual contributions of features. Besides presenting a new metric and a set of known, but also a set of novel features, we target three questions of interest to the MT community:

- What kind of features are more helpful for sentence level evaluation?
- How does a simple linear model trained for ranking compare to the well-developed metric METEOR on sentence level evaluation?
- Should we train the model for each language pair separately or for a target language?

Our new metric dubbed BEER[1] outperforms METEOR on WMT12 data showing the effectiveness of dense features in a learning-to-rank framework. The metric and the code are available as free software[2].

2 Model

Our model is a linear combination of features trained for ranking similar to RankSVM (Herbrich et al., 1999) or, to readers familiar with SMT system tuning, to PRO tuning (Hopkins and May, 2011):

$$score(sys) = \vec{w} \cdot \vec{x}_{sys}$$

where $\vec{w}$ represents a weight vector and $\vec{x}_{sys}$ a vector of feature values for system output sys. Looking at evaluation as a ranking problem, we contrast (at least) two system translations $good$ and bad for the same source sentence. Assuming that $humanRank(good) > humanRank(bad)$ as ranked

[1] BEER participated on WMT14 evaluation metrics task where it was the highest scoring sentence level evaluation metric on average over all language pairs (Stanojević and Sima'an, 2014)

[2] https://github.com/stanojevic/beer

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 202–206,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

by human judgement, we expect metric $score(\cdot)$ to fulfill $score(good) > score(bad)$:

$$\begin{aligned} \vec{\boldsymbol{w}} \cdot \vec{\boldsymbol{x}}_{good} &> \vec{\boldsymbol{w}} \cdot \vec{\boldsymbol{x}}_{bad} \quad \Leftrightarrow \\ \vec{\boldsymbol{w}} \cdot \vec{\boldsymbol{x}}_{good} - \vec{\boldsymbol{w}} \cdot \vec{\boldsymbol{x}}_{bad} &> 0 \quad \Leftrightarrow \\ \vec{\boldsymbol{w}} \cdot (\vec{\boldsymbol{x}}_{good} - \vec{\boldsymbol{x}}_{bad}) &> 0 \quad \wedge \\ \vec{\boldsymbol{w}} \cdot (\vec{\boldsymbol{x}}_{bad} - \vec{\boldsymbol{x}}_{good}) &< 0 \end{aligned}$$

The two feature vectors $(\vec{\boldsymbol{x}}_{good} - \vec{\boldsymbol{x}}_{bad})$ and $(\vec{\boldsymbol{x}}_{bad} - \vec{\boldsymbol{x}}_{good})$ can be considered as positive and negative instances for training our linear classifier. For training this model, we use Logistic Regression from the Weka toolkit (Hall et al., 2009).

3 Features

Generally speaking we identify *adequacy* and *fluency* features. For both types we devise far less sparse features than word ngrams.

Adequacy features We use precision P, recall R and F1-score F as follows:

P_{func}, R_{func}, F_{func} on matched function words

P_{cont}, R_{cont}, F_{cont} on matched content words

P_{all}, R_{all}, F_{all} on matched words of any type

P_{char}, R_{char}, F_{char} matching of the char ngrams

By differentiating between function and non-function words, our metric weighs each kind of words according to importance for evaluation. Matching character ngrams, originally proposed in (Yang et al., 2013), rewards certain translations even if they did not get the morphology completely right. Existing metrics use stemmers for this, but using character ngrams is independent of the availability of a good quality stemmer. Higher-order character ngrams have less risk of sparse counts than word ngrams. In our experiments we used char ngrams for n up to 6, which makes the total number of adequacy features 27.

Fluency features To evaluate word order we follow (Isozaki et al., 2010; Birch and Osborne, 2010) in representing reordering as a permutation π over $[1..n]$ and then measuring the distance to the ideal monotone permutation $\langle 1, 2, \cdots, n \rangle$. We present a novel approach based on factorization into *permutation trees (PETs)* (Zhang and Gildea, 2007), and contrast it with Kendall τ (Birch and Osborne, 2010; Isozaki et al., 2010). PETs are factorizations of permutations, which allows for an abstract and *less sparse* view of word order as exemplified next. Kendall score was regularly shown to have high correlation with human judgment on distant language pairs (Isozaki et al., 2010; Birch and Osborne, 2010).

Features based on PETs We informally review PETs in order to exploit them for novel ordering features. We refer the reader to (Zhang and Gildea, 2007) and (Maillette de Buy Wenniger and Sima'an, 2011) for a formal treatment of PETs and efficient factorization algorithms.

A PET of permutation π is a tree organization of π's *unique, atomic building blocks*, called *operators*. Every operator on a PET node is an *atomic permutation* (not factorizing any further),[3] and it stands for the permutation of the direct children of that node. Figure 1a shows an example PET that has one 4-branching node with operator $\langle 2, 4, 1, 3 \rangle$, two binary branching nodes of which one decorated with the inverted operator $\langle 2, 1 \rangle$ and another with the monotone $\langle 1, 2 \rangle$.

PETs have two important properties making them attractive for measuring order difference: firstly, order difference is measured on the operators – the *atomic reordering building blocks* of the permutation, and secondly, the operators on higher level nodes capture *hidden ordering patterns* that cannot be observed without factorization. Statistics over ordering patterns in PETs are far less sparse than word or character ngram statistics.

Intuitively, among the atomic permutations, the binary monotone operator $\langle 1, 2 \rangle$ signifies no ordering difference at all, whereas the binary inverted $\langle 2, 1 \rangle$ signifies the shortest unit of order difference. Operators of length four like $\langle 2, 4, 1, 3 \rangle$ (Wu, 1997) are presumably more complex than $\langle 2, 1 \rangle$, whereas operators longer than four signify even more complex order difference. Therefore, we devise possible branching feature functions over the operator length for the nodes in PETs:

- factor 2 - with two features: $\Delta_{[\,]}$ and $\Delta_{<>}$ (there are no nodes with factor 3 (Wu, 1997))
- factor 4 - feature $\Delta_{=4}$
- factor bigger than 4 - feature $\Delta_{>4}$

Consider permutations $\langle 2, 1, 4, 3 \rangle$ and $\langle 4, 3, 2, 1 \rangle$, none of which has exactly matching ngrams beyond unigrams. Their PETs are in Figures 1b and 1c. Intuitively, $\langle 2, 1, 4, 3 \rangle$ is somewhat less scrambled than $\langle 4, 3, 2, 1 \rangle$ because it has at least some position in correct order. These "abstract ngrams" pertaining to correct ordering of full phrases could be counted using $\Delta_{[\,]}$ which would recognize that on top of the PET in 1b there is a binary monotone node, unlike the PET in Figure 1c which has no monotone nodes at all.

Even though the set of operators that describe a permutation is unique for the given permutation, the ways in which operators are combined (the derivation tree) is not unique. For example, for the fully monotone

[3] For example $\langle 2, 4, 1, 3 \rangle$ is atomic whereas $\langle 4, 3, 2, 1 \rangle$ is not. The former does not contain any contiguous sub-ranges of integers whereas the latter contains sub-range $\{2, 3, 4\}$ in reverse order $\langle 4, 3, 2 \rangle$, which factorizes into two binary inverting nodes cf. Fig. 1c.

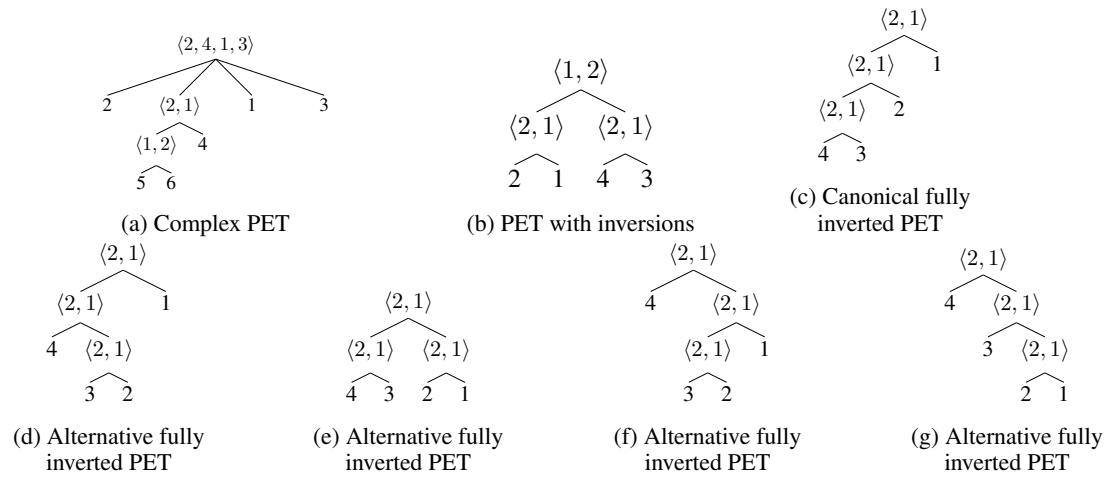


Figure 1: Examples of PETs

permutation $\langle 4, 3, 2, 1\rangle$ there are 5 possible derivations (PETs) presented in Figures 1c, 1d, 1e, 1f and 1g. The features on PETs that we described so far look at the operators independently (they treat a derivation as a set of operators) so differenct derivations do not influence the score–whichever derviation we use we will get the same feature score. However, the number of derivations might say something about the *goodness* of the permutation. Similar property of permutations was found to be helpful earlier in (Mylonakis and Sima'an, 2008) as an ITG prior for learning translation rule probabilities.

Permutations like $\langle 3, 2, 1, 4\rangle$ and $\langle 2, 4, 3, 1\rangle$ have the same set of operators, but the former factorizes into more PETs than the latter because $\langle 4, 3\rangle$ must group first before grouping it with 2 and then 1 in $\langle 2, 4, 3, 1\rangle$. The "freedom to bracket" in different ways could be a signal of better grouping of words (even if they have inverted word order). Hence we exploit one more feature:

Δ_{count} the ratio between the number of alternative PETs for the given permutation, to the number of PETs that could be built if permutation was perfectly grouped (fully monotone or fully inverted).

Finding the number of PETs that could be built does not require building all PETs or encoding them in the chart. The number can be computed directly from the canonical left-branching PET. Since multiple different PETs appear only in cases when there is a sequence of more than one node that is either $\langle 1, 2\rangle$ or $\langle 2, 1\rangle$ (Zhang et al., 2008), we can use these sequences to predict the number of PETs that could be built. Let X represent a set of sequences of the canonical derivation. The number of PETs is computed in the following way:

$$\#PETs = \prod_{x \in X} Cat(|x|) \quad (1)$$

$$Cat(n) = \frac{1}{n+1}\binom{2n}{n} \quad (2)$$

where $Cat(\cdot)$ is a Catalan number. The proof for this formula is beyond the scope of this paper. The reader can consider the example of the PET in Figure 1c. That derivation has one sequence of monotone operators of length 3. So the number of PETs that could be built is $Cat(3) = 5$.

4 Experiments

We use human judgments from the WMT tasks: WMT13 is used for training whereas WMT12 for testing. The baseline is METEOR's latest version (Michael Denkowski and Alon Lavie, 2014), one of the best metrics on sentence level. To avoid contaminating the results with differences with METEOR due to resources, we use the same alignment, tokenization and lowercasing (*-norm* in METEOR) algorithms, and the same tables of function words, synonyms, paraphrases and stemmers.

Kendall τ correlation is borrowed from WMT12 (Callison-Burch et al., 2012):

$$\tau = \frac{\#concordant - \#discordant - \#ties}{\#concordant + \#discordant + \#ties}$$

$\#concordant$ represents the number of pairs ordered in the same way by metric and by human, $\#discordant$ the number of opposite orderings and $\#ties$ the number of tied rankings by metric.

Beside testing our full metric BEER, we perform experiments where we remove one kind of the following features at a time:

1. char n-gram features (P, R and F-score)
2. all word features (P, R and F-score for all, function and content words),
3. all function and content words features
4. all F-scores (all words, function words, content words, char ngrams)

metric	en-cs	en-fr	en-de	en-es	cs-en	fr-en	de-en	es-en	avg τ
BEER without char features	0.124	0.178	0.168	0.149	0.121	0.17	0.179	0.078	0.146
BEER without all word features	0.184	0.237	0.223	0.217	0.192	0.209	0.243	0.199	0.213
BEER without all F-scores	0.197	0.243	0.219	0.22	0.177	0.227	0.254	0.211	0.219
METEOR	0.156	0.252	0.173	0.202	0.208	0.249	0.273	0.246	0.22
BEER without PET features	0.202	0.248	0.243	0.225	0.198	0.249	0.268	0.234	0.233
BEER without function words	0.2	0.245	0.231	0.227	0.189	0.268	0.267	0.253	0.235
BEER without fluency features	0.201	0.248	0.236	0.223	0.202	0.257	0.283	0.243	0.237
BEER without Kendall τ	0.205	0.246	0.244	0.227	0.202	0.257	0.282	0.248	0.239
BEER full	0.206	0.245	0.244	0.23	0.198	0.263	0.283	0.245	0.239

Table 1: Kendall τ scores on WMT12 data

5. PET features
6. Kendall τ features
7. all fluency features (PET and Kendall τ)

Table 1 shows the results sorted by their average Kendall τ correlation with human judgment.

5 Analysis

Given these experimental results, we are coming back to the questions we asked in the introduction.

5.1 What kind of features are more helpful for sentence level evaluation?

Fluency vs. Adequacy The fluency features play a smaller role than adequacy features. Apparently, many SMT systems participating in this task have rather similar reordering models, trained on similar data, which makes the fluency features not that discriminative relative to adequacy features. Perhaps in a different application, for example MT system tuning, the reordering features would be far more relevant because ignoring them would basically imply disregarding the importance of the reordering model in MT.

Character vs. Word features We observe that, precision, recall and F-score on *character ngrams* are crucial. We think that this shows that less sparse features are important for sentence level evaluation. The second best features are *word features*. Without word features, BEER scores just below METEOR, which suggests that word boundaries play a role as well. In contrast, differentiating between *function and content words* does not seem to be important.

PETs vs. Kendall τ Despite the smaller role for reordering features we can make a few observations. Firstly, while PETs and Kendall seem to have similar effect on English-Foreign cases, in all four cases of Foreign-English PETs give better scores. We hypothesize that the quality of the permutations (induced between system output and reference) is better for English than for the other target languages. Discarding PET features has far larger impact than discarding Kendall. Most interestingly, for de-en it makes the difference in outperforming METEOR. In many cases discarding Kendall τ improves the BEER score, likely because it conflicts with the PET features that are found more effective.

5.2 Is a linear model sufficient?

A further insight, from our perspective, is that *F-score* features constitute a crucial set of features, *even when the corresponding precision and recall features are included*. Because our model merely allows for linear interpolation, whereas F-score is a non-linear function of precision and recall, we think this suggests that a non-linear interpolation of precision and recall is useful.[4] By formulating the evaluation as a ranking problem it is relatively easy to "upgrade" for using non-linear models while using the same (or larger) set of features.

5.3 Train for the language pair or only for the target language?

All our models were trained for each language pair. This is not the case with many other metrics which train their models for each target language instead of language pair. We contrast these two settings in Table 2. Training for each language pair separately does not give significant improvement over training for the target language only. A possible reason could be that by training for the target language we have more training data (in this case four times more).

Train for	cs-en	fr-en	de-en	es-en	avg τ
target lang	0.199	0.257	0.273	0.248	0.244
lang pair	0.198	0.263	0.283	0.245	0.247

Table 2: Kendall τ scores on WMT12 for different training data

5.4 BEER vs. METEOR

The results across individual language pairs are mostly consistent with the averages with a few exceptions. BEER outperforms METEOR in five out of eight language pairs, ties at one (the difference is only 0.001 on es-en) and loses in two (en-fr and cs-en). In some cases BEER is better than METEOR by a large margin (see, e.g., en-cs, en-de).

[4] Interestingly, METEOR tunes β in F_β.

6 Conclusion

In this work we show that combining less sparse features at the sentence level into a linear model that is trained on ranking we can obtain state-of-the-art results. The analysis of the results shows that features on character ngrams are crucial, besides the standard word level features. The reordering features, while rather important, are less effective within this WMT task, albeit the more abstract PET features have larger impact than the often used Kendall. Good performance of F-score features leads to the conclusion that linear models might not be sufficient for modeling human sentence level ranking and to learn the right relation between precision and recall it could be worthwhile exploring non-linear models.

Acknowledgments

This work is supported by STW grant nr. 12271 and NWO VICI grant nr. 277-89-002. We also thank TAUS and the other DatAptor project User Board members.

References

Alexandra Birch and Miles Osborne. 2010. LRscore for Evaluating Lexical and Reordering Quality in MT. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 327–332, Uppsala, Sweden, July. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June. Association for Computational Linguistics.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 1999. Support Vector Learning for Ordinal Regression. In *In International Conference on Artificial Neural Networks*, pages 97–102.

Mark Hopkins and Jonathan May. 2011. Tuning as Ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic Evaluation of Translation Quality for Distant Language Pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 944–952, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hang Li. 2011. *Learning to Rank for Information Retrieval and Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Gideon Maillette de Buy Wenniger and Khalil Sima'an. 2011. Hierarchical Translation Equivalence over Word Alignments. In *ILLC Prepublication Series, PP-2011-38*. University of Amsterdam.

Michael Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the ACL 2014 Workshop on Statistical Machine Translation*.

Markos Mylonakis and Khalil Sima'an. 2008. Phrase Translation Probabilities with ITG Priors and Smoothing as Learning Objective. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 630–639, Honolulu, USA, October. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Miloš Stanojević and Khalil Sima'an. 2014. BEER: BEtter Evaluation as Ranking. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 414–419, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

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

Muyun Yang, Junguo Zhu, Sheng Li, and Tiejun Zhao. 2013. Fusion of Word and Letter Based Metrics for Automatic MT Evaluation. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI'13, pages 2204–2210. AAAI Press.

Yang Ye, Ming Zhou, and Chin-Yew Lin. 2007. Sentence Level Machine Translation Evaluation As a Ranking Problem: One Step Aside from BLEU. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 240–247, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hao Zhang and Daniel Gildea. 2007. Factorization of synchronous context-free grammars in linear time. In *In NAACL Workshop on Syntax and Structure in Statistical Translation (SSST*.

Hao Zhang, Daniel Gildea, and David Chiang. 2008. Extracting Synchronous Grammar Rules From Word-Level Alignments in Linear Time. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*, pages 1081–1088, Manchester, UK.

A Human Judgment Corpus and a Metric for Arabic MT Evaluation

Houda Bouamor, Hanan Alshikhabobakr, Behrang Mohit and Kemal Oflazer
Carnegie Mellon University in Qatar
{hbouamor,halshikh,behrang,ko}@cmu.edu

Abstract

We present a human judgments dataset and an adapted metric for evaluation of Arabic machine translation. Our medium-scale dataset is the first of its kind for Arabic with high annotation quality. We use the dataset to adapt the BLEU score for Arabic. Our score (AL-BLEU) provides partial credits for stem and morphological matchings of hypothesis and reference words. We evaluate BLEU, METEOR and AL-BLEU on our human judgments corpus and show that AL-BLEU has the highest correlation with human judgments. We are releasing the dataset and software to the research community.

1 Introduction

Evaluation of Machine Translation (MT) continues to be a challenging research problem. There is an ongoing effort in finding simple and scalable metrics with rich linguistic analysis. A wide range of metrics have been proposed and evaluated mostly for European target languages (Callison-Burch et al., 2011; Macháček and Bojar, 2013). These metrics are usually evaluated based on their correlation with human judgments on a set of MT output. While there has been growing interest in building systems for translating into Arabic, the evaluation of Arabic MT is still an under-studied problem. Standard MT metrics such as BLEU (Papineni et al., 2002) or TER (Snover et al., 2006) have been widely used for evaluating Arabic MT (El Kholy and Habash, 2012). These metrics use strict word and phrase matching between the MT output and reference translations. For morphologically rich target languages such as Arabic, such criteria are too simplistic and inadequate. In this paper, we present: (a) the first human judgment dataset for Arabic MT (b) the Arabic Language BLEU (AL-BLEU), an extension of the BLEU score for Arabic MT evaluation.

Our annotated dataset is composed of the output of six MT systems with texts from a diverse set of topics. A group of ten native Arabic speakers annotated this corpus with high-levels of inter- and intra-annotator agreements. Our AL-BLEU metric uses a rich set of morphological, syntactic and lexical features to extend the evaluation beyond the exact matching. We conduct different experiments on the newly built dataset and demonstrate that AL-BLEU shows a stronger average correlation with human judgments than the BLEU and METEOR scores. Our dataset and our AL-BLEU metric provide useful testbeds for further research on Arabic MT and its evaluation.[1]

2 Related Work

Several studies on MT evaluation have pointed out the inadequacy of the standard n-gram based evaluation metrics for various languages (Callison-Burch et al., 2006). For morphologically complex languages and those without word delimiters, several studies have attempted to improve upon them and suggest more reliable metrics that correlate better with human judgments (Denoual and Lepage, 2005; Homola et al., 2009).

A common approach to the problem of morphologically complex words is to integrate some linguistic knowledge in the metric. METEOR (Denkowski and Lavie, 2011), TER-Plus (Snover et al., 2010) incorporate limited linguistic resources. Popović and Ney (2009) showed that n-gram based evaluation metrics calculated on POS sequences correlate well with human judgments, and recently designed and evaluated MPF, a BLEU-style metric based on morphemes and POS tags (Popović, 2011). In the same direc-

[1] The dataset and the software are available at: http://nlp.qatar.cmu.edu/resources/AL-BLEU

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 207–213,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

tion, Chen and Kuhn (2011) proposed AMBER, a modified version of BLEU incorporating recall, extra penalties, and light linguistic knowledge about English morphology. Liu et al. (2010) propose TESLA-M, a variant of a metric based on n-gram matching that utilizes light-weight linguistic analysis including lemmatization, POS tagging, and WordNet synonym relations. This metric was then extended to TESLA-B to model phrase synonyms by exploiting bilingual phrase tables (Dahlmeier et al., 2011). Tantug et al. (2008) presented BLEU+, a tool that implements various extension to BLEU computation to allow for a better evaluation of the translation performance for Turkish.

To the best of our knowledge the only human judgment dataset for Arabic MT is the small corpus which was used to tune parameters of the METEOR metric for Arabic (Denkowski and Lavie, 2011). Due to the shortage of Arabic human judgment dataset, studies on the performance of evaluation metrics have been constrained and limited. A relevant effort in this area is the upper-bound estimation of BLEU and METEOR scores for Arabic MT output (El Kholy and Habash, 2011). As part of its extensive functionality, the AMEANA system provides the upper-bound estimate by an exhaustive matching of morphological and lexical features between the hypothesis and the reference translations. Our use of morphological and lexical features overlaps with the AMEANA framework. However, we extend our partial matching to a supervised tuning framework for estimating the value of partial credits. Moreover, our human judgment dataset allows us to validate our framework with a large-scale gold-standard data.

3 Human judgment dataset

We describe here our procedure for compiling a diverse Arabic MT dataset and annotating it with human judgments.

3.1 Data and systems

We annotate a corpus composed of three datasets: (1) the standard English-Arabic NIST 2005 corpus, commonly used for MT evaluations and composed of news stories. We use the first English translation as the source and the single corresponding Arabic sentence as the reference. (2) the MEDAR corpus (Maegaard et al., 2010) that consists of texts related to the climate change with four Arabic reference translations. We only use the first reference in this study. (3) a small dataset of Wikipedia articles (WIKI) to extend our corpus and metric evaluation to topics beyond the commonly-used news topics. This sub-corpus consists of our in-house Arabic translations of seven English Wikipedia articles. The articles are: *Earl Francis Lloyd*, *Western Europe*, *Citizenship*, *Marcus Garvey*, *Middle Age translation*, *Acadian*, *NBA*. The English articles which do not exist in the Arabic Wikipedia were manually translated by a bilingual linguist.

Table 1 gives an overview of these sub-corpora characteristics.

	NIST	**MEDAR**	**WIKI**
# of Documents	100	4	7
# of Sentences	1056	509	327

Table 1: Statistics on the datasets.

We use six state-of-the-art English-to-Arabic MT systems. These include four research-oriented phrase-based systems with various morphological and syntactic features and different Arabic tokenization schemes and also two commercial off-the-shelf systems.

3.2 Annotation of human judgments

In order conduct a manual evaluation of the six MT systems, we formulated it as a ranking problem. We adapt the framework used in the WMT 2011 shared task for evaluating MT metrics on European language pairs (Callison-Burch et al., 2011) for Arabic MT. We gather human ranking judgments by asking ten annotators (each native speaker of Arabic with English as a second language) to assess the quality of the English-Arabic systems, by ranking sentences relative to each other, from the best to the worst (ties are allowed).

We use the Appraise toolkit (Federmann, 2012) designed for manual MT evaluation. The tool displays to the annotator, the source sentence and translations produced by various MT systems. The annotators received initial training on the tool and the task with ten sentences. They were presented with a brief guideline indicating the purpose of the task and the main criteria of MT output evaluation.

Each annotator was assigned to 22 ranking tasks. Each task included ten screens. Each screen involveed ranking translations of ten sentences. In total, we collected $22,000$ rankings for 1892 sen-

tences (22 tasks×10 screens×10 judges). In each annotation screen, the annotator was shown the source-language (English) sentences, as well as five translations to be ranked. We did not provide annotators with the reference to avoid any bias in the annotation process. Each source sentence was presented with its direct context. Rather than attempting to get a complete ordering over the systems, we instead relied on random selection and a reasonably large sample size to make the comparisons fair (Callison-Burch et al., 2011).

An example of a source sentence and its five translations to be ranked is given in Table 2.

3.3 Annotation quality and analysis

In order to ensure the validity of any evaluation setup, a reasonable of *inter-* and *intra*-annotator agreement rates in ranking should exist. To measure these agreements, we deliberately reassigned 10% of the tasks to second annotators. Moreover, we ensured that 10% of the screens are redisplayed to the same annotator within the same task. This procedure allowed us to collect reliable quality control measure for our dataset.

	κ_{inter}	κ_{intra}
EN-AR	0.57	0.62
Average EN-EU	0.41	0.57
EN-CZ	0.40	0.54

Table 3: Inter- and intra-annotator agreement scores for our annotation compared to the average scores for five English to five European languages and also English-Czech (Callison-Burch et al., 2011).

We measured head-to-head pairwise agreement among annotators using Cohen's kappa (κ) (Cohen, 1968), defined as follows:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

where P(A) is the proportion of times annotators agree and P(E) is the proportion of agreement by chance.

Table 3 gives average values obtained for inter-annotator and intra-annotator agreement and compare our results to similar annotation efforts in WMT-13 on different European languages. Here we compare against the average agreement for English to five languages and also from English to one morphologically rich language (Czech).[4]

Based on Landis and Koch (1977) κ interpretation, the κ_{inter} value (57%) and also comparing our agreement scores with WMT-13 annotations, we believe that we have reached a reliable and consistent annotation quality.

4 AL-BLEU

Despite its well-known shortcomings (Callison-Burch et al., 2006), BLEU continues to be the de-facto MT evaluation metric. BLEU uses an exact n-gram matching criterion that is too strict for a morphologically rich language like Arabic. The system outputs in Table 2 are examples of how BLEU heavily penalizes Arabic. Based on BLEU, the best hypothesis is from Sys_5 which has three unigram and one bigram exact matches with the reference. However, the sentence is the 4$^{\text{th}}$ ranked by annotators. In contrast, the output of Sys_3 (ranked 1$^{\text{st}}$ by annotators) has only one exact match, but several partial matches when morphological and lexical information are taken into consideration.

We propose the Arabic Language BLEU (AL-BLEU) metric which extends BLEU to deal with Arabic rich morphology. We extend the matching to morphological, syntactic and lexical levels with an optimized partial credit. AL-BLEU starts with the exact matching of hypothesis tokens against the reference tokens. Furthermore, it considers the following: (a) morphological and syntactic feature matching, (b) stem matching. Based on Arabic linguistic intuition, we check the matching of a subset of 5 morphological features: (i) POS tag, (ii) gender (iii) number (iv) person (v) definiteness. We use the MADA package (Habash et al., 2009) to collect the stem and the morphological features of the hypothesis and reference translation.

Figure 1 summarizes the function in which we consider partial matching ($m(t_h, t_r)$) of a hypothesis token (t_h) and its associated reference token (t_r). Starting with the BLEU criterion, we first check if the hypothesis token is same as the reference one and provide the full credit for it. If the exact matching fails, we provide partial credit for matching at the stem and morphological level. The value of the partial credits are the sum of the stem weight (w_s) and the morphological fea-

[4]We compare against the agreement score for annotations performed by WMT researchers which are higher than the WMT annotations on Mechanical Turk.

Source	France plans to attend ASEAN emergency summit.						
Reference	*frnsaA tEtzm HDwr qmp AaAlaAsyaAn AaAlTaAr}ip*						فرنسا تعتزم حضور قمة الاسيان الطارئة.
Hypothesis	**Systems**	**Rank**$_{Annot}$	**BLEU**	**Rank**$_{BLEU}$	**AL-BLEU**	**Rank**$_{AL-BLEU}$	
	Sys$_1$	2	0.0047	2	**0.4816**	**1**	وتخطط فرنسا لحضور قمة الآسيان الطارئة *wtxTaT frnsaA lHDwr qmp AaAl—syaAn AaAlTaAr}ip*
	Sys$_2$	3	0.0037	3	0.0840	3	وتخطط فرنسا لحضور قمة الأسيان *wtxTaT frnsaA lHDwr qmp AaAlOasyaAn*
	Sys$_3$	1	0.0043	4	0.0940	2	فرنسا تخطط لحضور القمة الطارئة للأسيان *frnsaA txTaT lHDwr AaAlqmp AaAlTaAr}ip lalOasyaAn*
	Sys$_4$	5	0.0043	4	0.0604	5	خطط فرنسا لحضور قمة آسيان الطوارئ *xTaT frnsaA lHDwr qmp —syaAn AaAlTwaAri}*
	Sys$_5$	4	**0.0178**	**1**	0.0826	4	فرنسا لحضور قمة الاسيان خطط الطوارئ *frnsaA lHDwr qmp AaAlaAsyaAn xTaT AaAlTwaAri}*

Table 2: Example of ranked MT outputs in our gold-standard dataset. The first two rows specify the English input and the Arabic reference, respectively. The third row of the table lists the different MT system as ranked by annotators, using BLEU scores (column 4) and AL-BLEU (column 6). The different translation candidates are given here along with their associated Bucklwalter transliteration.[3] This example, shows clearly that AL-BLEU correlates better with human decision.

$$m(t_h, t_r) = \begin{cases} 1, & \textbf{if } t_h = t_r \\ w_s + \sum_{i=1}^{5} w_{fi} & \textbf{otherwise} \end{cases}$$

Figure 1: Formulation of our partial matching.

ture weights (w_{fi}). Each weight is included in the partial score, if such matching exist (e.g., stem match). In order to avoid over-crediting, we limit the range of weights with a set of constraints. Moreover, we use a development set to optimize the weights towards improvement of correlation with human judgments, using a hill-climbing algorithm (Russell and Norvig, 2009). Figure 2 illustrates these various samples of partial matching highlighted in different colors.

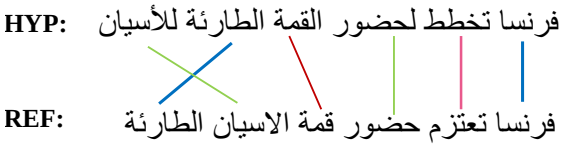


Figure 2: An MT example with exact matchings (blue), stem and morphological matching (green), stem only matching (red) and morphological-only matching (pink).

Following the BLEU-style exact matching and scoring of different n-grams, AL-BLEU updates the n-gram scores with the partial credits from non-exact matches. We use a minimum partial credit for n-grams which have tokens with different matching score. The contribution of a partially-matched n-gram is not 1 (as counted in BLEU), but the minimum value that individual tokens within the bigram are credited. For example, if a bigram is composed of a token with exact matching and a token with stem matching, this bigram receives a credit equal to a unigram with the stem matching (a value less than 1). While partial credits are added for various n-grams, the final computation of the AL-BLEU is similar to the original BLEU based on the geometric mean of the different matched n-grams. We follow BLEU in using a very small smoothing value to avoid zero n-gram counts and zero score.

5 Experiments and results

An automatic evaluation metric is said to be successful if it is shown to have high agreement with human-performed evaluations (Soricut and Brill, 2004). We use Kendall's tau τ (Kendall, 1938), a coefficient to measure the correlation between the system rankings and the human judgments at the sentence level. Kendall's tau τ is calculated as follows:

$$\tau = \frac{\text{\# of concordant pairs - \# of discordant pairs}}{\text{total pairs}}$$

where a *concordant pair* indicates two translations of the same sentence for which the ranks obtained from the manual ranking task and from the corresponding metric scores agree (they disagree in a *discordant pair*). The possible values of τ range from -1 (all pairs are discordant) to 1 (all pairs

	Dev	Test
BLEU	0.3361	0.3162
METEOR	0.3331	0.3426
AL-BLEU$_{Morph}$	**0.3746**	0.3535
AL-BLEU$_{Lex}$	0.3732	**0.3564**
AL-BLEU	**0.3759**	**0.3521**

Table 4: Comparison of the average Kendall's τ correlation.

are concordant). Thus, an automatic evaluation metric with a higher τ value is making predictions that are more similar to the human judgments than an automatic evaluation metric with a lower τ. We calculate the τ score for each sentence and average the scores to reach the corpus-level correlation. We conducted a set of experiments to compare the correlation of AL-BLEU against the state-of-the art MT evaluation metrics. For this we use a subset of 900 sentences extracted from the dataset described in Section 3.1. As mentioned above, the stem and morphological features in AL-BLEU are parameterized each by weights which are used to calculate the partial credits. We optimize the value of each weight towards correlation with human judgment by hill climbing with 100 random restarts using a development set of 600 sentences. The 300 remaining sentences (100 from each corpus) are kept for testing. The development and test sets are composed of equal portions of sentences from the three sub-corpora (NIST, MEDAR, WIKI).

As baselines, we measured the correlation of BLEU and METEOR with human judgments collected for each sentence. We did not observe a strong correlation with the Arabic-tuned METEOR. We conducted our experiments on the standard METEOR which was a stronger baseline than its Arabic version. In order to avoid the zero n-gram counts and artificially low BLEU scores, we use a smoothed version of BLEU. We follow Liu and Gildea (2005) to add a small value to both the matched n-grams and the total number of n-grams (epsilon value of 10^{-3}). In order to reach an optimal ordering of partial matches, we conducted a set of experiments in which we compared different orders between the morphological and lexical matchings to settle with the final order which was presented in Figure 1.

Table 4 shows a comparison of the average correlation with human judgments for BLEU, METEOR and AL-BLEU. AL-BLEU shows a strong improvement against BLEU and a competitive improvement against METEOR both on the test and development sets. The example in Table 2 shows a sample case of such improvement. In the example, the sentence ranked the highest by the annotator has only two exact matching with the reference translation (which results in a low BLEU score). The stem and morphological matching of AL-BLEU, gives a score and ranking much closer to human judgments.

6 Conclusion

We presented AL-BLEU, our adaptation of BLEU for the evaluation of machine translation into Arabic. The metric uses morphological, syntactic and lexical matching to go beyond exact token matching. We also presented our annotated corpus of human ranking judgments for evaluation of Arabic MT. The size and diversity of the topics in the corpus, along with its relatively high annotation quality (measured by IAA scores) makes it a useful resource for future research on Arabic MT. Moreover, the strong performance of our AL-BLEU metric is a positive indicator for future exploration of richer linguistic information in evaluation of Arabic MT.

7 Acknowledgements

We thank Michael Denkowski, Ahmed El Kholy, Francisco Guzman, Nizar Habash, Alon Lavie, Austin Matthews, Preslav Nakov for their comments and help in creation of our dataset. We also thank our team of annotators from CMU-Qatar. This publication was made possible by grants YSREP-1-018-1-004 and NPRP-09-1140-1-177 from the Qatar National Research Fund (a member of the Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the Role of BLEU in Machine Translation Research. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar Zaidan. 2011. Findings of the 2011 Workshop on Statistical Machine Translation. In

Proceedings of the Sixth Workshop on Statistical Machine Translation, Edinburgh, Scotland.

Boxing Chen and Roland Kuhn. 2011. AMBER: A Modified BLEU, Enhanced Ranking Metric. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 71–77, Edinburgh, Scotland.

Jacob Cohen. 1968. Weighted Kappa: Nominal Scale Agreement Provision for Scaled Disagreement or Partial Credit. *Psychological bulletin*, 70(4):213.

Daniel Dahlmeier, Chang Liu, and Hwee Tou Ng. 2011. TESLA at WMT 2011: Translation Evaluation and Tunable Metric. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 78–84, Edinburgh, Scotland, July. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the EMNLP 2011 Workshop on Statistical Machine Translation*, Edinburgh, UK.

Etienne Denoual and Yves Lepage. 2005. BLEU in Characters: Towards Automatic MT Evaluation in Languages Without Word Delimiters. In *Proceedings of the Second International Joint Conference on Natural Language Processing*, Jeju Island, Republic of Korea.

Ahmed El Kholy and Nizar Habash. 2011. Automatic Error Analysis for Morphologically Rich Languages. In *Proceedings of the MT Summit XIII*, pages 225–232, Xiamen, China.

Ahmed El Kholy and Nizar Habash. 2012. Orthographic and Morphological Processing for English-Arabic Statistical Machine Translation. *Machine Translation*, 26(1):25–45.

Christian Federmann. 2012. Appraise: an Open-Source Toolkit for Manual Evaluation of MT Output. *The Prague Bulletin of Mathematical Linguistics*, 98(1):25–35.

N. Habash, O. Rambow, and R. Roth. 2009. Mada+ Tokan: A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, POS Tagging, Stemming and Lemmatization. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools (MEDAR)*, Cairo, Egypt.

Petr Homola, Vladislav Kuboň, and Pavel Pecina. 2009. A Simple Automatic MT Evaluation Metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 33–36, Athens, Greece, March. Association for Computational Linguistics.

Maurice G Kendall. 1938. A New Measure of Rank Correlation. *Biometrika*.

J Richard Landis and Gary G Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–174.

Ding Liu and Daniel Gildea. 2005. Syntactic Features for Evaluation of Machine Translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 25–32.

Chang Liu, Daniel Dahlmeier, and Hwee Tou Ng. 2010. TESLA: Translation Evaluation of Sentences with Linear-Programming-Based Analysis. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics (MATR)*, pages 354–359.

Matouš Macháček and Ondřej Bojar. 2013. Results of the WMT13 Metrics Shared Task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 45–51, Sofia, Bulgaria.

Bente Maegaard, Mohamed Attia, Khalid Choukri, Olivier Hamon, Steven Krauwer, and Mustafa Yaseen. 2010. Cooperation for Arabic Language Resources and Tools–The MEDAR Project. In *Proceedings of LREC*, Valetta, Malta.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania.

Maja Popović and Hermann Ney. 2009. Syntax-oriented Evaluation Measures for Machine Translation Output. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 29–32, Athens, Greece.

Maja Popović. 2011. Morphemes and POS Tags for n-gram Based Evaluation Metrics. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 104–107, Edinburgh, Scotland.

Stuart Russell and Peter Norvig. 2009. *Artificial Intelligence: A Modern Approach*. Prentice Hall Englewood Cliffs.

Matthew Snover, Bonnie J. Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of AMTA*, Boston, USA.

Matthew Snover, Nitin Madnani, Bonnie J. Dorr, and Richard Schwartz. 2010. TER-Plus: Paraphrase, Semantic, and Alignment Enhancements to Translation Edit Rate. *Machine Translation*, 23(2-3).

Radu Soricut and Eric Brill. 2004. A Unified Framework For Automatic Evaluation Using 4-Gram Co-occurrence Statistics. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 613–620, Barcelona, Spain, July.

Cüneyd Tantug, Kemal Oflazer, and Ilknur Durgar El-Kahlout. 2008. BLEU+: a Tool for Fine-Grained BLEU Computation. In *Proceedings of the 6th edition of the Language Resources and Evaluation Conference*, Marrakech, Morocco.

Learning to Differentiate Better from Worse Translations

Francisco Guzmán Shafiq Joty Lluís Màrquez
Alessandro Moschitti Preslav Nakov Massimo Nicosia
ALT Research Group
Qatar Computing Research Institute — Qatar Foundation
{fguzman,sjoty,lmarquez,amoschitti,pnakov,mnicosia}@qf.org.qa

Abstract

We present a pairwise learning-to-rank approach to machine translation evaluation that learns to differentiate *better* from *worse* translations in the context of a given reference. We integrate several layers of linguistic information encapsulated in tree-based structures, making use of both the reference and the system output simultaneously, thus bringing our ranking closer to how humans evaluate translations. Most importantly, instead of deciding upfront which types of features are important, we use the learning framework of preference re-ranking kernels to learn the features automatically. The evaluation results show that learning in the proposed framework yields better correlation with humans than computing the direct similarity over the same type of structures. Also, we show our structural kernel learning (SKL) can be a general framework for MT evaluation, in which syntactic and semantic information can be naturally incorporated.

1 Introduction

We have seen in recent years fast improvement in the overall quality of machine translation (MT) systems. This was only possible because of the use of automatic metrics for MT evaluation, such as BLEU (Papineni et al., 2002), which is the de-facto standard; and more recently: TER (Snover et al., 2006) and METEOR (Lavie and Denkowski, 2009), among other emerging MT evaluation metrics. These automatic metrics provide fast and inexpensive means to compare the output of different MT systems, without the need to ask for human judgments each time the MT system has been changed.

As a result, this has enabled rapid development in the field of statistical machine translation (SMT), by allowing to train and tune systems as well as to track progress in a way that highly correlates with human judgments.

Today, MT evaluation is an active field of research, and modern metrics perform analysis at various levels, e.g., lexical (Papineni et al., 2002; Snover et al., 2006), including synonymy and paraphrasing (Lavie and Denkowski, 2009); syntactic (Giménez and Màrquez, 2007; Popović and Ney, 2007; Liu and Gildea, 2005); semantic (Giménez and Màrquez, 2007; Lo et al., 2012); and discourse (Comelles et al., 2010; Wong and Kit, 2012; Guzmán et al., 2014; Joty et al., 2014).

Automatic MT evaluation metrics compare the output of a system to one or more human references in order to produce a *similarity* score. The quality of such a metric is typically judged in terms of correlation of the scores it produces with scores given by human judges. As a result, some evaluation metrics have been trained to reproduce the scores assigned by humans as closely as possible (Albrecht and Hwa, 2008). Unfortunately, humans have a hard time assigning an absolute score to a translation. Hence, direct human evaluation scores such as *adequacy* and *fluency*, which were widely used in the past, are now discontinued in favor of ranking-based evaluations, where judges are asked to rank the output of 2 to 5 systems instead. It has been shown that using such ranking-based assessments yields much higher inter-annotator agreement (Callison-Burch et al., 2007).

While evaluation metrics still produce numerical scores, in part because MT evaluation shared tasks at NIST and WMT ask for it, there has also been work on a ranking formulation of the MT evaluation task for a given set of outputs. This was shown to yield higher correlation with human judgments (Duh, 2008; Song and Cohn, 2011).

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 214–220,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Learning automatic metrics in a pairwise setting, i.e., learning to distinguish between two alternative translations and to decide which of the two is better (which is arguably one of the easiest ways to produce a ranking), emulates closely how human judges perform evaluation assessments in reality. Instead of learning a similarity function between a translation and the reference, they learn how to differentiate a better from a worse translation given a corresponding reference. While the pairwise setting does not provide an absolute quality scoring metric, it is useful for most evaluation and MT development scenarios.

In this paper, we propose a pairwise learning setting similar to that of Duh (2008), but we extend it to a new level, both in terms of feature representation and learning framework. First, we integrate several layers of linguistic information encapsulated in tree-based structures; Duh (2008) only used lexical and POS matches as features. Second, we use information about both the reference and two alternative translations *simultaneously*, thus bringing our ranking closer to how humans rank translations. Finally, instead of deciding upfront which types of features between hypotheses and references are important, we use a our structural kernel learning (SKL) framework to generate and select them automatically.

The structural kernel learning (SKL) framework we propose consists in: (*i*) designing a structural representation, e.g., using syntactic and discourse trees of translation hypotheses and a references; and (*ii*) applying structural kernels (Moschitti, 2006; Moschitti, 2008), to such representations in order to automatically inject structural features in the preference re-ranking algorithm. We use this method with translation-reference pairs to directly learn the features themselves, instead of learning the importance of a predetermined set of features. A similar learning framework has been proven to be effective for question answering (Moschitti et al., 2007), and textual entailment recognition (Zanzotto and Moschitti, 2006).

Our goals are twofold: (*i*) in the short term, to demonstrate that structural kernel learning is suitable for this task, and can effectively learn to rank hypotheses at the segment-level; and (*ii*) in the long term, to show that this approach provides a unified framework that allows to integrate several layers of linguistic analysis and information and to improve over the state-of-the-art.

Below we report the results of some initial experiments using syntactic and discourse structures. We show that learning in the proposed framework yields better correlation with humans than applying the traditional translation–reference similarity metrics using the same type of structures. We also show that the contributions of syntax and discourse information are cumulative. Finally, despite the limited information we use, we achieve correlation at the segment level that outperforms BLEU and other metrics at WMT12, e.g., our metric would have been ranked higher in terms of correlation with human judgments compared to TER, NIST, and BLEU in the WMT12 Metrics shared task (Callison-Burch et al., 2012).

2 Kernel-based Learning from Linguistic Structures

In our pairwise setting, each sentence s in the source language is represented by a tuple $\langle t_1, t_2, r\rangle$, where t_1 and t_2 are two alternative translations and r is a reference translation. Our goal is to develop a classifier of such tuples that decides whether t_1 is a better translation than t_2 given the reference r.

Engineering features for deciding whether t_1 is a better translation than t_2 is a difficult task. Thus, we rely on the automatic feature extraction enabled by the SKL framework, and our task is reduced to choosing: (*i*) a meaningful structural representation for $\langle t_1, t_2, r\rangle$, and (*ii*) a feature function ϕ_{mt} that maps such structures to substructures, i.e., our feature space. Since the design of ϕ_{mt} is complex, we use tree kernels applied to two simpler structural mappings $\phi_M(t_1, r)$ and $\phi_M(t_2, r)$. The latter generate the tree representations for the translation-reference pairs (t_1, r) and (t_2, r). The next section shows such mappings.

2.1 Representations

To represent a translation-reference pair (t, r), we adopt shallow syntactic trees combined with RST-style discourse trees. Shallow trees have been successfully used for question answering (Severyn and Moschitti, 2012) and semantic textual similarity (Severyn et al., 2013b); while discourse information has proved useful in MT evaluation (Guzmán et al., 2014; Joty et al., 2014). Combined shallow syntax and discourse trees worked well for concept segmentation and labeling (Saleh et al., 2014a).

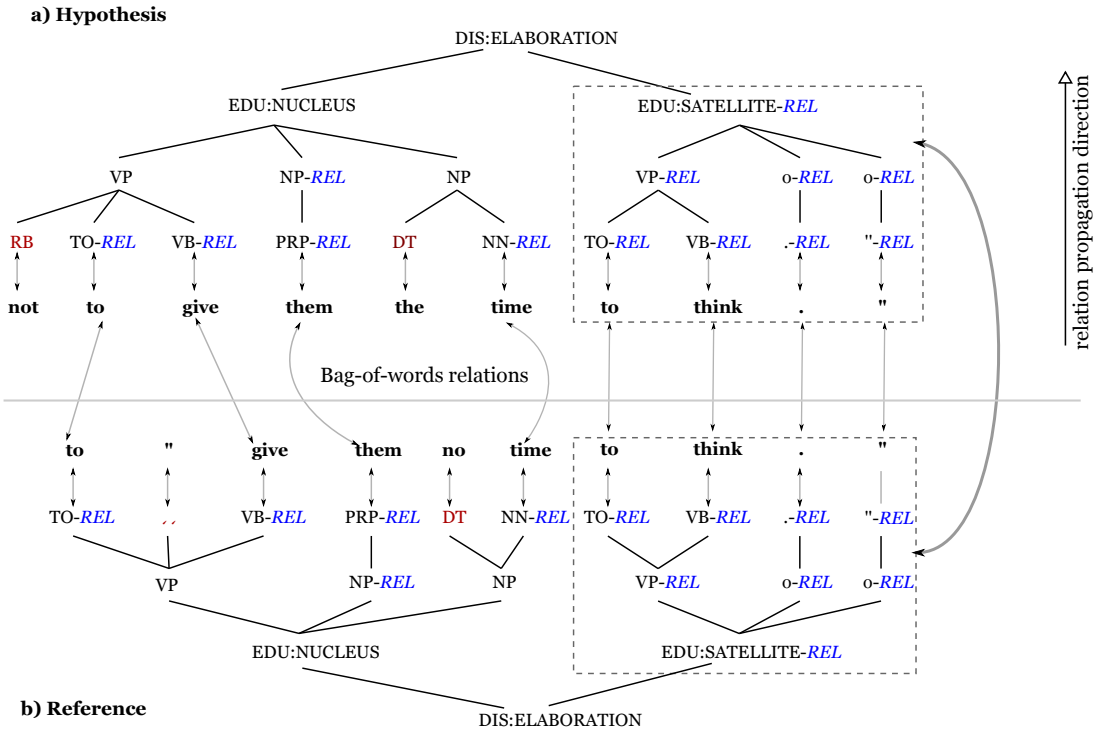


Figure 1: Hypothesis and reference trees combining discourse, shallow syntax and POS.

Figure 1 shows two example trees combining discourse, shallow syntax and POS: one for a translation hypothesis (top) and the other one for the reference (bottom). To build such structures, we used the Stanford POS tagger (Toutanova et al., 2003), the Illinois chunker (Punyakanok and Roth, 2001), and the discourse parser[1] of (Joty et al., 2012; Joty et al., 2013).

The lexical items constitute the leaves of the tree. The words are connected to their respective POS tags, which are in turn grouped into chunks. Then, the chunks are grouped into elementary discourse units (EDU), to which the nuclearity status is attached (i.e., NUCLEUS or SATELLITE). Finally, EDUs and higher-order discourse units are connected by discourse relations (e.g., DIS:ELABORATION).

2.2 Kernels-based modeling

In the SKL framework, the *learning objects* are pairs of translations $\langle t_1, t_2 \rangle$. Our objective is to automatically learn which pair features are important, independently of the source sentence. We achieve this by using kernel machines (KMs) over two learning objects $\langle t_1, t_2 \rangle, \langle t_1^{'}, t_2^{'} \rangle$, along with an explicit and structural representation of the pairs (see Fig. 1).

[1]The discourse parser can be downloaded from `http://alt.qcri.org/tools/`

More specifically, KMs carry out learning using the scalar product

$$K_{mt}(\langle t_1, t_2 \rangle, \langle t_1', t_2' \rangle) = \phi_{mt}(t_1, t_2) \cdot \phi_{mt}(t_1', t_2'),$$

where ϕ_{mt} maps pairs into the feature space.

Considering that our task is to decide whether t_1 is better than t_2, we can conveniently represent the vector for the pair in terms of the difference between the two translation vectors, i.e., $\phi_{mt}(t_1, t_2) = \phi_K(t_1) - \phi_K(t_2)$. We can approximate K_{mt} with a preference kernel PK to compute this difference in the kernel space K:

$$\begin{aligned} & PK(\langle t_1, t_2 \rangle, \langle t_1', t_2' \rangle) \qquad (1) \\ & = K(t_1) - \phi_K(t_2)) \cdot (\phi_K(t_1') - \phi_K(t_2')) \\ & = K(t_1, t_1') + K(t_2, t_2') - K(t_1, t_2') - K(t_2, t_1') \end{aligned}$$

The advantage of this is that now $K(t_i, t_j') = \phi_K(t_i) \cdot \phi_K(t_j')$ is defined between two translations only, and not between two pairs of translations. This simplification enables us to map translations into simple trees, e.g., those in Figure 1, and then to apply them tree kernels, e.g., the Partial Tree Kernel (Moschitti, 2006), which carry out a scalar product in the subtree space.

We can further enrich the representation ϕ_K, if we consider all the information available to the human judges when they are ranking translations. That is, the two alternative translations along with their corresponding reference.

In particular, let r and r' be the references for the pairs $\langle t_1, t_2 \rangle$ and $\langle t'_1, t'_2 \rangle$, we can redefine all the members of Eq. 1, e.g., $K(t_1, t'_1)$ becomes

$K(\langle t_1, r \rangle, \langle t'_1, r' \rangle) = \text{PTK}(\phi_M(t_1, r), \phi_M(t'_1, r'))$
$+ \quad \text{PTK}(\phi_M(r, t_1), \phi_M(r', t'_1))$,

where ϕ_M maps a pair of texts to a single tree.

There are several options to produce the bitext-to-tree mapping for ϕ_M. A simple approach is to only use the tree corresponding to the first argument of ϕ_M. This leads to the basic model $K(\langle t_1, r \rangle, \langle t'_1, r' \rangle) = \text{PTK}(\phi_M(t_1), \phi_M(t'_1)) + \text{PTK}(\phi_M(r), \phi_M(r'))$, i.e., the sum of two tree kernels applied to the trees constructed by ϕ_M (we previously informally mentioned it).

However, this simple mapping may be ineffective since the trees within a pair, e.g., (t_1, r), are treated independently, and no meaningful features connecting t_1 and r can be derived from their tree fragments. Therefore, we model $\phi_M(r, t_1)$ by using word-matching *relations* between t_1 and r, such that connections between words and constituents of the two trees are established using position-independent word matching. For example, in Figure 1, the thin dashed arrows show the links connecting the matching words between t_1 and r. The propagation of these relations works from the bottom up. Thus, if all children in a constituent have a link, their parent is also linked.

The use of such connections is essential as it enables the comparison of the structural properties and relations between two translation-reference pairs. For example, the tree fragment [ELABORATION [SATELLITE]] from the translation is connected to [ELABORATION [SATELLITE]] in the reference, indicating a link between two entire discourse units (drawn with a thicker arrow), and providing some reliability to the translation[2].

Note that the use of connections yields a graph representation instead of a tree. This is problematic as effective models for graph kernels, which would be a natural fit to this problem, are not currently available for exploiting linguistic information. Thus, we simply use K, as defined above, where the mapping $\phi_M(t_1, r)$ only produces a tree for t_1 annotated with the marker REL representing the connections to r. This marker is placed on all node labels of the tree generated from t_1 that match labels from the tree generated from r.

In other words, we only consider the trees enriched by markers separately, and ignore the edges connecting both trees.

3 Experiments and Discussion

We experimented with datasets of segment-level human rankings of system outputs from the WMT11 and the WMT12 Metrics shared tasks (Callison-Burch et al., 2011; Callison-Burch et al., 2012): we used the WMT11 dataset for training and the WMT12 dataset for testing. We focused on translating into English only, for which the datasets can be split by source language: Czech (cs), German (de), Spanish (es), and French (fr). There were about 10,000 non-tied human judgments per language pair per dataset. We scored our pairwise system predictions with respect to the WMT12 human judgments using the Kendall's Tau (τ), which was official at WMT12.

Table 1 presents the τ scores for all metric variants introduced in this paper: for the individual language pairs and overall. The left-hand side of the table shows the results when using as similarity the direct kernel calculation between the corresponding structures of the candidate translation and the reference[3], e.g., as in (Guzmán et al., 2014; Joty et al., 2014). The right-hand side contains the results for structured kernel learning.

We can make the following observations:
(*i*) The overall results for all SKL-trained metrics are higher than the ones when applying direct similarity, showing that learning tree structures is better than just calculating similarity.
(*ii*) Regarding the linguistic representation, we see that, when learning tree structures, syntactic and discourse-based trees yield similar improvements with a slight advantage for the former. More interestingly, when both structures are put together in a combined tree, the improvement is cumulative and yields the best results by a sizable margin. This provides positive evidence towards our goal of a unified tree-based representation with multiple layers of linguistic information.
(*iii*) Comparing to the best evaluation metrics that participated in the WMT12 Metrics shared task, we find that our approach is competitive and would have been ranked among the top 3 participants.

[2]Note that a non-pairwise model, i.e., $K(t_1, r)$, could also be used to match the structural information above, but it would not learn to compare it to a second pair (t_2, r).

[3]Applying tree kernels between the members of a pair to generate one feature (for each different kernel function) has become a standard practice in text similarity tasks (Severyn et al., 2013b) and in question answering (Severyn et al., 2013a).

		Similarity					Structured Kernel Learning				
	Structure	*cs*-en	*de*-en	*es*-en	*fr*-en	all	*cs*-en	*de*-en	*es*-en	*fr*-en	all
1	SYN	0.169	0.188	0.203	0.222	0.195	0.190	0.244	0.198	0.158	0.198
2	DIS	0.130	0.174	0.188	0.169	0.165	0.176	0.235	0.166	0.160	0.184
3	DIS+POS	0.135	0.186	0.190	0.178	0.172	0.167	0.232	0.202	0.133	0.183
4	DIS+SYN	0.156	0.205	0.206	0.203	0.192	**0.210**	**0.251**	**0.240**	**0.223**	**0.231**

Table 1: Kendall's (τ) correlation with human judgements on WMT12 for each language pair.

Furthermore, our result (0.237) is ahead of the correlation obtained by popular metrics such as TER (0.217), NIST (0.214) and BLEU (0.185) at WMT12. This is very encouraging and shows the potential of our new proposal.

In this paper, we have presented only the first exploratory results. Our approach can be easily extended with richer linguistic structures and further combined with some of the already existing strong evaluation metrics.

		Testing				
	Train	*cs*-en	*de*-en	*es*-en	*fr*-en	all
1	*cs*-en	0.210	0.204	0.217	0.204	0.209
2	*de*-en	0.196	0.251	0.203	0.202	0.213
3	*es*-en	0.218	0.204	**0.240**	0.223	0.221
4	*fr*-en	0.203	0.218	0.224	0.223	0.217
5	all	**0.231**	**0.258**	0.226	**0.232**	**0.237**

Table 2: Kendall's (τ) on WMT12 for cross-language training with DIS+SYN.

Note that the results in Table 1 were for training on WMT11 and testing on WMT12 for each language pair in isolation. Next, we study the impact of the choice of training language pair. Table 2 shows cross-language evaluation results for DIS+SYN: lines 1-4 show results when training on WMT11 for one language pair, and then testing for each language pair of WMT12.

We can see that the overall differences in performance (see the last column: *all*) when training on different source languages are rather small, ranging from 0.209 to 0.221, which suggests that our approach is quite independent of the source language used for training. Still, looking at individual test languages, we can see that for de-en and es-en, it is best to train on the same language; this also holds for fr-en, but there it is equally good to train on es-en. Interestingly, training on es-en improves a bit for cs-en.

These somewhat mixed results have motivated us to try tuning on the full WMT11 dataset; as line 5 shows, this yielded improvements for all language pairs except for es-en. Comparing to line 4 in Table 1, we see that the overall Tau improved from 0.231 to 0.237.

4 Conclusions and Future Work

We have presented a pairwise learning-to-rank approach to MT evaluation, which learns to differentiate good from bad translations in the context of a given reference. We have integrated several layers of linguistic information (lexical, syntactic and discourse) in tree-based structures, and we have used the structured kernel learning to identify relevant features and learn pairwise rankers.

The evaluation results have shown that learning in the proposed SKL framework is possible, yielding better correlation (Kendall's τ) with human judgments than computing the direct kernel similarity between translation and reference, over the same type of structures. We have also shown that the contributions of syntax and discourse information are cumulative, indicating that this learning framework can be appropriate for the combination of different sources of information. Finally, despite the limited information we used, we achieved better correlation at the segment level than BLEU and other metrics in the WMT12 Metrics task.

In the future, we plan to work towards our long-term goal, i.e., including more linguistic information in the SKL framework and showing that this can help. This would also include more semantic information, e.g., in the form of Brown clusters or using semantic similarity between the words composing the structure calculated with latent semantic analysis (Saleh et al., 2014b).

We further want to show that the proposed framework is flexible and can include information in the form of quality scores predicted by other evaluation metrics, for which a vector of features would be combined with the structured kernel.

Acknowledgments

This research is part of the Interactive sYstems for Answer Search (Iyas) project, conducted by the Arabic Language Technologies (ALT) group at Qatar Computing Research Institute (QCRI) within the Qatar Foundation.

References

Joshua Albrecht and Rebecca Hwa. 2008. Regression for machine translation evaluation at the sentence level. *Machine Translation*, 22(1-2):1–27.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (Meta-) evaluation of machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, WMT '07, pages 136–158, Prague, Czech Republic.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar Zaidan. 2011. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 22–64, Edinburgh, Scotland, UK.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, WMT '12, pages 10–51, Montréal, Canada.

Elisabet Comelles, Jesús Giménez, Lluís Màrquez, Irene Castellón, and Victoria Arranz. 2010. Document-level automatic MT evaluation based on discourse representations. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, pages 333–338, Uppsala, Sweden.

Kevin Duh. 2008. Ranking vs. regression in machine translation evaluation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, WMT '08, pages 191–194, Columbus, Ohio, USA.

Jesús Giménez and Lluís Màrquez. 2007. Linguistic features for automatic evaluation of heterogenous MT systems. In *Proceedings of the Second Workshop on Statistical Machine Translation*, WMT '07, pages 256–264, Prague, Czech Republic.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2014. Using discourse structure improves machine translation evaluation. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics*, ACL '14, pages 687–698, Baltimore, Maryland, USA.

Shafiq Joty, Giuseppe Carenini, and Raymond Ng. 2012. A Novel Discriminative Framework for Sentence-Level Discourse Analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 904–915, Jeju Island, Korea.

Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Yashar Mehdad. 2013. Combining Intra- and Multi-sentential Rhetorical Parsing for Document-level Discourse Analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, pages 486–496, Sofia, Bulgaria.

Shafiq Joty, Francisco Guzmán, Lluís Màrquez, and Preslav Nakov. 2014. DiscoTK: Using discourse structure for machine translation evaluation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, WMT '14, pages 402–408, Baltimore, Maryland, USA.

Alon Lavie and Michael Denkowski. 2009. The METEOR metric for automatic evaluation of machine translation. *Machine Translation*, 23(2–3):105–115.

Ding Liu and Daniel Gildea. 2005. Syntactic features for evaluation of machine translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 25–32, Ann Arbor, Michigan, USA.

Chi-kiu Lo, Anand Karthik Tumuluru, and Dekai Wu. 2012. Fully automatic semantic MT evaluation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, WMT '12, pages 243–252, Montréal, Canada.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question answer classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, ACL '07, pages 776–783, Prague, Czech Republic.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of 17th European Conference on Machine Learning and the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, ECML/PKDD '06, pages 318–329, Berlin, Germany.

Alessandro Moschitti. 2008. Kernel methods, syntax and semantics for relational text categorization. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 253–262, Napa Valley, California, USA.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meting of the Association for Computational Linguistics*, ACL '02, pages 311–318, Philadelphia, Pennsylvania, USA.

Maja Popović and Hermann Ney. 2007. Word error rates: Decomposition over POS classes and applications for error analysis. In *Proceedings of the Second Workshop on Statistical Machine Translation*, WMT '07, pages 48–55, Prague, Czech Republic.

Vasin Punyakanok and Dan Roth. 2001. The use of classifiers in sequential inference. In *Advances in Neural Information Processing Systems 14*, NIPS '01, pages 995–1001, Vancouver, Canada.

Iman Saleh, Scott Cyphers, Jim Glass, Shafiq Joty, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2014a. A study of using syntactic and semantic structures for concept segmentation and labeling. In *Proceedings of the 25th International Conference on Computational Linguistics*, COLING '14, pages 193–202, Dublin, Ireland.

Iman Saleh, Alessandro Moschitti, Preslav Nakov, Lluís Màrquez, and Shafiq Joty. 2014b. Semantic kernels for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, Doha, Qatar.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer re-ranking. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 741–750, Portland, Oregon, USA.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013a. Learning adaptable patterns for passage reranking. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, CoNLL '13, pages 75–83, Sofia, Bulgaria.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013b. Learning semantic textual similarity with structural representations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, ACL '13, pages 714–718, Sofia, Bulgaria.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas*, AMTA '06, Cambridge, Massachusetts, USA.

Xingyi Song and Trevor Cohn. 2011. Regression and ranking based optimisation for sentence-level MT evaluation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 123–129, Edinburgh, Scotland, UK.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, HLT-NAACL '03, pages 173–180, Edmonton, Canada.

Billy Wong and Chunyu Kit. 2012. Extending machine translation evaluation metrics with lexical cohesion to document level. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1060–1068, Jeju Island, Korea.

Fabio Massimo Zanzotto and Alessandro Moschitti. 2006. Automatic learning of textual entailments with cross-pair similarities. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, COLING-ACL '06, pages 401–408, Sydney, Australia.

Two Improvements to Left-to-Right Decoding for Hierarchical Phrase-based Machine Translation

Maryam Siahbani and **Anoop Sarkar**
School of Computing Science
Simon Fraser University
Burnaby BC. Canada
msiahban,anoop@cs.sfu.ca

Abstract

Left-to-right (LR) decoding (Watanabe et al., 2006) is promising decoding algorithm for hierarchical phrase-based translation (Hiero) that visits input spans in arbitrary order producing the output translation in left to right order. This leads to far fewer language model calls, but while LR decoding is more efficient than CKY decoding, it is unable to capture some hierarchical phrase alignments reachable using CKY decoding and suffers from lower translation quality as a result. This paper introduces two improvements to LR decoding that make it comparable in translation quality to CKY-based Hiero.

1 Introduction

Hierarchical phrase-based translation (Hiero) (Chiang, 2007) uses a lexicalized synchronous context-free grammar (SCFG) extracted from word and phrase alignments of a bitext. Decoding for Hiero is typically done with CKY-style decoding with time complexity $O(n^3)$ for source input with n words. Computing the language model score for each hypothesis within CKY decoding requires two histories, the left and the right edge of each span, due to the fact that the target side is built inside-out from sub-spans (Heafield et al., 2011; Heafield et al., 2013).

LR-decoding algorithms exist for phrase-based (Koehn, 2004; Galley and Manning, 2010) and syntax-based (Huang and Mi, 2010; Feng et al., 2012) models and also for hierarchical phrase-based models (Watanabe et al., 2006; Siahbani et al., 2013), which is our focus in this paper.

Watanabe et al. (2006) first proposed left-to-right (LR) decoding for Hiero (LR-Hiero henceforth) which uses beam search and runs in $O(n^2b)$ in practice where n is the length of source sentence and b is the size of beam (Huang and Mi, 2010). To simplify target generation, SCFG rules are constrained to be prefix-lexicalized on target side aka Griebach Normal Form (GNF). Throughout this paper we abuse the notation for simplicity and use the term GNF grammars for such SCFGs. This constraint drastically reduces the size of grammar for LR-Hiero in comparison to Hiero grammar (Siahbani et al., 2013). However, the original LR-Hiero decoding algorithm does not perform well in comparison to current state-of-the-art Hiero and phrase-based translation systems. Siahbani et al. (2013) propose an augmented version of LR decoding to address some limitations in the original LR-Hiero algorithm in terms of translation quality and time efficiency.

Although, LR-Hiero performs much faster than Hiero in decoding and obtains BLEU scores comparable to phrase-based translation system on some language pairs, there is still a notable gap between CKY-Hiero and LR-Hiero (Siahbani et al., 2013). We show in this paper using instructive examples that CKY-Hiero can capture some complex phrasal re-orderings that are observed in language pairs such as Chinese-English that LR-Hiero cannot (c.f. Sec.3).

We introduce two improvements to LR decoding of GNF grammars: (1) We add *queue diversity* to the cube pruning algorithm for LR-Hiero, and (2) We extend the LR-Hiero decoder to capture all the hierarchical phrasal alignments that are reachable in CKY-Hiero (restricted to using GNF grammars). We evaluate our modifications on three language pairs and show that LR-Hiero can reach the translation scores comparable to CKY-Hiero in two language pairs, and reduce the gap between Hiero and LR-Hiero on the third one.

2 LR Decoding with Queue Diversity

LR-Hiero uses a constrained lexicalized SCFG which we call a GNF grammar: $X \rightarrow \langle \gamma, \bar{b}\, \beta \rangle$ where γ is a string of non-terminal and terminal symbols, $\bar{b}$ is a string of terminal symbols and β is a possibly empty sequence of non-terminals. This ensures that as each rule is used in a derivation,

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 221–226,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Algorithm 1: LR-Hiero Decoding

1: Input sentence: $\mathbf{f} = f_0 f_1 \dots f_n$
2: $\mathcal{F}$ = FutureCost($\mathbf{f}$) (Precompute future cost[1] for spans)
3: $S_0 = \{\}$ (Create empty initial stack)
4: $h_0 = (\langle s \rangle, [[0, n]], \emptyset, \mathcal{F}_{[0,n]})$ (Initial hypothesis 4-tuple)
5: Add h_0 to S_0 (Push initial hyp into first Stack)
6: **for** $i = 1, \dots, n$ **do**
7: $cubeList = \{\}$ (MRL is max rule length)
8: **for** $p = max(i - \text{MRL}, 0), \dots, i - 1$ **do**
9: $\{G\}$ = Grouped(S_p) (based on the first uncovered span)
10: **for** $g \in \{G\}$ **do**
11: $[u, v] = g_{span}$
12: R = GetSpanRules($[u, v]$)
13: **for** $R_s \in R$ **do**
14: $cube = [g_{hyps}, R_s]$
15: Add $cube$ to $cubeList$
16: S_i = Merge($cubeList, \mathcal{F}$) (Create stack S_i and add new hypotheses to it, see Figure 1)
17: **return** $\arg\max(S_n)$

18: **Merge**($CubeList, \mathcal{F}$)
19: $heapQ = \{\}$
20: **for** each (H, R) in $cubeList$ **do**
21: $hypList$ = getBestHypotheses($(H, R), \mathcal{F}, d$) (d best hypotheses of each cube)
22: **for** each h' in $hypList$ **do**
23: push($heapQ, (h'_c, h', [H, R])$ (Push new hyp in queue)
24: $hypList = \{\}$
25: **while** $|heapQ| > 0$ and $|hypList| < K$ **do**
26: $(h'_c, h', [H, R])$ = pop($heapQ$) (pop the best hypothesis)
27: push($heapQ, GetNeighbours([H, R])$ (Push neighbours to queue)
28: Add h' to $hypList$
29: **return** $hypList$

the target string is generated from left to right. The rules are obtained from a word and phrase aligned bitext using the rule extraction algorithm in (Watanabe et al., 2006).

LR-Hiero decoding uses a top-down depth-first search, which strictly grows the hypotheses in target surface ordering. Search on the source side follows an Earley-style search (Earley, 1970), the dot jumps around on the source side of the rules based on the order of nonterminals on the target side. This search is integrated with beam search or cube pruning to find the k-best translations.

Algorithm 1 shows the pseudocode for LR-Hiero decoding with cube pruning (Chiang, 2007) (CP). LR-Hiero with CP was introduced in (Siahbani et al., 2013). In this pseudocode, we have introduced the notion of *queue diversity* (explained below). However to understand our change we need to understand the algorithm in more detail.

[1]The future cost is precomputed in a way similar to the phrase-based models (Koehn et al., 2007) using only the terminal rules of the grammar.

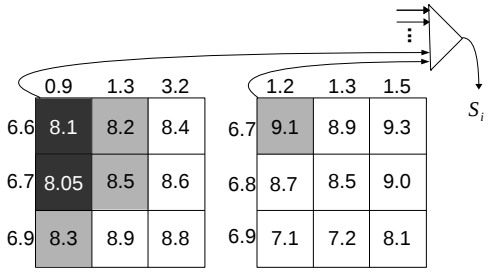


Figure 1: Cubes (grids) are fed to a priority queue (triangle) and generated hypotheses are iteratively popped from the queue and added to stack S_i. Lower scores are better. Scores of rules and hypotheses appear on the top and left side of the grids respectively. Shaded entries are hypotheses in the queue and black ones are popped from the queue and added to S_i.

Each source side non-terminal is instantiated with the legal spans given the input source string, e.g. if there is a Hiero rule $\langle aX_1, a'X_1 \rangle$ and if a only occurs at position 3 in the input then this rule can be applied to span $[3, i]$ for all i, $4 < i \leq n$ for input of length n and source side X_1 is instantiated to span $[4, i]$. A worked out example of how the decoder works is shown in Figure 2. Each partial hypothesis h is a 4-tuple (h_t, h_s, h_{cov}, h_c): consisting of a translation prefix h_t, a (LIFO-ordered) *list* h_s of uncovered spans, source words coverage set h_{cov} and the hypothesis cost h_c. The initial hypothesis is a null string with just a sentence-initial marker $\langle s \rangle$ and the list h_s containing a span of the whole sentence, $[0, n]$. The hypotheses are stored in stacks $S_0, \dots, S_n$, where S_p contains hypotheses covering p source words just like in stack decoding for phrase-based SMT (Koehn et al., 2003).

To fill stack S_i we consider hypotheses in each stack S_p[2], which are first partitioned into a set of groups $\{G\}$, based on their first uncovered span (line 9). Each group g is a 2-tuple (g_{span}, g_{hyps}), where g_{hyps} is a list of hypotheses which share the same first uncovered span g_{span}. Rules matching the span g_{span} are obtained from routine *GetSpanRules*. Each g_{hyps} and possible R_s create a cube which is added to $cubeList$.

The *Merge* routine gets the best hypotheses from all cubes (see Fig.1). Hypotheses (rows) and columns (rules) are sorted based on their scores. $GetBestHypotheses((H, R), \mathcal{F}, d)$ uses current hypothesis H and rule R to produce new hypotheses. The first best hypothesis, h' along with its score h'_c and corresponding cube (H, R) is placed in a priority queue $heapQ$ (triangle in Figure 1 and line 23 in Algorithm 1). Iteratively the K best

[2]As the length of rules are limited (at most MRL), we can ignore stacks with index less than $i - \text{MRL}$

rules	hypotheses
	$\langle s \rangle [0, 15]$
G 1)$\langle Taiguo\ shi\ X_1 / Thailand\ X_1 \rangle$	$\langle s \rangle$ Thailand [2,15]
G 2)$\langle yao\ X_1 / wants\ X_1 \rangle$	$\langle s \rangle$Thailand wants [3,15]
G 3)$\langle liyong\ X_1 / to\ utilize\ X_1 \rangle$	$\langle s \rangle$Thailand wants to utilize [4,15]
4)$\langle zhe\ bi\ qian\ X_1 / this\ money\ X_1 \rangle$	$\langle s \rangle$Thailand wants to utilize this money [7,15]
5)$\langle X_1\ zhuru\ geng\ duo\ X_2 / to\ inject\ more\ X_2 X_1 \rangle$	$\langle s \rangle$Thailand wants to utilize this money to inject more [12,15][7,9]
6)$\langle liudong\ X_1 / circulating\ X_1 \rangle$	$\langle s \rangle$Thailand wants to utilize this money to inject more circulating [13,15][7,9]
G 7)$\langle zijin\ X_1 / capital\ X_1 \rangle$	$\langle s \rangle$Thailand wants to utilize this money to inject more circulating capital [14,15][7,9]
8)$\langle ./. \rangle$	$\langle s \rangle$Thailand wants to utilize this money to inject more circulating capital . [7,9]
9)$\langle xiang\ jingji / to\ the\ economy \rangle$	$\langle s \rangle$Thailand wants to utilize this money to inject more circulating capital . to the economy$\langle /s \rangle$

Figure 2: The process of translating the Chinese sentence in Figure 3(b) in LR-Hiero. Left side shows the rules used in the derivation (G indicates glue rules as defined in (Watanabe et al., 2006)). The hypotheses column shows the translation prefix and the ordered list of yet-to-be-covered spans.

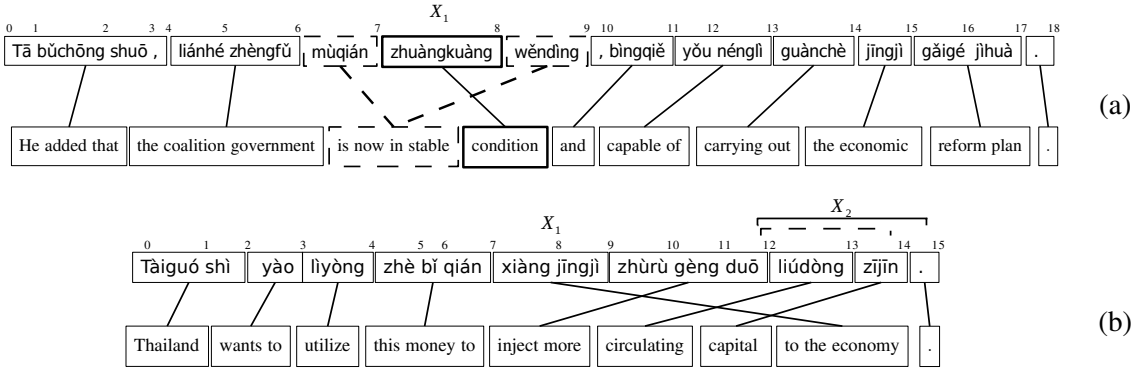


Figure 3: Two Chinese-English sentence pairs from devset data in experiments. (a) Correct rule cannot be matched to [6,18], our modifications match the rule to the first subspan [6,9] (b) LR-Hiero detects a wrong span for X_2 [12,15], we modify the rule matching match X_2 to all subspans [12,13], [12,14] and [12,15], corresponding to 3 hypotheses.

hypotheses in the queue are popped (line 26) and for each hypothesis its neighbours in the cube are added to the priority queue (line 27). Decoding finishes when stack S_n has been filled.

The language model (LM) score violates the hypotheses generation assumption of CP and can cause search errors. In Figure 1, the topmost and leftmost entry of the right cube has a score worse than many hypotheses in the left cube due to the LM score. This means the right cube has hypotheses that are ignored. This type of search error hurts LR-Hiero more than CKY-Hiero, due to the fact that hypotheses scores in LR-Hiero rely on a future cost, while CKY-Hiero uses the inside score for each hypothesis. To solve this issue for LR-Hiero we introduce the notion of *queue diversity* which is the parameter d in $GetBestHypotheses((H, R), \mathcal{F}, d)$. This parameter guarantees that each cube will produce at least d candidate hypotheses for the priority queue. d=1 in standard cube pruning for LR-Hiero (Siahbani et al., 2013). We apply the idea of diversity at queue level, before generating K best hypothesis, such that the *GetBestHypotheses* routine generates d best hypotheses from each cube and all these hypotheses are pushed to the priority queue (line 22-23). We fill each stack differently from CKY-Hiero and so queue diversity is different from lazy cube pruning (Pust and Knight, 2009) or cube growing (Huang and Chiang, 2007; Vilar and Ney, 2009; Xu and Koehn, 2012).

3 Capturing Missing Alignments

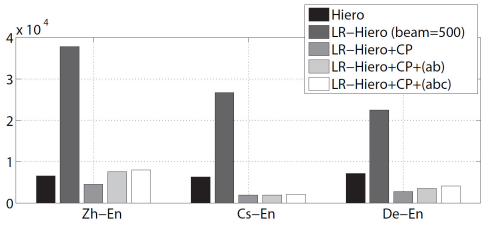
Figure 3(a) and Figure 3(b) show two examples of a common problem in LR-Hiero decoding. The decoder steps for Figure 3(b) are shown in Figure 2. The problem occurs in Step 5 of Figure 2 where rule #5 is matched to span $[7, 15]$. During decoding LR-Hiero maintains a stack (*last-in-first-out*) of yet-to-be-covered spans and tries to translate the first uncovered span (span $[7, 15]$ in Step 5). LR-Hiero should match rule #5 to span $[7, 15]$, therefore X_2 is forced to match span $[12, 15]$ which leads to the translation of span $[7, 9]$ (corresponding to X_1) being reordered around it

	Corpus	Train/Dev/Test
Cs-En	Europarl(v7) + CzEng(v0.9); News commentary(nc) 2008&2009; nc 2011	7.95M/3000/3003
De-En	Europarl(v7); WMT2006; WMT2006	1.5M/2000/2000
Zh-En	HK + GALE phase-1; MTC part 1&3; MTC part 4	2.3M/1928/919

Table 1: Corpus statistics in number of sentences. Tuning and test sets for Chinese-English has 4 references.

Model	Cs-En	De-En	Zh-En
Hiero	20.77	25.72	27.65
LR-Hiero (Watanabe et al., 2006)	20.72	25.10	25.99
LR-Hiero+CP (Siahbani et al., 2013)	20.15	24.83	-
LR-Hiero+CP (QD=1)	20.68	25.14	24.44
LR-Hiero+CP (QD=15)	-	-	26.10
LR-Hiero+CP+(ab)	20.88	25.22	26.55
LR-Hiero+CP+(abc)	20.89	25.22	26.52

(a) BLEU scores for different baselines and modifications of this paper. QD=15 for Zh-En in last three rows.



(b) Average number of language model queries.

Table 2: (a) BLEU (b) LM calls

causing the incorrect translation in Step 9. If we use the same set of rules for translation in Hiero (CKY-based decoder), the decoder is able to generate the correct translation for span $[7, 14]$ (it works bottom-up and generate best translation for each source span). Then it combines translation of $[7, 14]$ with translation of spans $[0, 7]$ and $[14, 15]$ using glue rules (monotonic combination).

In Figure 3(a) monotonic translations after span $[6, 9]$ are out of reach of the LR-Hiero decoder which has to use the non-terminals to support the reordering within span $[6, 9]$. In this example the first few phrases are translated monotonically, then for span $[6, 18]$ we have to apply rule $\langle$muqian X_1 wending, is now in stable $X_1\rangle$ to obtain the correct translation. But this rule cannot be matched to span $[6, 18]$ and the decoder fails to generate the correct translation. While CKY-Hiero can apply this rule to span $[6, 9]$, generate correct translation for this span and monotonically combine it with translation of other spans ($[0, 6]$, $[9, 18]$).

In both these cases, CKY-Hiero has no difficulty in reaching the target sentence *with the same GNF rules*. The fact that we have to process spans as they appear in the stack in LR-Hiero means that we cannot combine arbitrary adjacent spans to deal with such cases. So purely bottom-up decoders such as CKY-Hiero can capture the alignments in Figure 3 but LR-Hiero cannot.

We extend the LR-Hiero decoder to handle such cases by making the GNF grammar more expressive. Rules are partitioned to three types based on the right boundary in the source and target side. The rhs after the $\Rightarrow$ shows the new rules we create within the decoder using a new non-terminal X_r to match the right boundary.

$$\begin{aligned} &\text{(a)}\ \langle \gamma\bar{a}, \bar{b}\beta\rangle \Rightarrow \langle \gamma\bar{a}X_r, \bar{b}\beta X_r\rangle \\ &\text{(b)}\ \langle \gamma X_n, \bar{b}\beta X_n\rangle \Rightarrow \langle \gamma X_n X_r, \bar{b}\beta X_n X_r\rangle \\ &\text{(c)}\ \langle \gamma X_n, \bar{b}\beta X_m\rangle \Rightarrow \langle \gamma X_n X_r, \bar{b}\beta X_m X_r\rangle \end{aligned} \quad (1)$$

where γ is a string of terminals and non-terminals, $\bar{a}$ and $\bar{b}$ are terminal sequences of source and target respectively, β is a possibly empty sequence of non-terminals and X_n and X_m are different non-terminals distinct from X_r[3]. The extra non-terminal X_r lets us add a new yet-to-be-covered span to the bottom of the stack at each rule application which lets us match any two adjacent spans just as in CKY-Hiero. This captures the missing alignments that could not be previously captured in the LR-Hiero decoder[4].

In Table 4 we translated devset sentences using forced decoding to show that our modifications to LR-Hiero in this section improves the alignment coverage when compared to CKY-Hiero.

4 Experiments

We evaluate our modifications to LR-Hiero decoder on three language pairs (Table 1): German-English (De-En), Czech-English (Cs-En) and Chinese-English (Zh-En).

[3]In rule type (c) X_n will be in β and X_m will be in γ.

[4]For the sake of simplicity, in rule type (b) we can merge X_n and X_r as they are in the same order on both source and target side.

We use a 5-gram LM trained on the Gigaword corpus and use KenLM (Heafield, 2011). We tune weights by minimizing BLEU loss on the dev set through MERT (Och, 2003) and report BLEU scores on the test set. Pop limit for Hiero and LR-Hiero+CP is 500 and beam size LR-Hiero is 500. Other extraction and decoder settings such as maximum phrase length, etc. were identical across settings. To make the results comparable we use the same feature set for all baselines, Hiero as well (including new features proposed by (Siahbani et al., 2013)).

We use 3 baselines: (i) our implementation of (Watanabe et al., 2006): LR-Hiero with beam search (LR-Hiero) and (ii) LR-Hiero with cube pruning (Siahbani et al., 2013): (LR-Hiero+CP); and (iii) Kriya, an open-source implementation of Hiero in Python, which performs comparably to other open-source Hiero systems (Sankaran et al., 2012).

Table 3 shows model sizes for LR-Hiero (GNF) and Hiero (SCFG). Typical Hiero rule extraction excludes phrase-pairs with unaligned words on boundaries (loose phrases). We use similar rule extraction as Hiero, except that exclude non-GNF rules and include loose phrase-pairs as terminal rules.

Table 2a shows the translation quality of different systems in terms of BLEU score. Row 3 is from (Siahbani et al., 2013)[5]. As we discussed in Section 2, LR-Hiero+CP suffers from severe search errors on Zh-En (1.5 BLEU) but using queue diversity (QD=15) we fill this gap. We use the same QD(=15) in next rows for Zh-en. For Cs-En and De-En we use regular cube pruning (QD=1), as it works as well as beam search (compare rows 4 and 2).

We measure the benefit of the new modified rules from Section 3: (*ab*): adding modifications for rules type (a) and (b); (*abc*): modification of all rules. We can see that for all language pairs (ab) constantly improves performance of LR-Hiero, significantly better than LR-Hiero+CP and LR-Hiero (p-value$<$0.05) on Cs-En and Zh-En, evaluated by MultEval (Clark et al., 2011). But modifying rule type (c) does not show any improvement due to spurious ambiguity created by type (c) rules.

[5]We report results on Cs-En and De-En in (Siahbani et al., 2013). Row 4 is the same translation system as row 3 (LR-Hiero+CP). We achieve better results than our previous work (Siahbani et al., 2013) (row 4 vs. row 3) due to bug corrections and adding loose phrases as terminal rules.

Model	**Cs-En**	**De-En**	**Zh-En**
Hiero	1,961.6	858.5	471.9
LR-Hiero	266.5	116.0	100.9

Table 3: Model sizes (millions of rules).

Model	Cs-En	De-En	Zh-En
Hiero	318	351	187
LR-Hiero	278	300	132
LR-Hiero+(abc)	338	361	174

Table 4: No. of sentence covered in forced decoding of a sample of sentences from the devset. We improve the coverage by 31% for Chinese-English and more than 20% for the other two language pairs.

Figure 2b shows the results in terms of average number of language model queries on a sample set of 50 sentences from test sets. All of the baselines use the same wrapper to KenLM (Heafield, 2011) to query the language model, and we have instrumented the wrapper to count the statistics. In (Siahbani et al., 2013) we discuss that LR-Hiero with beam search (Watanabe et al., 2006) does not perform at the same level of state-of-the-art Hiero (more LM calls and less translation quality). As we can see in this figure, adding new modified rules slightly increases the number of language model queries on Cs-En and De-En so that LR-Hiero+CP still works 2 to 3 times faster than Hiero. On Zh-En, LR-Hiero+CP applies queue diversity (QD=15) which reduces search errors and improves translation quality but increases the number of hypothesis generation as well. LR-Hiero+CP with our modifications works substantially faster than LR-Hiero while obtain significantly better translation quality on Zh-En.

Comparing Table 2a with Figure 2b we can see that overall our modifications to LR-Hiero decoder significantly improves the BLEU scores compared to previous LR decoders for Hiero. We obtain comparable results to CKY-Hiero for Cs-En and De-En and remarkably improve results on Zh-En, while at the same time making 2 to 3 times less LM calls on Cs-En and De-En compared to CKY-Hiero.

Acknowledgments

This research was partially supported by NSERC, Canada RGPIN: 262313 and RGPAS: 446348 grants to the second author. The authors wish to thank Baskaran Sankaran for his valuable discussions and the anonymous reviewers for their helpful comments.

References

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 176–181, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102, February.

Yang Feng, Yang Liu, Qun Liu, and Trevor Cohn. 2012. Left-to-right tree-to-string decoding with prediction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1191–1200, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michel Galley and Christopher D. Manning. 2010. Accurate non-hierarchical phrase-based translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 966–974, Los Angeles, California, June. Association for Computational Linguistics.

Kenneth Heafield, Hieu Hoang, Philipp Koehn, Tetsuo Kiso, and Marcello Federico. 2011. Left language model state for syntactic machine translation. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 183–190, San Francisco, California, USA, 12.

Kenneth Heafield, Philipp Koehn, and Alon Lavie. 2013. Grouping language model boundary words to speed K-Best extraction from hypergraphs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia, USA, 6.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *In Proc. of the Sixth Workshop on Statistical Machine Translation*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *In ACL 07*.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 273–283, Cambridge, MA, October. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *AMTA*, pages 115–124.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Pust and Kevin Knight. 2009. Faster mt decoding through pervasive laziness. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 141–144, Boulder, Colorado, June. Association for Computational Linguistics.

Baskaran Sankaran, Majid Razmara, and Anoop Sarkar. 2012. Kriya - an end-to-end hierarchical phrase-based mt system. *The Prague Bulletin of Mathematical Linguistics (PBML)*, 97(97):83–98, apr.

Maryam Siahbani, Baskaran Sankaran, and Anoop Sarkar. 2013. Efficient left-to-right hierarchical phrase-based translation with improved reordering. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, USA, October. Association for Computational Linguistics.

David Vilar and Hermann Ney. 2009. On lm heuristics for the cube growing algorithm. In *Annual Conference of the European Association for Machine Translation*, pages 242–249, Barcelona, Spain, may.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase-based translation. In *Proc. of ACL*.

Wenduan Xu and Philipp Koehn. 2012. Extending hiero decoding in moses with cube growing. *Prague Bull. Math. Linguistics*, 98:133–.

Reordering Model for Forest-to-String Machine Translation

Martin Čmejrek
IBM Watson Group
Prague, Czech Republic
martin.cmejrek@us.ibm.com

Abstract

In this paper, we present a novel extension of a forest-to-string machine translation system with a reordering model. We predict reordering probabilities for every pair of source words with a model using features observed from the input parse forest. Our approach naturally deals with the ambiguity present in the input parse forest, but, at the same time, takes into account only the parts of the input forest used by the current translation hypothesis. The method provides improvement from 0.6 up to 1.0 point measured by (T − B)/2 metric.

1 Introduction

Various commonly adopted statistical machine translation (SMT) approaches differ in the amount of linguistic knowledge present in the rules they employ.

Phrase-based (Koehn et al., 2003) models are strong in lexical coverage in local contexts, and use external models to score reordering options (Tillman, 2004; Koehn et al., 2005).

Hierarchical models (Chiang, 2005) use lexicalized synchronous context-free grammar rules to produce local reorderings. The grammaticality of their output can be improved by additional reordering models scoring permutations of the source words. Reordering model can be either used for source pre-ordering (Tromble and Eisner,), integrated into decoding via translation rules extension (Hayashi et al., 2010), additional lexical features (He et al.,), or using external sources of information, such as source syntactic features observed from a parse tree (Huang et al., 2013).

Tree-to-string (T2S) models (Liu et al., 2006; Galley et al., 2006) use rules with syntactic structures, aiming at even more grammatically appropriate reorderings.

Forest-to-string (F2S) systems (Mi et al., 2008; Mi and Huang, 2008) use source syntactic forest as the input to overcome parsing errors, and to alleviate sparseness of translation rules.

The parse forest may often represent several meanings for an ambiguous input that may need to be transtated differently using different word orderings. The following example of an ambiguous Chinese sentence with ambiguous part-of-speech labeling motivates our interest in the reordering model for the F2S translation.

tǎolùn (0)	hùi (1)	zěnmeyàng (2)
discussion/NN discuss/VV	meeting/NN will/VV	how/VV

There are several possible meanings based on the different POS tagging sequences. We present translations for two of them, together with the indices to their original source words:

(a) NN NN VV:
How$_2$ *was*$_2$ *the*$_0$ *discussion*$_0$ *meeting*$_1$*?*

(b) VV VV VV:
Discuss$_0$ *what*$_2$ *will*$_1$ *happen*$_1$.

A T2S system starts from a single parse corresponding to one of the possible POS sequences, the same tree can be used to predict word reorderings. On the other hand, a F2S system deals with the ambiguity through exploring translation hypotheses for all competing parses representing the different meanings. As our example suggests, different meanings also tend to reorder differently

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 227–232,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

id	rule
r_1	NP(tǎolùn/NN) → discussion
r_2	NP(hùi/NN) → meeting
r_3	NP(x_1:NP x_2:NP) → the x_1 x_2
r_4	IP(x_1:NP zěnmeyàng/VV) → how was x_1
r_5	IP(hùi/VV zěnmeyàng/VV) → what will happen
r_6	IP(tǎolùn/VV x_1:IP) → discuss x_1

Table 1: Tree-to-string translation rules (without internal structures).

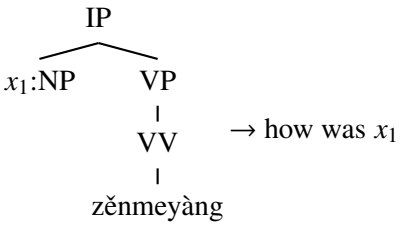


Figure 1: Tree-to-string rule r_4.

during translation. First, the reordering model suitable for F2S translation should allow for translation of all meanings present in the input. Second, as the process of deriving a partial translation hypothesis rules out some of the meanings, the reordering model should restrict itself to features originating in the relevant parts of the input forest. Our work presents a novel technique satisfying both these requirements, while leaving the disambuiguation decision up to the model using global features.

The paper is organized as follows: We briefly overview the F2S and Hiero translation models in Section 2, present the proposed forest reordering model in Section 3, describe our experiment and present results in Section 4.

2 Translation Models

Forest-to-string translation (Mi et al., 2008) is an extension of the tree-to-string model (Liu et al., 2006; Huang et al., 2006) allowing it to use a packed parse forest as the input instead of a single parse tree.

Figure 1 shows a tree-to-string **translation rule** (Huang et al., 2006), which is a tuple $\langle lhs(r), rhs(r), \psi(r)\rangle$, where $lhs(r)$ is the source-side tree fragment, whose internal nodes are labeled by nonterminal symbols (like NP), and whose frontier nodes are labeled by source-language words (like "zěnmeyàng") or variables from a finite set $\mathcal{X} = \{x_1, x_2, \ldots\}$; $rhs(r)$ is the target-side string expressed in target-language words (like "how was") and variables; and $\psi(r)$ is a mapping from $\mathcal{X}$ to nonterminals. Each variable $x_i \in \mathcal{X}$ occurs *exactly once* in $lhs(r)$ and *exactly once* in $rhs(r)$.

The Table 1 lists all rules necessary to derive translations (a) and (b), with their internal structure removed for simplicity.

Typically, an F2S system translates in two steps (shown in Figure 2): parsing and decoding. In the parsing step, the source language input is converted into a *parse forest* (A). In the decoding step, we first convert the parse forest into a *translation forest* F^t in (B) by using the fast pattern-matching technique (Zhang et al., 2009). Then the decoder uses dynamic programing with beam search and cube pruning to find the approximation to the best scoring derivation in the translation forest, and outputs the target string.

3 Forest Reordering Model

In this section, we describe the process of applying the reordering model scores. We score pairwise translation reorderings for every pair of source words similarly as described by Huang et al. (2013). In their approach, an external model of ordering distributions of *sibling* constituent pairs predicts the reordering of word pairs. Our approach deals with parse forests rather than with single trees, thus we have to model the scores differently. We model ordering distributions for every pair of *close relatives*–nodes in the parse forest that may occur together as frontier nodes of a single matching rule. We further condition the distribution on a third node–a common ancestor of the node pair that corresponds to the root node of the matching rule. This way our external model takes into acount the syntactic context of the hypothesis. For example, nodes $\mathrm{NP}_{0,1}$ and $\mathrm{NP}_{1,2}$ are close relatives, $\mathrm{NP}_{0,2}$ and $\mathrm{IP}_{0,3}$ are their common ancestors; $\mathrm{NP}_{0,1}$ and $\mathrm{VV}_{2,3}$ are close relatives, $\mathrm{IP}_{0,3}$ is their common ancestor; $\mathrm{NP}_{0,1}$ and $\mathrm{VV}_{1,2}$ are not close relatives.

More formally, let us have an input sentence $(w_0, ..., w_n)$ and its translation hypothesis h. For every i and j such that $0 \leq i < j \leq n$ we assume that the translations of w_i and w_j are in the hypothesis h either in the same or inverted ordering $o_{ij} \in \{Inorder, Reorder\}$, with a probability $P_{order}(o_{ij}|h)$. Conditioning on h signifies that the probabilistic model takes the current hypothesis as a parameter. The reordering score of the entire hy-

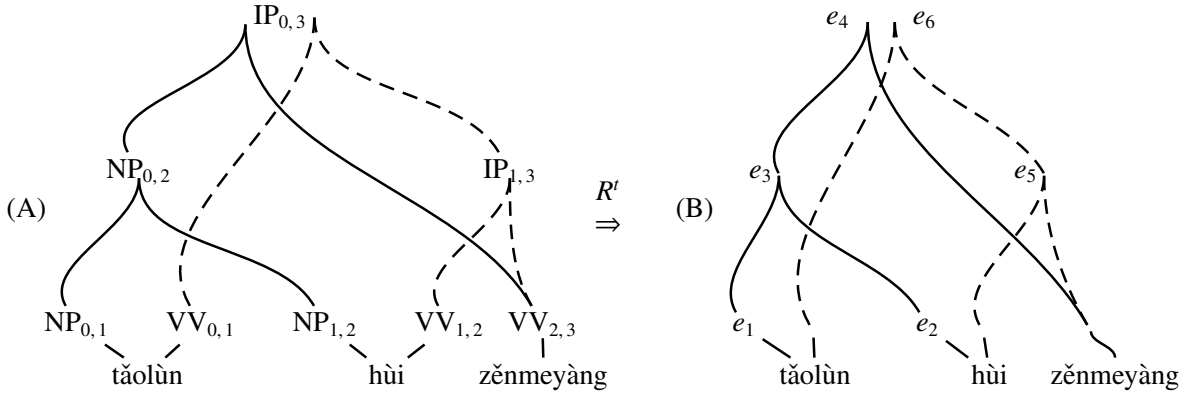


Figure 2: Parse and translation hypergraphs. (A) The parse forest of the example sentence. Solid hyperedges denote the best parse, dashed hyperedges denote the second best parse. Unary edges were collapsed. (B) The corresponding translation forest F^t after applying the tree-to-string translation rule set R^t. Each translation hyperedge (e.g. e_4) has the same index as the corresponding rule (r_4). The forest-to-string system can produce the example translation (a) (solid derivation: r_1, r_2, r_3, and r_4) and (b) (dashed derivation: r_5, r_6).

pothesis $f_{order}(h)$ is then computed as

$$f_{order} = \sum_{0 \le i < j \le n} -\log P_{order}(o_{ij} = o^h_{ij} \mid h), \quad (1)$$

where o^h_{ij} denotes the actual ordering used in h.

The score f_{order} can be computed recursively by dynamic programing during the decoding. As an example, we show in Table 2 reordering probabilities retrieved in decoding of our sample sentence.

(a) If h is a hypothesis formed by a single translation rule r with no frontier nonterminals, we evaluate all word pairs w_i and w_j covered by h such that $i < j$. For each such pair we find the frontier nodes x and y matched by r such that x spans exactly w_i and y spans exactly w_j. (In this case, x and y match preterminal nodes, each spanning one position). We also find the node z matching the root of r. Then we directly use the Equation 1 to compute the score using an external model $P_{order}(o_{ij}|xyz)$ to estimate the probability of reordering the relative nodes. For example, when applying rule r_5, we use the ordering distribution $P_{order}(o_{1,2}|\mathrm{VV}_{1,2}, \mathrm{VV}_{2,3}, \mathrm{IP}_{1,3})$ to score reorderings of hùi and zěnmeyàng.

(b) If h is a hypothesis formed by a T2S rule with one or more frontier nonterminals, we evaluate all word pairs as follows: If both w_i and w_j are spanned by the same frontier nonterminal (e.g., tǎolùn and hùi when applying the rule r_4), the score f_{order} had been already computed for the underlying subhypothesis, and therefore was already included in the total score. Otherwise, we compute the word pair ordering cost. We find the close relatives x and y representing each w_i and w_j. If w_i is matched by a terminal in r, we select x as the node matching r and spanning exactly w_i. If w_i is spanned by a frontier nonterminal in r (meaning that it was translated in a subhypothesis), we select x as the node matching that nonterminal. We proceed identically for w_j and y. For example, when applying the rule r_4, the word zěnmeyàng will be represented by the node $\mathrm{VV}_{2,3}$, while tǎolùn and hùi will be represented by the node $\mathrm{NP}_{0,2}$.

Note that the ordering o^h_{ij} cannot be determined in some cases, sometimes a source word does not produce any translation, or the translation of one word is entirely surrounded by the translations of another word. A weight corresponding to the binary discount feature $f_{o_{unknown}}$ is added to the score for each such case.

The external model $P_{order}(o_{ij}|xyz)$ is implemented as a maximum entropy model. Features of the model are observed from paths connecting node z with nodes x and y as follows: First, we pick paths $z \to x$ and $z \to y$. Let z' be the last node shared by both paths (the closest common ancestor of x and y). Then we distinguish three types of path: (1) The *common prefix* $z \to z'$ (it may have zero length), the *left path* $z \to x$, and the *right path* $z \to y$. We observe the following features on each path: the syntactic labels of the nodes, the production rules, the spans of nodes, a list of stop words immediately preceding and following the span of the node. We merge the features observed from different paths $z \to x$ and $z \to y$. This approach

rule	word pair	order	probability
a) how$_2$ was$_2$ the discussion$_0$ meeting$_1$			
r_3	(tǎolùn,hùi)	Inorder	$P_{order}\,(o_{0,1}\|\mathrm{NP}_{0,1}, \mathrm{NP}_{1,2}, \mathrm{NP}_{0,2})$
r_4	(tǎolùn,zěnmeyàng)	Reorder	$P_{order}\,(o_{0,2}\|\mathrm{NP}_{0,2}, \mathrm{VV}_{2,3}, \mathrm{IP}_{0,3})$
	(hùi,zěnmeyàng)	Reorder	$P_{order}\,(o_{1,2}\|\mathrm{NP}_{0,2}, \mathrm{VV}_{2,3}, \mathrm{IP}_{0,3})$
b) discuss$_0$ what$_2$ will$_1$ happen$_1$			
r_5	(hùi, zěnmeyàng)	Reorder	$P_{order}\,(o_{1,2}\|\mathrm{VV}_{1,2}, \mathrm{VV}_{2,3}, \mathrm{IP}_{1,3})$
r_6	(tǎolùn, hùi)	Inorder	$P_{order}\,(o_{0,1}\|\mathrm{VV}_{0,1}, \mathrm{IP}_{1,3}, \mathrm{IP}_{0,3})$
	(tǎolùn, zěnmeyàng	Inorder	$P_{order}\,(o_{0,2}\|\mathrm{VV}_{0,1}, \mathrm{IP}_{1,3}, \mathrm{IP}_{0,3})$

Table 2: Example of reordering scores computed for derivations (a) and (b).

ignores the internal structure of each rule[1], relying on frontier node annotation. On the other hand it is still feasible to precompute the reordering probabilities for all combinations of *xyz*.

4 Experiment

In this section we describe the setup of the experiment, and present results. Finally, we propose future directions of research.

4.1 Setup

Our baseline is a strong F2S system (Čmejrek et al., 2013) built on large data with the full set of model features including rule translation probabilities, general lexical and provenance translation probabilities, language model, and a variety of sparse features. We build it as follows. The training corpus consists of 16 million sentence pairs available within the DARPA BOLT Chinese-English task. The corpus includes a mix of newswire, broadcast news, webblog data coming from various sources such as LDC, HK Law, HK Hansard and UN data. The Chinese text is segmented with a segmenter trained on CTB data using conditional random fields (CRF).

Bilingual word alignments are trained and combined from two sources: GIZA (Och, 2003) and maximum entropy word aligner (Ittycheriah and Roukos, 2005).

Language models are trained on the English side of the parallel corpus, and on monolingual corpora, such as Gigaword (LDC2011T07) and Google News, altogether comprising around 10 billion words.

We parse the Chinese part of the training data with a modified version of the Berkeley parser (Petrov and Klein, 2007), then prune the obtained parse forests for each training sentence with the marginal probability-based inside-outside algorithm to contain only $3n$ CFG nodes, where n is the sentence length.

We extract tree-to-string translation rules from forest-string sentence pairs using the forest-based GHKM algorithm (Mi and Huang, 2008; Galley et al., 2004).

In the decoding step, we use larger input parse forests than in training, we prune them to contain $10n$ nodes. Then we use fast pattern-matching (Zhang et al., 2009) to convert the parse forest into the translation forest.

The proposed reordering model is trained on 100,000 automatically aligned forest-string sentence pairs from the parallel training data. These sentences provide 110M reordering events that are used by `megam` (Daumé III, 2004) to train the maximum entropy model.

The current implementation of the reordering model requires offline preprocessing of the input hypergraphs to precompute reordering probabilities for applicable triples of nodes (x, y, z). Since the number of levels in the syntactic trees in T2S rules is limited to 4, we only need to consider such triples, where z is up to 4 levels above x or y.

We tune on 1275 sentences, each with 4 references, from the LDC2010E30 corpus, initially released under the DARPA GALE program.

We combine two evaluation metrics for tuning and testing: B (Papineni et al., 2002) and T (Snover et al., 2006). Both the baseline and the reordering experiments are optimized with MIRA (Crammer et al., 2006) to maximize (T - B)/2.

We test on three different test sets: GALE Web test set from LDC2010E30 corpus (1239 sentences, 4 references), NIST MT08 Newswire

[1]Only to some extent, the rule still has to match the input forest, but the reordering model decides based on the sum of paths observed between the root and frontier nodes.

System	GALE Web			MT08 Newswire			MT08 Web		
	$\frac{T-B}{2}$	B	T	$\frac{T-B}{2}$	B	T	$\frac{T-B}{2}$	B	T
F2S	8.8	36.1	53.7	5.6	40.6	51.8	12.0	31.3	55.3
+Reordering	8.2	36.4	52.7	4.8	41.7	50.5	11.0	31.7	53.7
Δ	*-0.6*	*+0.3*	*-1.0*	*-0.8*	*+1.1*	*-1.3*	*-1.0*	*+0.4*	*-1.6*

Table 3: Results.

portion (691 sentences, 4 references), and NIST MT08 Web portion (666 sentences, 4 references).

4.2 Results

Table 3 shows all results of the baseline and the system extended with the forest reordering model. The (T − B)/2 score of the baseline system is 12.0 on MT08 Newswire, showing that it is a strong baseline. The system with the proposed reordering model significantly improves the baseline by 0.6, 0.8, and 1.0 (T − B)/2 points on GALE Web, MT08 Newswire, and MT08 Web.

The current approach relies on frontier node annotations, ignoring to some extent the internal structure of the T2S rules. As part of future research, we would like to compare this approach with the one that takes into accout the internal structure as well.

5 Conclusion

We have presented a novel reordering model for the forest-to-string MT system. The model deals with the ambiguity of the input forests, but also predicts specifically to the current parse followed by the translation hypothesis. The reordering probabilities can be precomputed by an offline process, allowing for efficient scoring in runtime. The method provides improvement from 0.6 up to 1.0 point measured by (T − B)/2 metrics.

Acknowledgments

We thank Jiří Havelka for proofreading and helpful suggestions. We would like to acknowledge the support of DARPA under Grant HR0011-12-C-0015 for funding part of this work. The views, opinions, and/or findings contained in this article are those of the author and should not be interpreted as representing the official views or policies, either expressed or implied, of the DARPA.

References

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the ACL*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7.

Hal Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://pub.hal3.name#daume04cg-bfgs`, implementation available at `http://hal3.name/megam/`.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of the HLT-NAACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the COLING-ACL*.

Katsuhiko Hayashi, Hajime Tsukada, Katsuhito Sudoh, Kevin Duh, and Seiichi Yamamoto. 2010. Hierarchical Phrase-based Machine Translation with Word-based Reordering Model. In *Proceedings of the COLING*.

Zhongjun He, Yao Meng, and Hao Yu. Maximum entropy based phrase reordering for hierarchical phrase-based translation. In *Proceedings of the EMNLP*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the AMTA*.

Zhongqiang Huang, Jacob Devlin, and Rabih Zbib. 2013. Factored soft source syntactic constraints for hierarchical machine translation. In *Proceeedings of the EMNLP*.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of the HLT and EMNLP*.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *Proceedings of the IWSLT*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING-ACL*.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL: HLT*.

Franz Joseph Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

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

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the AMTA*.

Christoph Tillman. 2004. A unigram orientation model for statistical machine translation. Proceedings of the HLT-NAACL.

Roy Tromble and Jason Eisner. Learning linear ordering problems for better translation. In *Proceedings of the EMNLP*.

Martin Čmejrek, Haitao Mi, and Bowen Zhou. 2013. Flexible and efficient hypergraph interactions for joint hierarchical and forest-to-string decoding. In *Proceedings of the EMNLP*.

Hui Zhang, Min Zhang, Haizhou Li, and Chew Lim Tan. 2009. Fast translation rule matching for syntax-based statistical machine translation. In *Proceedings of EMNLP*, pages 1037–1045, Singapore, August.

Aligning context-based statistical models of language with brain activity during reading

Leila Wehbe[1,2], **Ashish Vaswani**[3], **Kevin Knight**[3] **and Tom Mitchell**[1,2]
[1] Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA
[2] Center for the Neural Basis of Computation, Carnegie Mellon University, Pittsburgh, PA
[3] Information Sciences Institute, University of Southern California, Los Angeles, CA
lwehbe@cs.cmu.edu, vaswani@usc.edu, knight@isi.edu, tom.mitchell@cs.cmu.edu

Abstract

Many statistical models for natural language processing exist, including context-based neural networks that (1) model the previously seen context as a latent feature vector, (2) integrate successive words into the context using some learned representation (embedding), and (3) compute output probabilities for incoming words given the context. On the other hand, brain imaging studies have suggested that during reading, the brain (a) continuously builds a context from the successive words and every time it encounters a word it (b) fetches its properties from memory and (c) integrates it with the previous context with a degree of effort that is inversely proportional to how probable the word is. This hints to a parallelism between the neural networks and the brain in modeling context (1 and a), representing the incoming words (2 and b) and integrating it (3 and c). We explore this parallelism to better understand the brain processes and the neural networks representations. We study the alignment between the latent vectors used by neural networks and brain activity observed via Magnetoencephalography (MEG) when subjects read a story. For that purpose we apply the neural network to the same text the subjects are reading, and explore the ability of these three vector representations to predict the observed word-by-word brain activity.

Our novel results show that: before a new word i is read, brain activity is well predicted by the neural network latent representation of context and the predictability decreases as the brain integrates the word and changes its own representation of context. Secondly, the neural network embedding of word i can predict the MEG activity when word i is presented to the subject, revealing that it is correlated with the brain's own representation of word i. Moreover, we obtain that the activity is predicted in different regions of the brain with varying delay. The delay is consistent with the placement of each region on the processing pathway that starts in the visual cortex and moves to higher level regions. Finally, we show that the output probability computed by the neural networks agrees with the brain's own assessment of the probability of word i, as it can be used to predict the brain activity after the word i's properties have been fetched from memory and the brain is in the process of integrating it into the context.

1 Introduction

Natural language processing has recently seen a surge in increasingly complex models that achieve impressive goals. Models like deep neural networks and vector space models have become popular to solve diverse tasks like sentiment analysis and machine translation. Because of the complexity of these models, it is not always clear how to assess and compare their performances as they might be useful for one task and not the other. It is also not easy to interpret their very high-dimensional and mostly unsupervised representations. The brain is another computational system that processes language. Since we can record brain activity using neuroimaging, we propose a new direction that promises to improve our understanding of both how the brain is processing language and of what the neural networks are modeling by aligning the brain data with the neural networks representations.

In this paper we study the representations of two kinds of neural networks that are built to predict the incoming word: recurrent and finite context models. The first model is the Recurrent Neural Network Language Model (Mikolov et al., 2011) which uses the entire history of words to model context. The second is the Neural Probabilistic Language Model (NPLM) which uses limited context constrained to the recent words (3 grams or 5 grams). We trained these models on a large Harry Potter fan fiction corpus and we then used them to predict the words of chapter 9 of *Harry Potter and the Sorcerer's Stone* (Rowling, 2012). In parallel, we ran an MEG experiment in which 3 subject read the words of chapter 9 one by one while their brain activity was recorded. We then looked for the alignment between the word-by-word vectors produced by the neural networks and the word-by-word neural activity recorded by MEG.

Our neural networks have 3 key constituents: a hidden layer that summarizes the history of the previous words ; an embeddings vector that summarizes the (constant) properties of a given word and finally the output probability of a word given

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 233–243,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

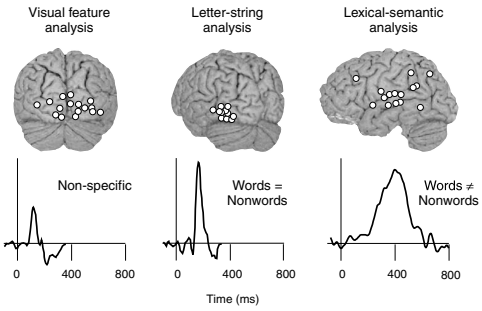


Figure 1: Cortical dynamics of silent reading. This figure is adapted from (Salmelin, 2007). Dots represent projected sources of activity in the visual cortex (left brain sketch) and the temporal cortex (right brain sketch). The curves display the mean time course of activation in the depicted source areas for different conditions. The initial visual feature analysis in the visual cortex at $\sim$100 ms is non-specific to language. Comparing responses to letter strings and other visual stimuli reveals that letter string analysis occurs around 150 ms. Finally comparing the responses to words and nonwords (made-up words) reveals lexical-semantic analysis in the temporal cortex at $\sim$200-500ms.

the context. We set out to find the brain analogs of these model constituents using an MEG decoding task. We compare the different models and their representations in terms of how well they can be used to decode the word being read from MEG data. We obtain correspondences between the models and the brain data that are consistent with a model of language processing in which brain activity encodes story context, and where each new word generates additional brain activity, flowing generally from visual processing areas to more high level areas, culminating in an updated story context, and reflecting an overall magnitude of neural effort influenced by the probability of that new word given the previous context.

1.1 Neural processes involved in reading

Humans read with an average speed of 3 words per second. Reading requires us to perceive incoming words and gradually integrate them into a representation of the meaning. As words are read, it takes 100ms for the visual input to reach the visual cortex. 50ms later, the visual input is processed as letter strings in a specialized region of the left visual cortex (Salmelin, 2007). Between 200-500ms, the word's semantic properties are processed (see Fig. 1). Less is understood about the cortical dynamics of word integration, as multiple theories exist (Friederici, 2002; Hagoort, 2003).

Magnetoencephalography (MEG) is a brain-imaging tool that is well suited for studying language. MEG records the change in the magnetic field on the surface of the head that is caused by a large set of aligned neurons that are changing their firing patterns in synchrony in response to a stimulus. Because of the nature of the signal, MEG recordings are directly related to neural activity and have no latency. They are sampled at a high frequency (typically 1kHz) that is ideal for tracking the fast dynamics of language processing.

In this work, we are interested in the mechanism of human text understanding as the meaning of incoming words is fetched from memory and integrated with the context. Interestingly, this is analogous to neural network models of language that are used to predict the incoming word. The mental representation of the previous context is analogous to the latent layer of the neural network which summarizes the relevant context before seeing the word. The representation of the meaning of a word is analogous to the embedding that the neural network learns in training and then uses. Finally, one common hypotheses is that the brain integrates the word with inversely proportional effort to how predictable the word is (Frank et al., 2013). There is a well studied response known as the N400 that is an increase of the activity in the temporal cortex that has been recently shown to be graded by the amount of surprisal of the incoming word given the context (Frank et al., 2013). This is analogous to the output probability of the incoming word from the neural network.

Fig. 2 shows a hypothetical activity in an MEG sensor as a subject reads a story in our experiment, in which words are presented one at a time for 500ms each. We conjecture that the activity in time window a, i.e. before word i is understood, is mostly related to the previous context before seeing word i. We also conjecture that the activity in time window b is related to understanding word i and integrating it into the context, leading to a new representation of context in window c.

Using three types of features from neural networks (hidden layer context representation, output probabilities and word embeddings) from three different models of language (one recurrent model and two finite context models), we therefore set to predict the activity in the brain in different time windows. We want to align the brain data with the various model constituents to understand where and when different types of processes are computed in the brain, and simultaneously, we want to

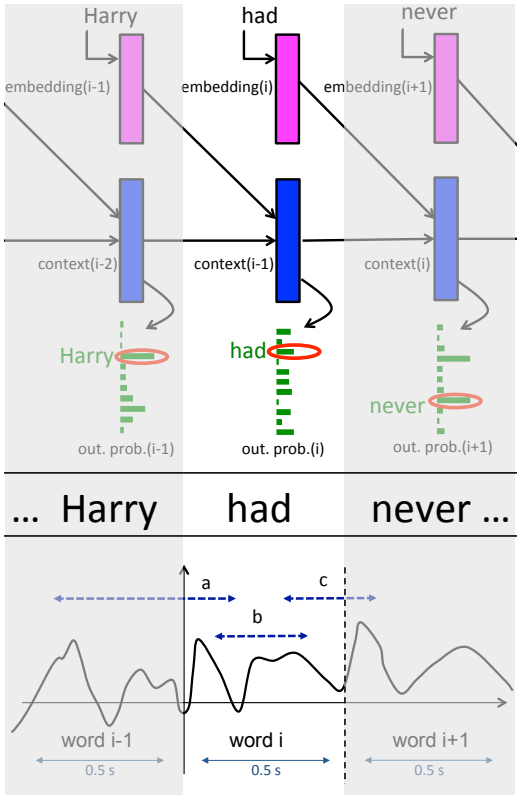


Figure 2: [Top] Sketch of the updates of a neural network reading chapter 9 after it has been trained. Every word corresponds to a fixed embedding vector (magenta). A context vector (blue) is computed before the word is seen given the previous words. Given the context vector, the probability of every word can be computed (symbolized by the histogram in green). We only use the output probability of the actual word (red circle). [Bottom] Hypothetical activity in an MEG sensor when the subject reads the corresponding words. The time periods approximated as a, b and c can be tested for information content relating to: the context of the story before seeing word i (modeled by the context vector at i), the representation of the properties of word i (the embedding of word i) and the integration of word i into the context (the output probability of word i). The periods drawn here are only a conjecture on the timings of such cognitive events.

use the brain data to shed light on what the neural network vectors are representing.

Related work

Decoding cognitive states from brain data is a recent field that has been growing in popularity. Most decoding studies that study language use functional Magnetic Resonance Imaging (fMRI), while some studies use MEG. MEG's high temporal resolution makes it invaluable for looking at the dynamics of language understanding. (Sudre et al., 2012) decode from MEG the word a subject is reading. The authors estimate from the MEG data the semantic features of the word and use these as an intermediate step to decode what the word is. This is in principle similar to the classification approach we follow, as we will also use the feature vectors as an intermediate step for word classification. However the experimental paradigm in (Sudre et al., 2012) is to present to the subjects single isolated words and to find how the brain represents their semantic features; whereas we have a much more complex and "naturalistic" experiment in which the subjects read a non-artificial passage of text, and we look at processes that exceed individual word processing: the construction of the meanings of the successive words and the prediction/integration of incoming words.

In (Frank et al., 2013), the amount of surprisal that a word has given its context is used to predict the intensity of the N400 response described previously. This is the closest study we could find to our approach. This study was concerned with analyzing the brain processes related only to surprisal while we propose a more integral account of the processes in the brain. The study also didn't address the major contribution we propose here, which is to shed light on the inner constituents of language models using brain imaging.

1.2 Recurrent and finite context neural networks

Similar to standard language models, neural language models also learn probability distributions over words given their previous context. However, unlike standard language models, words are represented as real-valued vectors in a high dimensional space. These word vectors, referred to as *word embeddings*, can be different for input and output words, and are learned from training data. Thus, although at training and test time, the input and output to the neural language models are *one-hot* representation of words, it is their embeddings that are used to compute word probability distributions. After training the embedding vectors are fixed and it is these vectors that we will use later on to predict MEG data. To predict MEG data, we will also use the latent vector representations of context that these neural networks produce, as well as the probability of the current word given the context. In this section, we will describe how recurrent neural network language models and feedforward neural probabilistic language models compute word probabilities. In the interest of space, we keep this description brief, and for details, the reader is requested to refer to the original papers describing these models.

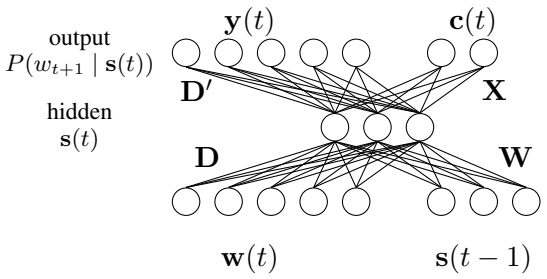


Figure 3: Recurrent neural network language model.

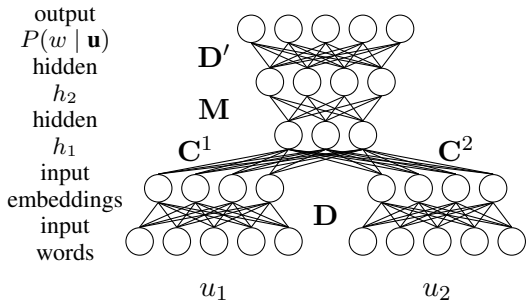


Figure 4: Neural probabilistic language model

Recurrent Neural Network Language Model

Unlike standard feedforward neural language models that only look at a fixed number of past words, recurrent neural network language models use all the previous history from position 1 to $t-1$ to predict the next word. This is typically achieved by *feedback* connections, where the hidden layer activations used for predicting the word in position $t-1$ are fed back into the network to compute the hidden layer activations for predicting the next word. The hidden layer thus stores the history of all previous words. We use the RNNLM architecture as described in Mikolov (2012), shown in Figure 3. The input to the RNNLM at position t are the one-hot representation of the current word, $\mathbf{w}(t)$, and the activations from the hidden layer at position $t-1$, $\mathbf{s}(t-1)$. The output of the hidden layer at position $t-1$ is

$$\mathbf{s}(t) = \phi\left(\mathbf{D}\mathbf{w}(t) + \mathbf{W}\mathbf{s}(t-1)\right),$$

where $\mathbf{D}$ is the matrix of input word embeddings, $\mathbf{W}$ is a matrix that transforms the activations from the hidden layer in position $t-1$, and ϕ is a sigmoid function, defined as $\phi(x) = \frac{1}{1+\exp(-x)}$, that is applied elementwise. We need to compute the probability of the next word $\mathbf{w}(t+1)$ given the hidden state $\mathbf{s}(t)$. For fast estimation of output word probabilities, Mikolov (2012) divides the computation into two stages: First, the probability distribution over *word classes* is computed, after which the probability distribution over the subset of words belonging to the class are computed. The class probability of a particular class with index m at position t is computed as:

$$P(\mathbf{c}_m(t) \mid \mathbf{s}(t)) = \frac{\exp\left(\mathbf{s}(t)\mathbf{X}\mathbf{v}_m\right)}{\sum_{c=1}^{C}\left(\exp\left(\mathbf{s}(t)\mathbf{X}\mathbf{v}_c\right)\right)},$$

where $\mathbf{X}$ is a matrix of class embeddings and $\mathbf{v_m}$ is a one-hot vector representing the class with index m. The normalization constant is computed over all classes C. Each class specifies a subset V' of words, potentially smaller than the entire vocabulary V. The probability of an output word l at position $t+1$ given that its class is m is defined as:

$$P(\mathbf{y}_l(t+1) \mid \mathbf{c}_m(t), \mathbf{s}(t)) = \frac{\exp\left(\mathbf{s}(t)\mathbf{D}'\mathbf{v}_l\right)}{\sum_{k=1}^{V'}\left(\exp\left(\mathbf{s}(t)\mathbf{D}'\mathbf{v}_k\right)\right)},$$

where $\mathbf{D}'$ is a matrix of output word embeddings and $\mathbf{v}_l$ is a one hot vector representing the word with index l. The probability of the word $\mathbf{w}(t+1)$ given its class c_i can now be computed as:

$$P(\mathbf{w}(t+1) \mid \mathbf{s}(t)) = P(\mathbf{w}(t+1) \mid c_i, \mathbf{s}(t))\, P(c_i \mid \mathbf{s}(t)).$$

Neural Probabilistic Language Model

We use the feedforward neural probabilistic language model architecture of Vaswani et al. (2013), as shown in Figure 4. Each context $\mathbf{u}$ comprises a sequence of words $\mathbf{u}_j$ ($1 \le j \le n-1$) represented as one-hot vectors, which are fed as input to the neural network. At the output layer, the neural network computes the probability $P(w \mid \mathbf{u})$ for each word w, as follows.

The output of the first hidden layer h_1 is

$$h_1 = \phi\left(\sum_{j=1}^{n-1}\mathbf{C}^j\mathbf{D}\mathbf{u}_j + \mathbf{b}_1\right),$$

where $\mathbf{D}$ is a matrix of input word embeddings which is shared across all positions, the $\mathbf{C}^j$ are the context matrices for each word in $\mathbf{u}$, $\mathbf{b}_1$ is a vector of biases with the same dimension as h_1, and ϕ is applied elementwise. Vaswani et al. (2013) use rectified linear units (Nair and Hinton, 2010) for

the hidden layers h_1 and h_2, which use the activation function $\phi(x) = \max(0, x)$.

The output of the second layer h_2 is

$$h_2 = \phi\left(\mathbf{M}h_1 + \mathbf{b}_2\right),$$

where $\mathbf{M}$ is a weight matrix between h_1 and h_2 and $\mathbf{b}_2$ is a vector of biases for h_2. The probability of the output word is computed at the output softmax layer as:

$$P(w \mid \mathbf{u}) = \frac{\exp\left(\mathbf{v}_w \mathbf{D}' h_2 + \mathbf{b}^T \mathbf{v}_w\right)}{\sum_{w'=1}^{V} \exp\left(\mathbf{v}_{w'} \mathbf{D}' h_2 + \mathbf{b}^T \mathbf{v}_{w'}\right)},$$

where $\mathbf{D}'$ is the matrix of output word embeddings, $\mathbf{b}$ is a vector of biases for every output word and $\mathbf{v}_w$ its the one hot representation of the word w in the vocabulary.

2 Methods

We describe in this section our approach. In summary, we trained the neural network models on a Harry Potter fan fiction database. We then ran these models on chapter 9 of *Harry Potter and the Sorcerer's Stone* (Rowling, 2012) and computed the context and embedding vectors and the output probability for each word. In parallel, 3 subjects read the same chapter in an MEG scanner. We build models that predict the MEG data for each word as a function of the different neural network constituents. We then test these models with a classification task that we explain below. We detect correspondences between the neural network components and the brain processes that underlie reading in the following fashion. If using a neural network vector (e.g. the RNNLM embedding vector) allows us to classify significantly better than chance in a given region of the brain at a given time (e.g. the visual cortex at time 100-200ms), then we can hypothesize a relationship between that neural network constituent and the time/location of the analogous brain process.

2.1 Training the Neural Networks

We used the freely available training tools provided by Mikolov (2012)[1] and Vaswani et al. (2013)[2] to train our RNNLM and NPLM models used in our brain data classification experiments. Our training data comprised around 67.5 million words for training and 100 thousand words for validation from the Harry Potter fan fiction database (http://harrypotterfanfiction.com). We restricted the vocabulary to the top 100 thousand words which covered all but 4 words from Chapter 9 of *Harry Potter and the Sorcerer's Stone*.

For the RNNLM, we trained models with different hidden layers and learning rates and found the RNNLM with 250 hidden units to perform best on the validation set. We extracted our word embeddings from the input matrix $\mathbf{D}$ (Figure 3). We used the default settings for all other hyper parameters.

We trained 3-gram and 5-gram NPLMs with 150 dimensional word embeddings and experimented with different number of units for the first hidden layer (h_1 in Figure 4), and different learning rates. For both the 3-gram and 5-gram models, we found 750 hidden units to perform the best on the validation set and chose those models for our final experiments. We used the output word embeddings $\mathbf{D}'$ in our experiments. We visually inspected the nearest neighbors in the 150 dimensional word embedding space for some words and didn't find the neighbors from $\mathbf{D}'$ or $\mathbf{D}$ to be distinctly better than each other. We leave the comparison of input and output embeddings on brain activity prediction for future work.

2.2 MEG paradigm

We recorded MEG data for three subjects (2 females and one male) while they read chapter 9 of *Harry Potter and the Sorcerer's Stone* (Rowling, 2012). The participants were native English speakers and right handed. They were chosen to be familiar with the material: we made sure they had read the Harry Potter books or seen the movies series and were familiar with the characters and the story. All the participants signed the consent form, which was approved by the University of Pittsburgh Institutional Review Board, and were compensated for their participation.

The words of the story were presented in rapid serial visual format (Buchweitz et al., 2009): words were presented one by one at the center of the screen for 0.5 seconds each. The text was shown in 4 experimental blocks of ∼11 minutes. In total, 5176 words were presented. Chapter 9 was presented in its entirety without modifications and each subject read the chapter only once.

One can think of an MEG machine as a large helmet, with sensors located on the helmet that

[1] http://rnnlm.org/

[2] http://nlg.isi.edu/software/nplm

record the magnetic activity. Our MEG recordings were acquired on an Elekta Neuromag device at the University of Pittsburgh Medical Center Presbyterian Hospital. This machine has 306 sensors distributed into 102 locations on the surface of the subject's head. Each location groups 3 sensors or two types: one magnometer that records the intensity of the magnetic field and two planar gradiometers that record the change in the magnetic field along two orthogonal planes[3].

Our sampling frequency was 1kHz. For preprocessing, we used Signal Space Separation method (SSS, (Taulu et al., 2004)), followed by its temporal extension (tSSS, (Taulu and Simola, 2006)).

For each subject, the experiment data consists therefore of a 306 dimensional time series of length ∼45 minutes. We averaged the signal in every sensor into 100ms non-overlapping time bins. Since words were presented for 500ms each, we therefore obtain for every word $p = 306 \times 5$ values corresponding to 306 vectors of 5 points.

2.3 Decoding experiment

To find which parts of brain activity are related to the neural network constituents (e.g. the RNNLM context vector), we run a prediction and classification experiment in a 10-fold cross validated fashion. At every fold, we train a linear model to predict MEG data as a function of one of the feature sets, using 90% of the data. On the remaining 10% of the data, we run a classification experiment.

MEG data is very noisy. Therefore, classifying single word waveforms yields a low accuracy, peaking at 60%, which might lead to false negatives when looking for correspondences between neural network features and brain data. To reveal informative features, one can boost signal by either having several repetitions of the stimuli in the experiment and then averaging (Sudre et al., 2012) or by combining the words into larger chunks (Wehbe et al., 2014). We chose the latter because the former sacrifices word and feature diversity.

At testing, we therefore repeat the following 300 times. Two sets of words are chosen randomly from the test fold. To form the first set, 20 words are sampled without replacement from the test sample (unseen by the classifier). To form the second set, the k^{th} word is chosen randomly from all words in the test fold having the same length as the k^{th} word of the first set. Since every fold of the data was used 9 times in the training phase and once in the testing phase, and since we use a high number of randomized comparisons, this averages out biases in the accuracy estimation. Classifying sets of 20 words improves the classification accuracy greatly while lowering its variance and makes it dissociable from chance performance. We compare only between words of equal length, to minimize the effect of the low level visual features on the classification accuracy.

After averaging out the results of multiple folds, we end up with average accuracies that reveal how related one of the models' constituents (e.g. the RNNLM context vector) is to brain data.

2.3.1 Annotation of the stimulus text

We have 9 sets of annotations for the words of the experiment. Each set j can be described as a matrix $\mathbf{F}_j$ in which each row i corresponds to the vector of annotations of word i. Our annotations correspond to the 3 model constituents for each of the 3 models: the hidden layer representation before word i, the output probability of word i and the learned embeddings for word i.

2.3.2 Classification

In order to align the brain processes and the different constituents of the different models, we use a classification task. The task is to classify the word a subject is reading out of two possible choices from its MEG recording. The classifier uses one type of feature in an intermediate classification step. For example, the classifier learns to predict the MEG activity for any setting of the RNNLM hidden layer. Given an unseen MEG recording for an unknown word i and two possible story words i' and i'' (one of which being the true word i), the classifier predicts the MEG activity when reading i' and i'' from their hidden layer vectors. It then assigns the label i' or i'' to the word recording i depending on which prediction is the closest to the recording. The following are the detailed steps of this complex classification task. However, for the rest of the paper the most useful point to keep in mind is that the main purpose of the classification is to find a correspondence between the brain data and a given feature set j.

1. Normalize the columns of $\mathbf{M}$ (zero mean, standard deviation = 1). Pick feature set $\mathbf{F}_j$ and normalize its columns to a minimum of 0 and a maximum of 1.

[3] In this paper, we treat these three different sensors as three different dimensions without further exploiting their physical properties.

2. Divide the data into 10 folds, for each fold b:

   (a) Isolate $\mathbf{M}^b$ and $\mathbf{F}_j^b$ as test data. The remainder $\mathbf{M}^{-b}$ and $\mathbf{F}_j^{-b}$ will be used for training[4].

   (b) Subtract the mean of the columns of $\mathbf{M}^{-b}$ from $\mathbf{M}^b$ and $\mathbf{M}^{-b}$ and the mean of the columns of $\mathbf{F}_j^{-b}$ from $\mathbf{F}_j^b$ and $\mathbf{F}_j^{-b}$

   (c) Use ridge regression to solve
   $$\mathbf{M}^{-b} = \mathbf{F}_j^{-b} \times \boldsymbol{\beta}_j^t$$
   by tuning the λ parameter to every one of the p output dimensions independently. λ is chosen via generalized cross validation (Golub et al., 1979).

   (d) Perform a binary classification. Sample from the set of words in b a set c of 20 words. Then sample from b another set of 20 words such that the k^{th} word in c and d have the same number of letters. For every sample (c,d):

      i. predict the MEG data for c and d as: $\mathbf{P}^c = \mathbf{F}_j^c \times \mathbf{\Gamma}_j^b$ and $\mathbf{P}^d = \mathbf{F}_j^d \times \mathbf{\Gamma}_j^b$

      ii. assign to $\mathbf{M}^c$ the label c or d depending on which of $\mathbf{P}^c$ or $\mathbf{P}^d$ is closest (Euclidean distance).

      iii. assign to $\mathbf{M}^d$ the label c or d depending on which of $\mathbf{P}^c$ or $\mathbf{P}^d$ is closest (Euclidean distance).

3. Compute the average accuracy.

2.3.3 Restricting the analysis spatially: a searchlight equivalent

We adapt the searchlight method (Kriegeskorte et al., 2006) to MEG. The searchlight is a discovery procedure used in fMRI in which a cube is slid over the brain and an analysis is performed in each location separately. It allows to find regions in the brain where a specific phenomenon is occurring. In the MEG sensor space, for every one of the 102 sensor locations ℓ, we assign a group of sensors g_ℓ. For every location ℓ, we identify the locations that immediately surround it in any direction (Anterior, Right Anterior, Right etc...) when looking at the 2D flat representation of the location of the sensors in the MEG helmet (see Fig. 9 for an illustration of the 2D helmet). g_ℓ therefore contains the 3 sensors at location ℓ and at the neighboring locations. The maximum number of sensors in a group is 3×9. The locations at the edge of the helmet have fewer sensors because of the missing neighbor locations.

[4]The rows from $\mathbf{M}^{-b}$ and $\mathbf{F}_j^{-b}$ that correspond to the five words before or after the test set are ignored in order to make the test set independent.

2.3.4 Restricting the analysis temporally

Instead of using the entire time course of the word, we can use only one of the corresponding 100ms time windows. Obtaining a high classification accuracy using one of the time windows and feature set j means that the analogous type of information is encoded at that time.

2.3.5 Classification accuracy by time and region

The above steps compute whole brain accuracy using all the time series. In order to perform a more precise spatio-temporal analysis, one can use only one time window m and one location ℓ for the classification. This can answer the question of when and where different information is represented by brain activity. For every location, we will use only the columns corresponding to the time point m for the sensors belonging to the group g_ℓ. Step (d) of the classification procedure is changed as such:

(d) Perform a binary classification. Sample from the set of words in b a set c of 20 words. Then sample from b another set of 20 words such that the k^{th} word in c and d have the same number of letters. For every sample (c,d), and for every setting of $\{m, \ell\}$:

   i. predict the MEG data for c and d as: $\mathbf{P}^c_{\{m,\ell\}} = \mathbf{F}_j^c \times \mathbf{\Gamma}^b_{j,\{m,\ell\}}$ and $\mathbf{P}^d_{\{m,\ell\}} = \mathbf{F}_j^d \times \mathbf{\Gamma}^b_{j,\{m,\ell\}}$

   ii. assign to $\mathbf{M}^c_{\{m,\ell\}}$ the label c or d depending on which of $\mathbf{P}^c_{\{m,\ell\}}$ or $\mathbf{P}^d_{\{m,\ell\}}$ is closest (Euclidean distance).

   iii. assign to $\mathbf{M}^d_{\{m,\ell\}}$ the label c or d depending on which of $\mathbf{P}^c_{\{m,\ell\}}$ or $\mathbf{P}^d_{\{m,\ell\}}$ is closest (Euclidean distance).

2.3.6 Statistical significance testing

We determine the distribution for chance performance empirically. Because the successive word samples in our MEG and feature matrices are not independent and identically distributed, we break the relationship between the MEG and feature matrices by shifting the feature matrices by large delays (e.g. 2000 to 2500 words) and we repeat the classification using the delayed matrices. This simulates chance performance more fairly than a permutation test because it keeps the time structure of the matrices. It was used in (Wehbe et al.,

2014) and inspired by (Chwialkowski and Gretton, 2014). For every $\{m, \ell\}$ setting we can therefore compute a standardized z-value by subtracting the mean of the shifted classifications and dividing by the standard deviation. We then compute the p-value for the true classification accuracy being due to chance. Since the three p-values for the three subjects for a given $\{m, \ell\}$ are independent, we combine them using Fisher's method for independent test statistics (Fisher, 1925). The statistics we obtain for every $\{m, \ell\}$ are dependent because they comprise nearby time and space windows. We control the false discovery rate using (Benjamini and Yekutieli, 2001) to adjust for the testing at multiple locations and time windows. This method doesn't assume any kind of independence or positive dependence.

3 Results

We present in Fig. 5 the accuracy using all the time windows and sensors. In Fig. 6 we present the classification accuracy when running the classification at every time window exclusively. In Fig. 9 we present the accuracy when running the classification using different time windows and groups of sensors centered at every one of the 102 locations.

It is important to lay down some conventions to understand the complex results in these plots. To recap, we are trying to find parallels between model constituents and brain processes. We use:

- a subset of the data (for example the time window 0-100ms and all the sensors)
- one type of feature (for example the hidden context layer from the NPLM 3g model)

and we obtain a classification accuracy A. If A is low, there is probably no relationship between the feature set and the subset of data. If A is high, it hints to an association between the subset of data and the mental process that is analogous to the feature set. For example, when using all the sensors and time window 0-100ms, along with the NPLM 3g hidden layer, we obtain an accuracy of 0.70 (higher than chance with p $< 10^{-14}$, see Fig. 6). Since the NPLM 3g hidden layer summarizes the context of the story before seeing word i, this suggests that the brain is still processing the context of the story before word i between 0-100ms.

Fig. 6 shows the accuracy for different types of features when using all of the time points and all the sensors to classify a word. We can see

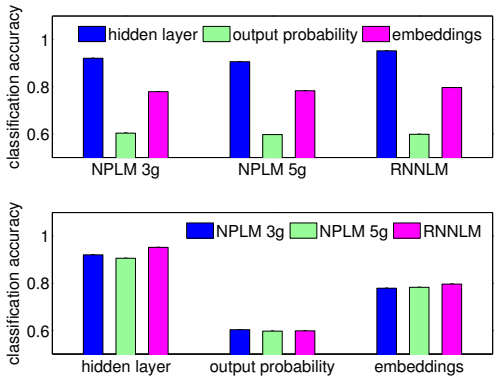


Figure 5: Average accuracy using all time windows and sensors, grouped by model (top) and type of feature (bottom). All accuracies are significantly higher than chance ($p < 10^{-8}$).

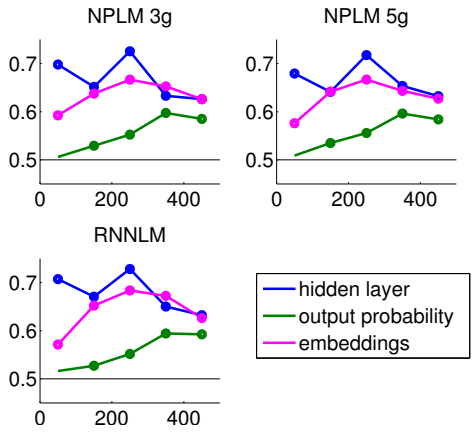


Figure 6: Average accuracy in different time windows when using different types of features as input to the classifier, for different models. Accuracy is plotted in the center of the respective time window. Points marked with a circle are significantly higher than chance accuracy for the given feature set and time window after correction.

similar classification accuracies for the three types of models, with RNNLM ahead for the hidden layer and embeddings and behind for the output probability features. The hidden layer features are the most powerful for classification. Between the three types of features, the hidden layer features are the best at capturing the information contained in the brain data, suggesting that most of the brain activity is encoding the previous context. The embedding features are the second best. Finally the output probability have the smallest accuracies. This makes sense considering that they capture much less information than the other two high dimensional descriptive vectors, as they do not represent the complex properties of the words, only a numerical assessment of their likelihood.

Fig. 6 shows the accuracy when using different windows of time exclusively, for the 100ms time

windows starting at $0, 100 \ldots 400$ms after word presentation. We can see that using the embedding vector becomes increasingly more useful for classification until 300-400ms, and then its performance starts decreasing. This results aligns with the following hypothesis: the word is being perceived and understood by the brain gradually after its presentation, and therefore the brain representation of the word becomes gradually similar to the neural network representation of the word (i.e. the embedding vector). The output probability feature accuracy peaks at a later time than the embeddings accuracy. Obtaining a higher than chance accuracy at time window m using the output probability as input to the classifier suggests strongly that the brain is integrating the word at time window m, because it is responding differently for predictable and unpredictable words[5]. The integration step happens after the perception step, which is probably why the output probability curves peak later than the embeddings curves.

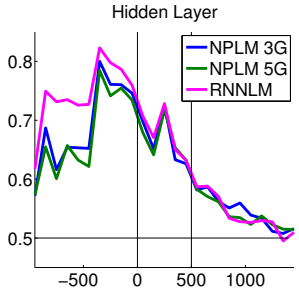


Figure 7: Average accuracy in time for the different hidden layers. The analysis is extended to the time windows before and after the word is presented, the input feature is restricted to be the hidden layer before the central word is seen. The first vertical bar indicates the onset of the word, the second one indicates the end of its presentation.

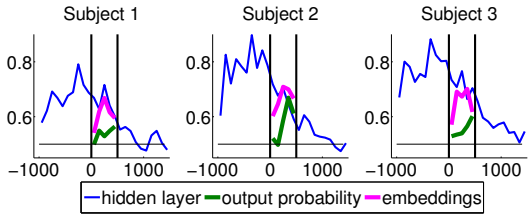


Figure 8: Accuracy in time when using the RNNLM features for each of the three subjects.

To understand the time dynamics of the hidden layer accuracy we need to see a larger time scale than the word itself. The hidden layer captures the context before word i is seen. Therefore it seems reasonable that the hidden layer is not only related to the activity when the word is on the screen, but also related to the activity before the word is presented, which is the time when the brain is integrating the previous words to build that context. On the other hand, as the word i and subsequent words are integrated, the context starts diverging from the context of word i (computed before seeing word i). We therefore ran the same analysis as before, but this time we also included the time windows before and after word i in the analysis, while maintaining the hidden layer vector to be the context before word i is seen. We see the behavior we predicted in the results: the context before seeing word i becomes gradually more useful for classification until word i is seen, and then it gradually decreases until it is no longer useful since the context has changed. We observe the RNNLM hidden layer has a higher classification accuracy than the finite context NPLMs. This might be due to the fact that the RNNLM has a more complete representation of context that captures more of the properties of the previous words.

To show the consistency of the results, we plot as illustration the three curves we obtain for each subject for the RNNLM (Fig. 8). The patterns seem very consistent indicating the phenomena we described can be detected at the subject level.

We now move on to the spatial decomposition of the analysis. When the visual input enters the brain, it first reaches the visual cortex at the back of the head, and then moves anteriorly towards the left and right temporal cortices and eventually the frontal cortex. As it flows through these areas, it is processed to higher levels of interpretations. In Fig. 9, we plot the accuracy for different regions of the brain and different time windows for the RNNLM features. To make the plots simpler we multiplied by zero the accuracies which were not significantly higher than chance. We expand a few characteristic plots. We see that in the back of the head the embedding features have an accuracy that seems to peak very early on. As we move forward in the brain towards the left and right temporal cortices, we see the embeddings accuracy peaking at a later time, reflecting the delay it takes for the information to reach this part of the brain. The output probability start being useful for classification after the embeddings, and specifically in the left temporal cortex which is the cite where the N400

[5]the fact that we can classify accurately during windows 300-400ms indicates that the classifier is taking advantage of the N400 response discussed in the introduction

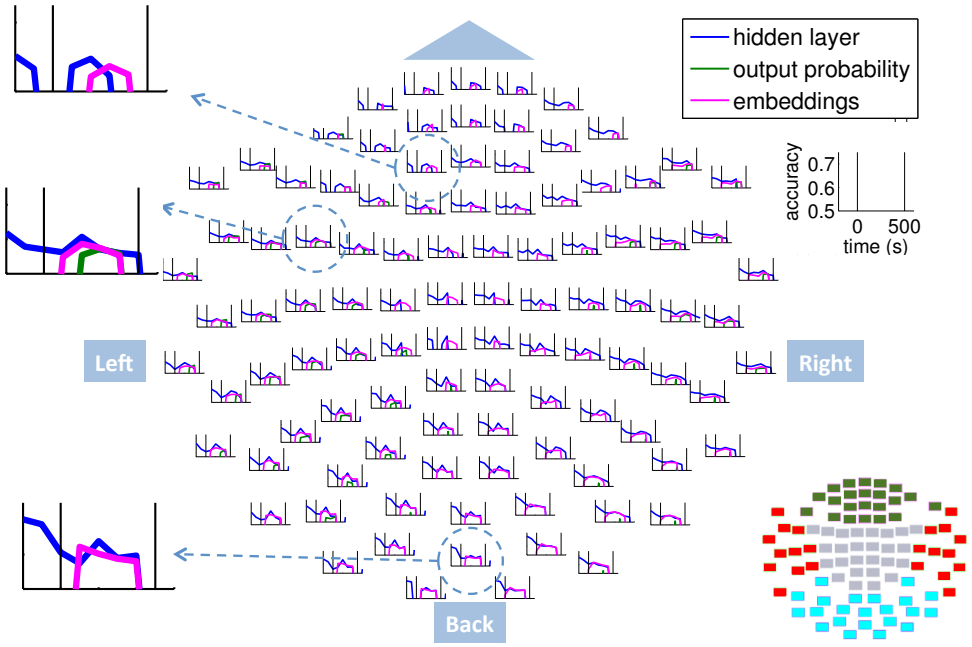


Figure 9: Average accuracy in time and space on the MEG helmet when using the RNNLM features. For each of the 102 locations the average accuracy for the group of sensors centered at that location is plotted versus time. The axes are defined in the rightmost, empty plot. Three plots have been magnified to show the increasing delay in high accuracy when using the embeddings feature, reflecting the delay in processing the incoming word as information travels through the brain. A sensor map is provided in the lower right corner: visual cortex = cyan, temporal = red, frontal = dark green.

is reported in the literature. Finally, as we reach the frontal cortex, we see that the embeddings features have an even later accuracy peak.

4 Conclusion and contributions

Novel brain data exploration We present here a novel and revealing approach to shed light on the brain processes involved in reading. This is a departure from the classical approach of controlling for a few variables in the text (e.g. showing a sentence with an expected target word versus an unexpected one). While we cannot make clear cut causal claims because we did not control for our variables, we are able to explore the data much more and offer a much richer interpretation than is possible with artificially constrained stimuli.

Comparing two models of language Adding brain data into the equation allowed us to compare the performance of the models and to identify a slight advantage for the RNNLM in capturing the text contents. Numerical comparison is however a secondary contribution of our approach. We showed that it might be possible to use brain data to understand, interpret and illustrate what exactly is being encoded by the obscure vectors that neural networks compute, by drawing parallels between the models constituents and brain processes.

Anecdotally, in the process of running the experiments, we noticed that the accuracy for the hidden layer of the RNNLM was peaking in the time window corresponding to word $i-2$, and that it was decreasing during word $i-1$. Since this was against our expectations, we went back and looked at the code and found that it was indeed returning a delayed value and corrected the features. We therefore used the brain data in order to correct a mis-specification in our neural network model. This hints if not proves the potential of our approach for assessing language models.

Future Work The work described here is our first attempt along the promising endeavor of matching complex computational models of language with brain processes using brain recordings. We plan to extend our efforts by (1) collecting data from more subjects and using various types of text and (2) make the brain data help us with training better statistical language models by using it to determine whether the models are expressive enough or have reached a sufficient degree of convergence.

Acknowledgements

This research was supported in part by NICHD grant 5R01HD07328-02. We thank Nicole Rafidi for help with data acquisition.

References

Yoav Benjamini and Daniel Yekutieli. 2001. The control of the false discovery rate in multiple testing under dependency. *Annals of statistics*, pages 1165–1188.

Augusto Buchweitz, Robert A Mason, Lêda Tomitch, and Marcel Adam Just. 2009. Brain activation for reading and listening comprehension: An fMRI study of modality effects and individual differences in language comprehension. *Psychology & neuroscience*, 2(2):111–123.

Kacper Chwialkowski and Arthur Gretton. 2014. A kernel independence test for random processes. *arXiv preprint arXiv:1402.4501*.

Ronald Aylmer Fisher. 1925. *Statistical methods for research workers*. Genesis Publishing Pvt Ltd.

Stefan L Frank, Leun J Otten, Giulia Galli, and Gabriella Vigliocco. 2013. Word surprisal predicts N400 amplitude during reading. In *Proceedings of the 51st annual meeting of the Association for Computational Linguistics*, pages 878–883.

Angela D Friederici. 2002. Towards a neural basis of auditory sentence processing. *Trends in cognitive sciences*, 6(2):78–84.

Gene H Golub, Michael Heath, and Grace Wahba. 1979. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223.

Peter Hagoort. 2003. How the brain solves the binding problem for language: a neurocomputational model of syntactic processing. *Neuroimage*, 20:S18–S29.

Nikolaus Kriegeskorte, Rainer Goebel, and Peter Bandettini. 2006. Information-based functional brain mapping. *Proceedings of the National Academy of Sciences of the United States of America*, 103(10):3863–3868.

Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and J Cernocky. 2011. RNNLM-recurrent neural network language modeling toolkit. In *Proc. of the 2011 ASRU Workshop*, pages 196–201.

Tomas Mikolov. 2012. *Statistical Language Models Based on Neural Networks*. Ph.D. thesis, Brno University of Technology.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of ICML*, pages 807–814.

Joanne K. Rowling. 2012. *Harry Potter and the Sorcerer's Stone*. Harry Potter US. Pottermore Limited.

Riitta Salmelin. 2007. Clinical neurophysiology of language: the MEG approach. *Clinical Neurophysiology*, 118(2):237–254.

Gustavo Sudre, Dean Pomerleau, Mark Palatucci, Leila Wehbe, Alona Fyshe, Riitta Salmelin, and Tom Mitchell. 2012. Tracking neural coding of perceptual and semantic features of concrete nouns. *NeuroImage*, 62(1):451–463.

Samu Taulu and Juha Simola. 2006. Spatiotemporal signal space separation method for rejecting nearby interference in MEG measurements. *Physics in medicine and biology*, 51(7):1759.

Samu Taulu, Matti Kajola, and Juha Simola. 2004. Suppression of interference and artifacts by the signal space separation method. *Brain topography*, 16(4):269–275.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation.

Leila Wehbe, Brian Murphy, Partha Talukdar, Alona Fyshe, Aaditya Ramdas, and Tom Mitchell. 2014. Simultaneously uncovering the patterns of brain regions involved in different story reading subprocesses. *in press*.

A Cognitive Model of Semantic Network Learning

Aida Nematzadeh, Afsaneh Fazly, and Suzanne Stevenson
Department of Computer Science
University of Toronto
{aida,afsaneh,suzanne}@cs.toronto.edu

Abstract

Child semantic development includes learning the meaning of words as well as the semantic relations among words. A presumed outcome of semantic development is the formation of a semantic network that reflects this knowledge. We present an algorithm for simultaneously learning word meanings and gradually growing a semantic network, which adheres to the cognitive plausibility requirements of incrementality and limited computations. We demonstrate that the semantic connections among words in addition to their context is necessary in forming a semantic network that resembles an adult's semantic knowledge.

1 Introduction

Child semantic development includes the acquisition of word-to-concept mappings (part of word learning), and the formation of semantic connections among words/concepts. There is considerable evidence that understanding the semantic properties of words improves child vocabulary acquisition. In particular, children are sensitive to commonalities of semantic categories, and this abstract knowledge facilitates subsequent word learning (Jones et al., 1991; Colunga and Smith, 2005). Furthermore, representation of semantic knowledge is significant as it impacts how word meanings are stored in, searched for, and retrieved from memory (Steyvers and Tenenbaum, 2005; Griffiths et al., 2007).

Semantic knowledge is often represented as a graph (a *semantic network*) in which nodes correspond to words/concepts[1], and edges specify the semantic relations (Collins and Loftus, 1975; Steyvers and Tenenbaum, 2005). Steyvers and Tenenbaum (2005) demonstrated that a semantic network that encodes adult-level knowledge of words exhibits a *small-world* and *scale-free* structure. That is, it is an overall sparse network with highly-connected local sub-networks, where these sub-networks are connected through high-degree hubs (nodes with many neighbours).

Much experimental research has investigated the underlying mechanisms of vocabulary learning and characteristics of semantic knowledge (Quine, 1960; Bloom, 1973; Carey and Bartlett, 1978; Gleitman, 1990; Samuelson and Smith, 1999; Jones et al., 1991; Jones and Smith, 2005). However, existing computational models focus on certain aspects of semantic acquisition: Some researchers develop computational models of word learning without considering the acquisition of semantic connections that hold among words, or how this semantic knowledge is structured (Siskind, 1996; Regier, 2005; Yu and Ballard, 2007; Frank et al., 2009; Fazly et al., 2010). Another line of work is to model formation of semantic categories but this work does not take into account how word meanings/concepts are acquired (Anderson and Matessa, 1992; Griffiths et al., 2007; Fountain and Lapata, 2011).

Our goal in this work is to provide a cognitively-plausible and unified account for both acquiring and representing semantic knowledge. The requirements for cognitive plausibility enforce some constraints on a model to ensure that it is comparable with the cognitive process it is formulating (Poibeau et al., 2013). As we model semantic acquisition, the first requirement is incrementality, which means that the model learns gradually as it processes the input. Also, there is a limit on the number of computations the model performs at each step.

In this paper, we present an algorithm for si-

[1]Here we assume that the nodes of a semantic network are word forms and its edges are determined by the semantic features of those words.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 244–254,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

multaneously learning word meanings and growing a semantic network, which adheres to the cognitive plausibility requirements of incrementality and limited computations. We examine networks created by our model under various conditions, and explore what is required to obtain a structure that has appropriate semantic connections and has a small-world and scale-free structure.

2 Related Work

Models of Word Learning. Given a word learning scenario, there are potentially many possible mappings between words in a sentence and their meanings (real-world referents), from which only some mappings are correct (the *mapping problem*). One of the most dominant mechanisms proposed for vocabulary acquisition is *cross-situational learning*: people learn word meanings by recognizing and tracking statistical regularities among the contexts of a word's usage across various situations, enabling them to narrow in on the meaning of a word that holds across its usages (Siskind, 1996; Yu and Smith, 2007; Smith and Yu, 2008). A number of computational models attempt to solve the mapping problem by implementing this mechanism, and have successfully replicated different patterns observed in child word learning (Siskind, 1996; Yu and Ballard, 2007; Fazly et al., 2010). These models have provided insight about underlying mechanisms of word learning, but none of them consider the semantic relations that hold among words, or how the semantic knowledge is structured. Recently, we have investigated properties of the semantic structure of the resulting (final) acquired knowledge of such a learner (Nematzadeh et al., 2014). However, that work did not address how such structural knowledge might develop and evolve incrementally within the learning model.

Models of Categorization. Computational models of categorization focus on the problem of forming semantic clusters given a defined set of features for words (Anderson and Matessa, 1992; Griffiths et al., 2007; Sanborn et al., 2010). Anderson and Matessa (1992) note that a cognitively plausible categorization algorithm needs to be incremental and only keep track of one potential partitioning; they propose a Bayesian framework (the Rational Model of Categorization or RMC) that specifies the joint distribution on features and category labels, and allows an unbounded number of clusters. Sanborn et al. (2010) examine different categorization models based on RMC. In particular, they compare the performance of the approximation algorithm of Anderson and Matessa (1992) (local MAP) with two other approximation algorithms (Gibbs Sampling and Particle Filters) in various human categorization paradigms. Sanborn et al. (2010) find that in most of the simulations the local MAP algorithm performs as well as the two other algorithms in matching human behavior.

The Representation of Semantic Knowledge. There is limited work on computational models of semantic acquisition that examine the representation of the semantic knowledge. Steyvers and Tenenbaum (2005) propose an algorithm for building a network with small-world and scale-free structure. The algorithm starts with a small complete graph, incrementally adds new nodes to the graph, and for each new node uses a probabilistic mechanism for selecting a subset of current nodes to connect to. However, their approach does not address the problem of learning word meanings or the semantic connections among them. Fountain and Lapata (2011) propose an algorithm for learning categories that also creates a semantic network by comparing all the possible word pairs. However, they too do not address the word learning problem, and do not investigate the structure of the learned semantic network to see whether it has the properties observed in adult knowledge.

3 The Incremental Network Model

We propose here a model that unifies the incremental acquisition of word meanings and formation of a semantic network structure that reflects the similarities among those meanings. We use an existing model to learn the meanings of words (Section 3.1), and use those incrementally developing meanings as the input to the algorithm proposed here for gradually growing a semantic network (Section 3.2).

3.1 The Word Learner

We use the model of Fazly et al. (2010); this learning algorithm is incremental and involves limited calculations, thus satisfying basic cognitive plausibility requirements. A naturalistic language learning scenario consists of linguistic data in the context of non-linguistic data, such as the objects,

Utterance: {*let*, *find*, *a*, *picture*, *to*, *color* }
Scene: {LET, PRONOUN, HAS_POSSESSION, CAUSE, ARTIFACT, WHOLE, CHANGE, ... }

Table 1: A sample utterance-scene pair.

events, and social interactions that a child perceives. This kind of input is modeled here as a pair of an *utterance* (the words a child hears) and a *scene* (the semantic features representing the meaning of those words), as shown in Table 1 (and described in more detail in Section 5.1). The word learner is an instance of cross-situational learning applied to a sequence of such input pairs: for each pair of a word w and a semantic feature f, the model incrementally learns $P(f|w)$ from co-occurrences of w and f across all the utterance-scene pairs.

For each word, the probability distribution over all semantic features, $P(.|w)$, represents the word's *meaning*. The estimation of $P(.|w)$ is made possible by introducing a set of latent variables, *alignments*, that correspond to the possible mappings between words and features in a given utterance–scene pair. The learning problem is then to find the mappings that best explain the data, which is solved by using an incremental version of the expectation–maximization (EM) algorithm (Neal and Hinton, 1998). We skip the details of the derivations and only report the resulting formulas.

The model processes one utterance-scene pair at a time. For the input pair processed at time t, first the probability of each possible alignment (alignment probability) is calculated as:[2]

$$P(a_{ij}|u, f_i) = \frac{P_{t-1}(f_i|w_j)}{\sum_{w' \in u} P_{t-1}(f_i|w')} \quad (1)$$

where u is the utterance, and a_{ij} is the alignment variable specifying the word w_j that is mapped to the feature f_i. $P_{t-1}(f_i|w_j)$ is taken from the model's current learned meaning of word w_j. Initially, $P_0(f_i|w_j)$ is uniformly distributed. After calculating the alignment probabilities, the learned meanings are updated as:

$$P_t(f_i|w_j) = \frac{\sum_{u \in U_t} P(a_{ij}|u, f_i)}{\sum_{f' \in \mathcal{M}} \sum_{u \in U_t} P(a_{ij}|u, f')} \quad (2)$$

where U_t is the set of utterances processed so far, and $\mathcal{M}$ is the set of features that the model has observed. Note that for each w–f pair, the value of the summations in this formula can be incrementally updated after processing any utterance that contains w; the summation does not have to be calculated at every step.

[2]This corresponds to the expectation step of EM.

3.2 Growing a Semantic Network

In our extended model, as we learn words incrementally (as above), we also structure those words into a semantic network based on the (partially) learned meanings. At any given point in time, the network will include as its nodes all the word types the word learner has been exposed to. Weighted edges (capturing semantic distance) will connect those pairs of word types whose learned meanings at that point are sufficiently semantically similar (according to a threshold). Since the probabilistic meaning of a word is adjusted each time it is observed, a word may either lose or gain connections in the network after each input is processed. Thus, to incrementally develop the network, at each time step, our algorithm must both examine existing connections (to see which edges should be removed) and consider potential new connections (to see which edges should be added).

A simple approach to achieve this is to examine the current semantic similarity between a word w in the input and all the current words in the network, and include edges between only those word pairs that are sufficiently similar. However, comparing w to all the words in the network each time it is observed is computationally intensive (and not cognitively plausible).

We present an approach for incrementally growing a semantic network that limits the computations when processing each input word w; see Algorithm 1. After the meaning of w is updated, we first check all the words that w is currently (directly) connected to, to see if any of those edges need to be removed, or have their weight adjusted. Next, to look for new connections for w, the idea is to select only a small subset of words, $\mathcal{S}$, to which w will be compared. The challenge then is to select $\mathcal{S}$ in a way that will yield a network whose semantic structure reasonably approximates the network that would result from full knowledge of comparing w to all the words.

Previous work has suggested picking "important" words (e.g., high-degree words) independently of the target word w — assuming these might be words for which a learner might need to understand their relationship to w in the future (Steyvers and Tenenbaum, 2005). Our proposal is instead to consider for $\mathcal{S}$ those words that are

Algorithm 1 Growing a network after each input u.

```
for all w in u do
  update P(.|w) using Eqn. (2)
  update current connections of w
  select S(w), a subset of words in the network
  for all w' in S(w) do
    if w and w' are sufficiently similar then
      connect w and w' with an edge
    end if
  end for
end for
```

likely to be similar to w. That is, since the network only needs to connect similar words to w, if we can guess what (some of) those words are, then we will do best at approximating the situation of comparing w to all words.

The question now is how to find semantically similar words to w that are not already connected to w in the network. To do so, we incrementally track semantic similarity among words usages as their meanings are developing. Specifically we cluster word tokens (not types) according to their current word meanings. Since the probabilistic meanings of words are continually evolving, incremental clusters of word tokens can capture developing similarities among the various usages of a word type, and be a clue to which words (types) w might be similar to. In the next section, we describe the Bayesian clustering process we use to identify potentially similar words.

3.3 Semantic Clustering of Word Tokens

We use the Bayesian framework of Anderson and Matessa (1992) to form semantic clusters.[3] Recall that for each word w, the model learns its meanings as a probability distribution over all semantic features, $P(.|w)$. We represent this probability distribution as a vector F whose length is the number of possible semantic features. Each element of the vector holds the value $P(f|w)$ (which is continuous). Given a word w and its vector F, we need to calculate the probability that w belongs to each existing cluster, and also allow for the possibility of it forming a new cluster. Using Bayes rule we have:

$$P(k|F) = \frac{P(k)P(F|k)}{\sum_{k'} P(k')P(F|k')} \quad (3)$$

[3]The distribution specified by this model is equivalent to that of a Dirichlet Process Mixture Model (Neal, 2000).

where k is a given cluster. We thus need to calculate the prior probability, $P(k)$, and the likelihood of each cluster, $P(F|k)$.

Calculation of Prior. The prior probability that word $n+1$ is assigned to cluster k is calculated as:

$$P(k) = \begin{cases} \frac{n_k}{n+\alpha} & n_k > 0 \\ \frac{\alpha}{n+\alpha} & n_k = 0 \text{ (new cluster)} \end{cases} \quad (4)$$

where n_k is the number of words in cluster k, n is the number of words observed so far, and α is a parameter that determines how likely the creation of a new cluster is. The prior favors larger clusters, and also discourages the creation of new clusters in later stages of learning.

Calculation of Likelihood. To calculate the likelihood $P(F|k)$ in Eqn. (3), we assume that the features are independent:

$$P(F|k) = \prod_{f_i \in F} P(f_i = v|k) \quad (5)$$

where $P(f_i = v|k)$ is the probability that the value of the feature in dimension i is equal to v given the cluster k. To derive $P(f_i|k)$, following Anderson and Matessa (1992), we assume that each feature given a cluster follows a Gaussian distribution with an unknown variance σ^2 and mean μ. (In the absence of any prior information about a variable, it is often assumed to have a Gaussian distribution.) The mean and variance of this distribution are inferred using Bayesian analysis: We assume the variance has an inverse χ^2 prior, where σ_0^2 is the prior variance and a_0 is the confidence in the prior variance:

$$\sigma^2 \sim \text{Inv-}\chi^2(a_0, \sigma_0^2) \quad (6)$$

The mean given the variance has a Gaussian distribution with μ_0 as the prior mean and λ_0 as the confidence in the prior mean.

$$\mu|\sigma \sim \text{N}(\mu_0, \frac{\sigma^2}{\lambda_0}) \quad (7)$$

Given the above conjugate priors, $P(f_i|k)$ can be calculated analytically and is a Student's t distribution with the following parameters:

$$P(f_i|k) \sim t_{a_i}(\mu_i, \sigma_i^2(1 + \frac{1}{\lambda_i})) \quad (8)$$

$$\lambda_i = \lambda_0 + n_k \quad (9)$$

$$a_i = a_0 + n_k \quad (10)$$

$$\mu_i = \frac{\lambda_0 \mu_0 + n_k \bar{f}}{\lambda_0 + n_k} \quad (11)$$

$$\sigma_i^2 = \frac{a_0 \sigma_0^2 + (n_k - 1)s^2 + \frac{\lambda_0 n_k}{\lambda_0 + n_k}(\mu_0 + \bar{f})^2}{a_0 + n_k} \quad (12)$$

where $\bar{f}$ and s^2 are the sample mean and variance of the values of f_i in k.

Note that in the above equations, the mean and variance of the distribution are simply derived by combining the sample mean and variance with the prior mean and variance while considering the confidence in the prior mean (λ_0) and variance (a_0). This means that the number of computations to calculate $P(F|K)$ is limited as w is only compared to the "prototype" of each cluster, which is represented by μ_i and σ_i of different features.

Adding a word w to a cluster. We add w to the cluster k with highest posterior probability, $P(k|F)$, as calculated in Eqn. (3).[4] The parameters of the selected cluster (k, μ_i, λ_i, σ_i, and a_i for each feature f_i) are then updated incrementally.

Using the Clusters to Select the Words in $\mathcal{S}(w)$. We can now form $S(w)$ in Algorithm 1 by selecting a given number of words n_s whose tokens are probabilistically chosen from the clusters according to how likely each cluster k is given w: the number of word tokens picked from each k is proportional to $P(k|F)$ and is equal to $P(k|F) \times n_s$.

4 Evaluation

We evaluate a semantic network in two regards: The semantic connectivity of the network – to what extent the semantically-related words are connected in the network; and the structure of the network – whether it exhibits a *small-world* and *scale-free* structure or not.

4.1 Evaluating Semantic Connectivity

The distance between the words in the network indicates their semantic similarity: the more similar a word pair, the smaller their distance. For word pairs that are connected via a path in the network, this distance is the weighted shortest path length between the two words. If there is no path between a word pair, their distance is considered to be ∞ (which is represented with a large number). We refer to this distance as the "learned" semantic similarity.

[4]This approach is referred to as local MAP (Sanborn et al., 2010); because of the incremental nature of the algorithm, it maximizes the current posterior distribution as opposed to the "global" posterior.

To evaluate the semantic connectivity of the learned network, we compare these learned similarity scores to "gold-standard" similarity scores that are calculated using the WordNet similarity measure of Wu and Palmer (1994) (also known as the WUP measure). We choose this measure since it captures the same type of similarity as in our model: words are considered similar if they belong to the same semantic category. Moreover, this measure does not incorporate information about other types of similarities, for example, words are not considered similar if they occur in similar contexts. Thus, the scores calculated with this measure are comparable with those of our learned network.

Given the gold-standard similarity scores for each word pair, we evaluate the semantic connectivity of the network based on two performance measures: coefficient of correlation and the median rank of the first five gold-standard associates. Correlation is a standard way to compare two lists of similarity scores (Budanitsky and Hirst, 2006). We create two lists, one containing the gold-standard similarity scores for all word pairs, and the other containing their corresponding learned similarity scores. We calculate the Spearman's rank correlation coefficient, ρ, between these two lists of similarity scores. Note that the learned similarity scores reflect the semantic distance among words whereas the WordNet scores reflect semantic closeness. Thus, a negative correlation is best in our evaluation, where the value of -1 corresponds to the maximum correlation.

Following Griffiths et al. (2007), we also calculate the median learned rank of the first five gold-standard associates for all words: For each word w, we first create a "gold-standard" associates list: we sort all other words based on their gold-standard similarity to w, and pick the five most similar words (associates) to w. Similarly, we create a "learned associate list" for w by sorting all words based on their learned semantic similarity to w. For all words, we find the ranks of their first five gold-standard associates in their learned associate list. For each associate, we calculate the median of these ranks for all words. We only report the results for the first three gold-standard associates since the pattern of results is similar for the fourth and fifth associates; we refer to the median rank of first three gold-standard associates as

1^{st}, 2^{nd}, and 3^{rd}.

4.2 Evaluating the Structure of the Network

A network exhibits a small-world structure when it is characterized by short path length between most nodes and highly-connected neighborhoods (Watts and Strogatz, 1998). We first explain how these properties are measured for a graph with N nodes and E edges. Then we discuss how these properties are used in assessing the small-world structure of a graph.[5].

Short path lengths. Most of the nodes of a small-world network are reachable from other nodes via relatively short paths. For a connected network (*i.e.*, all the node pairs are reachable from each other), this can be measured as the average distance between all node pairs (Watts and Strogatz, 1998). Since our networks are not connected, we instead measure this property using the median of the distances (d_{median}) between all node pairs (Robins et al., 2005), which is well-defined even when some node pairs have a distance of ∞.

Highly-connected neighborhoods. The neighborhood of a node n in a graph consists of n and all of the nodes that are connected to it. A neighborhood is maximally connected if it forms a complete graph —*i.e.*, there is an edge between all node pairs. Thus, the maximum number of edges in the neighborhood of n is $k_n(k_n-1)/2$, where k_n is the number of neighbors. A standard metric for measuring the connectedness of neighbors of a node n is called the *local clustering coefficient* (C) (Watts and Strogatz, 1998), which calculates the ratio of edges in the neighborhood of n (E_n) to the maximum number of edges possible for that neighborhood:

$$C = \frac{E_n}{k_n(k_n-1)/2} \tag{13}$$

The *local clustering coefficient* C ranges between 0 and 1. To estimate the connectedness of all neighborhoods in a network, we take the average of C over all nodes, *i.e.*, C_{avg}.

Small-world structure. A graph exhibits a small-world structure if d_{median} is relatively small and C_{avg} is relatively high. To assess this for a graph g, these values are typically compared to those of a random graph with the same number of nodes and edges as g (Watts and Strogatz, 1998; Humphries and Gurney, 2008). The random graph is generated by randomly rearranging the edges of the network under consideration (Erdos and Rényi, 1960). Because any pair of nodes is equally likely to be connected as any other, the median of distances between nodes is expected to be low for a random graph. In a small-world network, this value d_{median} is expected to be as small as that of a random graph: even though the random graph has edges more uniformly distributed, the small-world network has many locally-connected components which are connected via *hubs*. On the other hand, C_{avg} is expected to be much higher in a small-world network compared to its corresponding random graph, because the edges of a random graph typically do not fall into clusters forming highly connected neighborhoods.

Given these two properties, the "small-worldness" of a graph g is measured as follows (Humphries and Gurney, 2008):

$$\sigma_g = \frac{\frac{C_{avg}(g)}{C_{avg}(random)}}{\frac{d_{median}(g)}{d_{median}(random)}} \tag{14}$$

where $random$ is the random graph corresponding to g. In a small-world network, it is expected that $C_{avg}(g) \gg C_{avg}(random)$ and $d_{median}(g) \geq d_{median}(random)$, and thus $\sigma_g > 1$.

Note that Steyvers and Tenenbaum (2005) made the empirical observation that small-world networks of semantic knowledge had a single connected component that contained the majority of nodes in the network. Thus, in addition to σ_g, we also measure the relative size of a network's largest connected component having size N_{lcc}:

$$\text{size}_{\text{lcc}} = \frac{N_{lcc}}{N} \tag{15}$$

Scale-free structure. A scale-free network has a relatively small number of *high-degree* nodes that have a large number of connections to other nodes, while most of its nodes have a small degree, as they are only connected to a few nodes. Thus, if a network has a scale-free structure, its degree distribution (*i.e.*, the probability distribution of degrees over the whole network) will follow a power-law distribution (which is said to be "scale-free"). We evaluate this property of a network by plotting its degree distribution in the logarithmic scale, which (if a power-law distribution) should appear as a straight line. None of our networks ex-

[5]We take the description of these measures from Nematzadeh et al. (2014)

hibit a scale-free structure; thus, we do not report the results of this evaluation, and leave it to future work for further investigation.

5 Experimental Set-up

5.1 Input Representation

Recall that the input to the model consists of a sequence of utterance–scene pairs intended to reflect the linguistic data a child is exposed to, along with the associated meaning a child might grasp. As in much previous work (Yu and Ballard, 2007; Fazly et al., 2010), we take child-directed utterances from the CHILDES database (MacWhinney, 2000) in order to have naturalistic data. In particular, we use the Manchester corpus (Theakston et al., 2001), which consists of transcripts of conversations with 12 British children between the ages of $1;8$ and $3;0$. We represent each utterance as a bag of lemmatized words (see Utterance in Table 1).

For the scene representation, we have no large corpus to draw on that encodes the semantic portion of language acquisition data.[6] We thus automatically generate the semantics associated with an utterance, using a scheme first introduced in Fazly et al. (2010). The idea is to first create an input generation lexicon that provides a mapping between all the words in the input data and their associated *meanings*. A scene is then represented as a set that contains the meanings of all the words in the utterance. We use the input generation lexicon of Nematzadeh et al. (2012) because the word meanings reflect information about their semantic categories, which is crucial to forming the semantic clusters as in Section 3.3.

In this lexicon, the "true" meaning for each word w is a vector over a set of possible semantic features for each part of speech; in the vector, each feature is associated with a *score* for that word (see Figure 1). Depending on the word's part of speech, the features are extracted from various

[6] Yu and Ballard (2007) created a corpus by hand-coding the objects and cues that were present in the environment, but that corpus is very small. Frank et al. (2013) provide a larger manually annotated corpus (5000 utterances), but it is still very small for longitudinal simulations of word learning. (Our corpus contains more than 100,000 utterances.) Moreover, the corpus of Frank et al. (2013) is limited because a considerable number of words are not semantically coded. (Only a subset of concrete objects in the environment are coded.)

apple: { FOOD:1, SOLID:.72, $\cdots$, PLANT-PART:.22, PHYSICAL-ENTITY:.17, WHOLE:.06, $\cdots$ }

Figure 1: Sample true meaning features & their scores for *apple* from Nematzadeh et al. (2012).

lexical resources such as WordNet[7], VerbNet[8], and Harm (2002). The score for each feature is calculated using a measure similar to tf-idf that reflects the association of the feature with the word and with its semantic category: term frequency indicates the strength of association of the feature with the word, and inverse document frequency (where the documents are the categories) indicates how informative a feature is for that category. The semantic categories of nouns (which we focus on in our networks) are given by WordNet lex-names[9], a set of 25 general categories of entities. (We use only nouns in our semantic networks because the semantic similarity of words with different parts of speech cannot be compared, since their semantic features are drawn from different resources.)

The input generation lexicon is used to generate a scene representation for an utterance as follows: For each word w in the utterance, we probabilistically sample features, in proportion to their score, from the full set of features in its true meaning. The probabilistic sampling allows us to simulate the noise and uncertainty in the input a child perceives by omitting some meaning features from the scene. The scene representation is the union of all the features sampled for all the words in the utterance (see Scene in Table 1).

5.2 Methods

We experiment with our network-growth method that draws on the incremental clustering, and create "upper-bound" and baseline networks for comparison. Note that all the networks are created using our Algorithm 1 (page 4) to grow networks incrementally, drawing on the learned meanings of words and updating their connections on the basis of this evolving knowledge. The only difference in creating the networks resides in how the comparison set $\mathcal{S}(w)$ is chosen for each target word w that is being added to the growing network at each time step. We provide more details in the paragraphs below.

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

[8] http://verbs.colorado.edu/~mpalmer/projects/verbnet.html

[9] http://wordnet.princeton.edu/wordnet/man/lexnames.5WN.html

Upper-bound. Recall that one of our main goals is to substantially reduce the number of similarity comparisons needed to grow a semantic network, in contrast to the straightforward method of comparing each w to all current words. At the same time, we need to understand the impact of the increased efficiency on the quality of the resulting networks. We thus need to compare the target properties of our networks that are learned using a small comparison set $\mathcal{S}$, to those of an "upper-bound" network that takes into account all the pair-wise comparisons among words. We create this upper-bound network by setting $\mathcal{S}(w)$ to contain all words currently in the network.

Baselines. On the other hand, we need to evaluate the (potential) benefit of our cluster-driven selection process over a more simplistic approach to selecting $\mathcal{S}(w)$. To do so, we consider three baselines, each using a different criteria for choosing the comparison set $\mathcal{S}(w)$: The Random baseline chooses the members of this set randomly from the set of all observed words. The Context baseline can be seen as an "informed" baseline that attempts to incorporate some semantic knowledge: Here, we select words that are in the recent context prior to w in the input, assuming that such words are likely to be semantically related to w. We also include a third baseline, Random+Context, that picks half of the members of $\mathcal{S}$ randomly and half of them from the prior context.

Cluster-based Methods. We report results for three cluster-based networks that differ in their choice of $\mathcal{S}(w)$ as follows: The Clusters-only network chooses words in $\mathcal{S}(w)$ from the set of clusters, proportional to the probability of each cluster k given word w (as explained in Section 3.3). In order to incorporate different types of semantic information in selecting $\mathcal{S}$, we also create a Clusters+Context network that picks half of the members of $\mathcal{S}$ from clusters (as above), and half from the prior context. For completeness, we include a Clusters+Random network that similarly chooses half of words in $\mathcal{S}$ from clusters and half randomly from all observed words.

We have experimented with several other methods, but they all performed substantially worse than the baselines, and hence we do not report them here. E.g., we tried picking words in $\mathcal{S}$ from the best cluster. We also tried a few methods inspired by (Steyvers and Tenenbaum, 2005): E.g., we examined a method where if a member of $\mathcal{S}(w)$ was sufficiently similar to w, we added the direct neighbors of that word to $\mathcal{S}$. We also tried to grow networks by choosing the members of $\mathcal{S}$ according to the degree or frequency of nodes in the network.

5.3 Experimental Parameters

We use $20,000$ utterance–scene pairs as our training data. Recall that we use clustering to help guide our semantic network growth algorithm. Given the clustering algorithm in Section 3.3, we are interested to find the set of clusters that best explain the data. (Other clustering algorithms can be used instead of this algorithm.) We perform a search on the parameter space, and select the parameter values that result in the best clustering, based on the number of clusters and their average F-score. The value of the clustering parameters are as follows: $\alpha = 49$, $\lambda_0 = 1.0$, $a_0 = 2.0$, $\mu_0 = 0.0$, and $\sigma_0 = 0.05$. Two nouns with feature vectors F_1 and F_2 are connected in the network if $cosine(F1, F2)$ is greater than or equal to 0.6. (This threshold was selected following empirical examination of the similarity values we observe among the "true" meaning in our input generation lexicon.) The weight on the edge that connects these nouns specifies their semantic distance, which is calculated as $1 - cosine(F1, F2)$.

Because we aim for a network creation method that is cognitively plausible in performing a limited number of word-to-word comparisons, we need to ensure that all the different methods of selecting the comparison set $\mathcal{S}(w)$ yield roughly similar numbers of such comparisons. Keeping the size of $\mathcal{S}$ constant does not guarantee this, because each method can yield differing numbers of connections of the target word w to other words. We thus parameterize the size of $\mathcal{S}$ for each method to keep the number of computations similar, based on experiments on the development data. In development work we also found that having an increasing size of $\mathcal{S}$ over time improved the results, as more words were compared as the knowledge of learned meanings improved. To achieve this, we use a percentage of the words in the network as the size of $\mathcal{S}$. In practice, the setting of this parameter yields a number of comparisons across all methods that is about 8% of the maximum possible word-to-word comparisons that would be performed in the naive (computationally intensive) approach.

Note that all the Cluster-based, Random and Random+Context methods include a random selection mechanism; thus, we run each of these methods 50 times and report the average ρ, median ranks and size_{lcc} (see Section 4). For the networks (out of 50 runs) that exhibit a small-world structure (small-worldness greater than one), we report the average small-worldness. We also report the percentage of runs whose resulting network exhibit a small-world structure.

6 Experimental Results and Discussion

Table 2 presents our results, including the evaluation measures explained above, for the Upper-bound, Baseline, and Cluster-based networks created by the various methods described in Section 5.2.[10]

Recall that the Upper-bound network is formed from examining a word's similarity to all other (observed) words when it is added to the network. We can see that this network is highly connected (0.85) and has a small-world structure (5.5). There is a statistically significant correlation of the network's similarity measures with the gold standard ones (-0.38). For this Upper-bound structure, the median ranks of the first three associates are between 31 and 42. These latter two measures on the Upper-bound network give an indication of the difficulty of learning a semantic network whose knowledge matches gold-standard similarities.

Considering the baseline networks, we note that the Random network is actually somewhat better (in connectivity and median ranks) than the Context network that we thought would provide a more informed baseline. Interestingly, the correlation value for both networks is no worse than for the Upper-bound. The combination of Random+Context yields a slightly lower correlation, and no better ranks or connectivity than Random. Note that none of the baseline networks exhibit a small world structure ($\sigma_g \ll 1$ for all three, except for one out of 50 runs for the Random method).

Recall that the Random network is not a network resulting from randomly connecting word pairs, but one that incrementally compares each target word with a set of randomly chosen words when considering possible new connections. We suspect that this approach performs reasonably well because it enables the model to find a broad range of similar words to the target; this might be effective especially because the learned meanings of words are changing over time.

Turning to the Cluster-based methods, we see that indeed some diversity in the comparison set for a target word might be necessary to good performance. We find that the measures on the Clusters-only network are roughly the same as on the Random one, but when we combine the two in Clusters+Random we see an improvement in the ranks achieved. It is possible that the selection from clusters does not have sufficient diversity to find some of the valid new connections for a word.

We note that the best results overall occur with the Clusters+Context network, which combines two approaches to selecting words that have good potential to be similar to the target word. The correlation coefficient for this network is at a respectable 0.36, and the median ranks are the second best of all the network-growth methods. Importantly, this network shows the desired small-world structure in most of the runs (77%), with the highest connectivity and a small-world measure well over 1.

The fact that the Clusters+Context network is better overall than the networks of the Clusters-only and Context methods indicates that both clusters and context are important in making "informed guesses" about which words are likely to be similar to a target word. Given the small number of similarity comparisons used in our experiments (only around 8% of all possible word-to-word comparisons), these observations suggest that both the linguistic context and the evolving relations among word usages (captured by the incremental clustering of learned meanings) contain information crucial to the process of growing a semantic network in a cognitively plausible way.

7 Conclusions

We propose a unified model of word learning and semantic network formation, which creates a network of words in which connections reflect structured knowledge of semantic similarity between words. The model adheres to the cognitive plausibility requirements of incrementality and use of limited computations. That is, when incrementally adding or updating a word's connections in the network, the model only looks at a subset of words rather than comparing the target word to all the nodes in the network. We demonstrate that

[10] All the reported co-efficients of correlation (ρ) are statistically significant at $p < 0.01$.

Comparing all Pairs

Method	Semantic Connectivity				Small World	
	ρ	1^{st}	2^{nd}	3^{rd}	size_{lcc}	σ_g (%)
Upper-bound	-0.38	31	41	42	0.85	5.5
Baselines						
Random	-0.38	56	76.9	68.9	0.6	5.2 (2)
Context	$\mathbf{-0.39}$	97	115	89	0.5	0
Random+Context	-0.36	63.3	87.2	79.1	0.6	0 (0)
Cluster-based Methods						
Clusters-only	-0.32	58.6	72.0	71.6	**0.7**	5.5 (43)
Clusters+Context	-0.36	53.9	67.6	64.8	**0.7**	**7.2** (**77**)
Clusters+Random	-0.35	**48.1**	**61.2**	**58.1**	**0.7**	6.9 (48)

Table 2: Connectivity and small-worldness measures for the Upper-bound, Baseline, and Cluster-based network-growth methods; best performances across the Baseline and Cluster-based methods are shown in bold. ρ: co-efficient of correlation between similarities of word pairs in network and in gold-standard; 1^{st}, 2^{nd}, 3^{rd}: median ranks of corresponding gold-standard associates given network similarities; size_{lcc}: proportion of network in the largest connected component; σ_g: overall "small-worldness", should be greater than 1; %: the percentage of runs whose resulting networks exhibit a small-world structure. Note there are 1074 nouns in each network.

using the evolving knowledge of semantic connections among words as well as their context of usage enables the model to create a network that shows the properties of adult semantic knowledge. This suggests that the information in the semantic relations among words and their context can efficiently guide semantic network growth.

Acknowledgments

We would like to thank Varada Kolhatkar for valuable discussion and feedback. We are also grateful for the financial support from NSERC of Canada, and University of Toronto.

References

John R. Anderson and Michael Matessa. 1992. Explorations of an incremental, bayesian algorithm for categorization. *Machine Learning*, 9(4):275–308.

Lois Bloom. 1973. *One word at a time: The use of single word utterances before syntax*, volume 154. Mouton The Hague.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

Susan Carey and Elsa Bartlett. 1978. Acquiring a single new word.

Allan M. Collins and Elizabeth F. Loftus. 1975. A spreading-activation theory of semantic processing. *Psychological review*, 82(6):407.

Eliana Colunga and Linda B. Smith. 2005. From the lexicon to expectations about kinds: A role for associative learning. *Psychological Review*, 112(2):347–382.

Paul Erdos and Alfréd Rényi. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci*, 5:17–61.

Afsaneh Fazly, Afra Alishahi, and Suzanne Stevenson. 2010. A probabilistic computational model of cross-situational word learning. *Cognitive Science*, 34(6):1017–1063.

Trevor Fountain and Mirella Lapata. 2011. Incremental models of natural language category acquisition. In *Proceedings of the 32st Annual Conference of the Cognitive Science Society*.

Michael C. Frank, Noah D. Goodman, and Joshua B. Tenenbaum. 2009. Using speakers referential intentions to model early cross-situational word learning. *Psychological Science*.

Michael C. Frank, Joshua B. Tenenbaum, and Anne Fernald. 2013. Social and discourse contributions to the determination of reference in cross-situational word learning. *Language Learning and Development*, 9(1):1–24.

Lila Gleitman. 1990. The structural sources of verb meanings. *Language Acquisition*, 1(1):3–55.

Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological review*, 114(2):211.

Michael W. Harm. 2002. Building large scale distributed semantic feature sets with WordNet. Technical Report PDP.CNS.02.1, Carnegie Mellon University.

Mark D. Humphries and Kevin Gurney. 2008. Network small-world-ness: a quantitative method for determining canonical network equivalence. *PLoS One*, 3(4):e0002051.

Susan S. Jones and Linda B. Smith. 2005. Object name learning and object perception: a deficit in late talkers. *J. of Child Language*, 32:223–240.

Susan S. Jones, Linda B. Smith, and Barbara Landau. 1991. Object properties and knowledge in early lexical learning. *Child Development*, 62(3):499–516.

Brian MacWhinney. 2000. *The CHILDES Project: Tools for Analyzing Talk*, volume 2: The Database. Erlbaum, 3rd edition.

Radford M. Neal and Geoffrey E. Hinton. 1998. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer.

Radford M. Neal. 2000. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265.

Aida Nematzadeh, Afsaneh Fazly, and Suzanne Stevenson. 2012. Interaction of word learning and semantic category formation in late talking. In *Proc. of CogSci'12*.

Aida Nematzadeh, Afsaneh Fazly, and Suzanne Stevenson. 2014. Structural differences in the semantic networks of simulated word learners.

Thierry Poibeau, Aline Villavicencio, Anna Korhonen, and Afra Alishahi, 2013. *Computational Modeling as a Methodology for Studying Human Language Learning*. Springer.

Willard Van Orman Quine. 1960. *Word and Object*. MIT Press.

Terry Regier. 2005. The emergence of words: Attentional learning in form and meaning. *Cognitive Science*, 29:819–865.

Garry Robins, Philippa Pattison, and Jodie Woolcock. 2005. Small and other worlds: Global network structures from local processes1. *American Journal of Sociology*, 110(4):894–936.

Larissa K. Samuelson and Linda B. Smith. 1999. Early noun vocabularies: do ontology, category structure and syntax correspond? *Cognition*, 73(1):1 – 33.

Adam N. Sanborn, Thomas L. Griffiths, and Daniel J. Navarro. 2010. Rational approximations to rational models: alternative algorithms for category learning.

Jeffery Mark Siskind. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61:39–91.

Linda B. Smith and Chen Yu. 2008. Infants rapidly learn word-referent mappings via cross-situational statistics. *Cognition*, 106(3):1558–1568.

Mark Steyvers and Joshua B. Tenenbaum. 2005. The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive science*, 29(1):41–78.

Anna L. Theakston, Elena V. Lieven, Julian M. Pine, and Caroline F. Rowland. 2001. The role of performance limitations in the acquisition of verb–argument structure: An alternative account. *J. of Child Language*, 28:127–152.

Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440–442.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics.

Chen Yu and Dana H. Ballard. 2007. A unified model of early word learning: Integrating statistical and social cues. *Neurocomputing*, 70(1315):2149 – 2165. Selected papers from the 3rd International Conference on Development and Learning (ICDL 2004), Time series prediction competition: the CATS benchmark.

Chen Yu and Linda B. Smith. 2007. Rapid word learning under uncertainty via cross-situational statistics. *Psychological Science*, 18(5):414–420.

Learning Abstract Concept Embeddings from Multi-Modal Data: Since You Probably Can't See What I Mean

Felix Hill
Computer Laboratory
University of Cambridge
felix.hill@cl.cam.ac.uk

Anna Korhonen
Computer Laboratory
University of Cambridge
anna.korhonen@cl.cam.ac.uk

Abstract

Models that acquire semantic representations from both linguistic and perceptual input are of interest to researchers in NLP because of the obvious parallels with human language learning. Performance advantages of the multi-modal approach over language-only models have been clearly established when models are required to learn concrete noun concepts. However, such concepts are comparatively rare in everyday language. In this work, we present a new means of extending the scope of multi-modal models to more commonly-occurring abstract lexical concepts via an approach that learns multi-modal embeddings. Our architecture outperforms previous approaches in combining input from distinct modalities, and propagates perceptual information on concrete concepts to abstract concepts more effectively than alternatives. We discuss the implications of our results both for optimizing the performance of multi-modal models and for theories of abstract conceptual representation.

1 Introduction

Multi-modal models that learn semantic representations from both language and information about the perceptible properties of concepts were originally motivated by parallels with human word learning (Andrews et al., 2009) and evidence that many concepts are grounded in perception (Barsalou and Wiemer-Hastings, 2005). The perceptual information in such models is generally mined directly from images (Feng and Lapata, 2010; Bruni et al., 2012) or from data collected in psychological studies (Silberer and Lapata, 2012; Roller and Schulte im Walde, 2013).

By exploiting the additional information encoded in perceptual input, multi-modal models can outperform language-only models on a range of semantic NLP tasks, including modelling similarity (Bruni et al., 2014; Kiela et al., 2014) and free association (Silberer and Lapata, 2012), predicting compositionality (Roller and Schulte im Walde, 2013) and concept categorization (Silberer and Lapata, 2014). However, to date, these previous approaches to multi-modal concept learning focus on concrete words such as *cat* or *dog*, rather than abstract concepts, such as *curiosity* or *loyalty*. However, differences between abstract and concrete processing and representation (Paivio, 1991; Hill et al., 2013; Kiela et al., 2014) suggest that conclusions about concrete concept learning may not necessarily hold in the general case. In this paper, we therefore focus on multi-modal models for learning both abstract and concrete concepts.

Although concrete concepts might seem more basic or fundamental, the vast majority of open-class, meaning-bearing words in everyday language are in fact abstract. 72% of the noun or verb tokens in the British National Corpus (Leech et al., 1994) are rated by human judges[1] as more abstract than the noun *war*, for instance, a concept many would already consider to be quite abstract. Moreover, abstract concepts by definition encode higher-level (more general) principles than concrete concepts, which typically reside naturally in a single semantic category or domain (Crutch and Warrington, 2005). It is therefore likely that abstract representations may prove highly applicable for multi-task, multi-domain or transfer learning models, which aim to acquire 'general-purpose' conceptual knowledge without reference to a specific objective or task (Collobert and Weston, 2008; Mesnil et al., 2012).

In a recent paper, Hill et al. (2014) investigate whether the multi-modal models cited above are

[1] Contributors to the USF dataset (Nelson et al., 2004).

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 255–265,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

effective for learning concepts other than concrete nouns. They observe that representations of certain abstract concepts can indeed be enhanced in multi-modal models by combining perceptual and linguistic input with an *information propagation* step. Hill et al. (2014) propose ridge regression as an alternative to the nearest-neighbour averaging proposed by Johns and Jones (2012) for such propagation, and show that it is more robust to changes in the type of concept to be learned. However, both methods are somewhat inelegant, in that they learn separate linguistic and 'pseudo-perceptual' representations, which must be combined via a separate *information combination* step. Moreover, for the majority of abstract concepts, the best performing multi-modal model employing these techniques remains less effective than conventional text-only representation learning model.

Motivated by these observations, we introduce an architecture for learning both abstract and concrete representations that generalizes the skipgram model of Mikolov et al. (2013) from text-based to multi-modal learning. Aspects of the model design are influenced by considering the process of human language learning. The model moderates the training input to include more perceptual information about commonly-occurring concrete concepts and less information about rarer concepts. Moreover, it integrates the processes of combining perceptual and linguistic input and propagating information from concrete to abstract concepts into a single representation update process based on back-propagation.

We train our model on running-text language and two sources of perceptual descriptors for concrete nouns: the ESPGame dataset of annotated images (Von Ahn and Dabbish, 2004) and the CSLB set of concept property norms (Devereux et al., 2013). We find that our model combines information from the different modalities more effectively than previous methods, resulting in an improved ability to model the USF free association gold standard (Nelson et al., 2004) for concrete nouns. In addition, the architecture propagates the extra-linguistic input for concrete nouns to improve representations of abstract concepts more effectively than alternative methods. While this propagation can effectively extend the advantage of the multi-modal approach to many more concepts than simple concrete nouns, we observe that the benefit of adding perceptual input appears to decrease as target concepts become more abstract. Indeed, for the most abstract concepts of all, language-only models still provide the most effective learning mechanism.

Finally, we investigate the optimum quantity and type of perceptual input for such models. Between the most concrete concepts, which can be effectively represented directly in the perceptual modality, and the most abstract concepts, which cannot, we identify a set of concepts that cannot be represented effectively directly in the perceptual modality, but still benefit from perceptual input propagated in the model via concrete concepts.

The motivation in designing our model and experiments is both practical and theoretical. Taken together, the empirical observations we present are potentially important for optimizing the learning of representations of concrete and abstract concepts in multi-modal models. In addition, they offer a degree of insight into the poorly understood issue of how abstract concepts may be encoded in human memory.

2 Model Design

Before describing how our multi-modal architecture encodes and integrates perceptual information, we first describe the underlying corpus-based representation learning model.

Language-only Model Our multi-modal architecture builds on the continuous log-linear skipgram language model proposed by Mikolov et al. (2013). This model learns lexical representations in a similar way to neural-probabilistic language models (NPLM) but without a non-linear hidden layer, a simplification that facilitates the efficient learning of large vocabularies of dense representations, generally referred to as *embeddings* (Turian et al., 2010). Embeddings learned by the model achieve state-of-the-art performance on several evaluations including sentence completion and analogy modelling (Mikolov et al., 2013).

For each word type w in the vocabulary V, the model learns both a 'target-embedding' $r_w \in \mathbb{R}^d$ and a 'context-embedding' $\hat{r}_w \in \mathbb{R}^d$ such that, given a target word, its ability to predict nearby context words is maximized. The probability of seeing context word c given target w is defined as:

$$p(c|w) = \frac{e^{\hat{r}_c \cdot r_w}}{\sum_{v \in V} e^{\hat{r}_v \cdot r_w}}$$

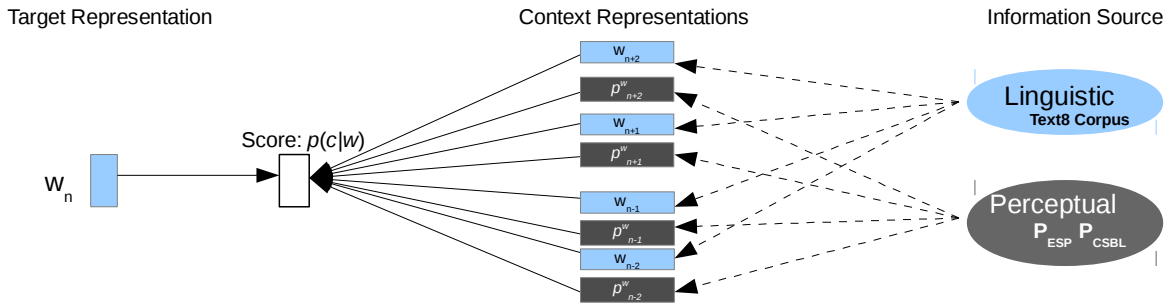


Figure 1: Our multi-modal model architecture. Light boxes are elements of the original Mikolov et al. (2013) model. For target words w_n in the domain of $\mathbf{P}$ (concrete concepts), the model updates its representations based on corpus context words $w_{n\pm i}$, then on words $p^w_{n\pm i}$ in perceptual pseudo-sentences. For w_n not in the domain of $\mathbf{P}$ (abstract concepts), updates are based solely on the $w_{n\pm i}$.

The model learns from a set of target-word, context-word pairs, extracted from a corpus of sentences as follows. In a given sentence S (of length N), for each position $n \leq N$, each word w_n is treated in turn as a target word. An integer $t(n)$ is then sampled from a uniform distribution on $\{1, \ldots k\}$, where $k > 0$ is a predefined maximum context-window parameter. The pair tokens $\{(w_n, w_{n+j}) : -t(n) \leq j \leq t(n), w_i \in S\}$ are then appended to the training data. Thus, target/context training pairs are such that (i) only words within a k-window of the target are selected as context words for that target, and (ii) words closer to the target are more likely to be selected than those further away.

The training objective is then to maximize the sum of the log probabilities T across of all such examples from S and across all sentences in the corpus, where T is defined as follows:

$$T = \frac{1}{N} \sum_{n=1}^{N} \sum_{-t(n) \leq j \leq t(n), j \neq 0} log(p(w_{n+j}|w_n))$$

The model free parameters (target-embeddings and context-embeddings of dimension d for each word in the corpus with frequency above a certain threshold f) are updated according to stochastic gradient descent and backpropation, with learning rate controlled by Adagrad (Duchi et al., 2011). For efficiency, the output layer is encoded as a hierarchical softmax function based on a binary Huffman tree (Morin and Bengio, 2005).

As with other distributional architectures, the model captures conceptual semantics by exploiting the fact that words appearing in similar linguistic contexts are likely to have similar meanings. Informally, the model adjusts its embeddings to increase the 'probability' of seeing the language in the training corpus. Since this probability increases with the $p(c|w)$, and the $p(c|w)$ increase with the dot product $\hat{r}_c \cdot r_w$, the updates have the effect of moving each target-embedding incrementally 'closer' to the context-embeddings of its collocates. In the target-embedding space, this results in embeddings of concept words that regularly occur in similar contexts moving closer together.

Multi-modal Extension We extend the Mikolov et al. (2013) architecture via a simple means of introducing perceptual information that aligns with human language learning. Based on the assumption that frequency in domain-general linguistic corpora correlates with the likelihood of 'experiencing' a concept in the world (Bybee and Hopper, 2001; Chater and Manning, 2006), perceptual information is introduced to the model whenever designated concrete concepts are encountered in the running-text linguistic input. This has the effect of introducing more perceptual input for commonly experienced concrete concepts and less input for rarer concrete concepts.

To implement this process, perceptual information is extracted from external sources and encoded in an associative array $\mathbf{P}$, which maps (typically concrete) words w to bags of perceptual features $\mathbf{b}(w)$. The construction of this array depends on the perceptual information source; the process for our chosen sources is detailed in Section 2.1.

Training our model begins as before on running-text. When a sentence S_m containing a word w in the domain of $\mathbf{P}$ is encountered, the model finishes training on S_m and begins learning from a perceptual pseudo-sentence $\hat{S_m}(w)$. $\hat{S_m}(w)$ is constructed by alternating the token w with a fea-

$\hat{S}(crocodile)$ = **Crocodile** legs **crocodile** teeth **crocodile** teeth **crocodile** scales **crocodile** green **crocodile**.

$\hat{S}(screwdriver)$ = **Screwdriver** handle **screwdriver** flat **screwdriver** long **screwdriver** handle **screwdriver** head.

Figure 2: Example pseudo-sentences generated by our model.

ture sampled at random from $\mathbf{b}(w)$ until $\hat{S_m}(w)$ is the same length as S_m (see Figure 2). Because we want the ensuing perceptual learning process to focus on how w relates to its perceptual properties (rather than how those properties relate to each other), we insert multiple instances of w into $\hat{S_m}(w)$. This ensures that the majority of training cases derived from $\hat{S_m}(w)$ are instances of (w, feature) rather than (feature, feature) pairs. Once training on $\hat{S_m}(w)$ is complete, the model reverts to the next 'genuine' (linguistic) sentence S_{m+1}, and the process continues. Thus, when a concrete concept is encountered in the corpus, its embedding is first updated based on language (moved incrementally closer to concepts appearing in similar linguistic contexts), and then on perception (moved incrementally closer to concepts with the same or similar perceptual features).

For greater flexibility, we introduce a parameter α reflecting the raw quantity of perceptual information relative to linguistic input. When $\alpha = 2$, two pseudo-sentences are generated and inserted for every corpus occurrence of a token from the domain of $\mathbf{P}$. For non-integral α, the number of sentences inserted is $\lfloor\alpha\rfloor$, and a further sentence is added with probability $\alpha - \lfloor\alpha\rfloor$.

In all experiments reported in the following sections we set the window size parameter $k = 5$ and the minimum frequency parameter $f = 3$, which guarantees that the model learns embeddings for all concepts in our evaluation sets. While the model learns both target and context-embeddings for each word in the vocabulary, we conduct our experiments with the target embeddings only. We set the dimension parameter $d = 300$ as this produces high quality embeddings in the language-only case (Mikolov et al., 2013).

2.1 Information Sources

We construct the associative array of perceptual information $\mathbf{P}$ from two sources typical of those used for multi-modal semantic models.

ESPGame Dataset The ESP-Game dataset (ESP) (Von Ahn and Dabbish, 2004) consists of 100,000 images, each annotated with a list of lexical concepts that appear in that image.

For any concept w identified in an ESP image, we construct a corresponding bag of features $\mathbf{b}(w)$. For each ESP image I that contains w, we append the other concept tokens identified in I to $\mathbf{b}(w)$. Thus, the more frequently a concept co-occurs with w in images, the more its corresponding lexical token occurs in $\mathbf{b}(w)$. The array $\mathbf{P_{ESP}}$ in this case then consists of the $(w, \mathbf{b}(w))$ pairs.

CSLB Property Norms The Centre for Speech, Language and the Brain norms (CSLB) (Devereux et al., 2013) is a recently-released dataset containing semantic properties for 638 concrete concepts produced by human annotators. The CSLB dataset was compiled in the same way as the McRae et al. (2005) property norms used widely in multi-modal models (Silberer and Lapata, 2012; Roller and Schulte im Walde, 2013); we use CSLB because it contains more concepts. For each concept, the proportion of the 30 annotators that produced a given feature can also be employed as a measure of the strength of that feature.

When encoding the CSLB data in $\mathbf{P}$, we first map properties to lexical forms (e.g. *is_green* becomes *green*). By directly identifying perceptual features and linguistic forms in this way, we treat features observed in the perceptual data as (sub)concepts to be acquired via the same multi-modal input streams and stored in the same domain-general memory as the evaluation concepts. This design decision in fact corresponds to a view of cognition that is sometimes disputed (Fodor, 1983). In future studies we hope to compare the present approach to architectures with domain-specific conceptual memories.

For each concept w in CSLB, we then construct a feature bag $\mathbf{b}(w)$ by appending lexical forms to $\mathbf{b}(w)$ such that the count of each feature word is equal to the strength of that feature for w. Thus, when features are sampled from $\mathbf{b}(w)$ to create pseudo-sentences (as detailed previously) the probability of a feature word occurring in a sentence reflects feature strength. The array $\mathbf{P_{CSLB}}$ then consists of all $(w, \mathbf{b}(w))$ pairs.

Linguistic Input The linguistic input to all models is the 400m word Text8 Corpus[2] of

[2]From http://mattmahoney.net/dc/textdata.html

ESPGame		CSLB	
Image 1	Image 2	Crocodile	Screwdriver
red	wreck	has 4 legs (7)	has handle (28)
chihuaua	cyan	has tail (18)	has head (5)
eyes	man	has jaw (7)	is long (9)
little	crash	has scales (8)	is plastic (18)
ear	accident	has teeth (20)	is metal (28)
nose	street	is green (10)	
small		is large (10)	

Table 1: Concepts identified in images in the ESP Game (left) and features produced for concepts by human annotators in the CSLB dataset (with feature strength, max=30).

Concept 1	Concept 2	Assoc.
abdomen (6.83)	stomach (6.04)	0.566
throw (4.05)	ball (6.08)	0.234
hope (1.18)	glory (3.53)	0.192
egg (5.79)	milk (6.66)	0.012

Table 2: Example concept pairs (with mean concreteness rating) and free-association scores from the USF dataset.

Wikipedia text, split into sentences and with punctuation removed.

2.2 Evaluation

We evaluate the quality of representations by how well they reflect *free association* scores, an empirical measure of cognitive conceptual proximity. The University of South Florida Norms (USF) (Nelson et al., 2004) contain free association scores for over 40,000 concept pairs, and have been widely used in NLP to evaluate semantic representations (Andrews et al., 2009; Feng and Lapata, 2010; Silberer and Lapata, 2012; Roller and Schulte im Walde, 2013). Each concept that we extract from the USF database has also been rated for conceptual concreteness on a Likert scale of 1-7 by at least 10 human annotators. Following previous studies (Huang et al., 2012; Silberer and Lapata, 2012), we measure the (Spearman ρ) correlation between association scores and the cosine similarity of vector representations.

We create separate abstract and concrete concept lists by ranking the USF concepts according to concreteness and sampling at random from the first and fourth quartiles respectively. We also introduce a complementary noun/verb dichotomy, on the intuition that information propagation may occur differently from noun to noun or from noun to verb (because of their distinct structural relationships in sentences). POS-tags are not assigned as part of the USF data, so we draw the noun/verb distinction based on the majority POS-tag of USF concepts in the lemmatized British National Corpus (Leech et al., 1994). The abstract/concrete and noun/verb dichotomies yield four distinct concept lists. For consistency, the concrete noun list is filtered so that each concrete noun concept w has a perceptual representation $\mathbf{b}(\mathrm{w})$ in both $\mathbf{P_{ESP}}$ and $\mathbf{P_{CSLB}}$. For the four resulting concept lists C (concrete/abstract, noun/verb), a corresponding set of evaluation pairs $\{(w_1, w_2) \in USF : w_1, w_2 \in C\}$ is extracted (see Table 3 for details).

Concept Type	List	Pairs	Examples
concrete nouns	541	1418	*yacht, cup*
abstract nouns	100	295	*fear, respect*
all nouns	666	1815	*fear, cup*
concrete verbs	50	66	*kiss, launch*
abstract verbs	50	127	*differ, obey*
all verbs	100	221	*kiss, obey*

Table 3: Details the subsets of USF data used in our evaluations, downloadable from our website.

3 Results and Discussion

Our experiments were designed to answer four questions, outlined in the following subsections: (1) Which model architectures perform best at *combining* information pertinent to multiple modalities when such information exists explicitly (as common for concrete concepts)? (2) Which model architectures best propagate perceptual information to concepts for which it does not exist explicitly (as is common for abstract concepts)? (3) Is it preferable to include all of the perceptual input that can be obtained from a given source, or to filter this input stream in some way? (4) How much perceptual vs. linguistic input is optimal for learning various concept types?

3.1 Combining information sources

To evaluate our approach as a method of information combination we compared its performance on the concrete noun evaluation set against three alternative methods. The first alternative is simple concatenation of these perceptual vectors with linguistic vectors embeddings learned

by the Mikolov et al. (2013) model on the Text8 Corpus. In the second alternative (proposed for multi-modal models by Silberer and Lapata (2012)), *canonical correlation analysis* (CCA) (Hardoon et al., 2004) was applied to the vectors of both modalities. CCA yields reduced-dimensionality representations that preserve underlying inter-modal correlations, which are then concatenated. The final alternative, proposed by Bruni et al. (2014) involves applying Singular Value Decomposition (SVD) to the matrix of concatenated multi-modal representations, yielding smoothed representations.[3]

When implementing the concatenation, CCA and SVD methods, we first encoded the perceptual input directly into sparse feature vectors, with coordinates for each of the 2726 features in CSLB and for each of the 100,000 images in ESP. This sparse encoding matches the approach taken by Silberer and Lapata (2012), for CCA and concatenation, and by Hill et al. (2014) for the ridge regression method of propagation (see below).

We compare these alternatives to our proposed model with $\alpha = 1$. In The CSLB and ESP models, all training pseudo-sentences are generated from the arrays $\mathbf{P_{CSLB}}$ and $\mathbf{P_{ESP}}$ respectively. In the models classed as *CSLB&ESP*, a random choice between $\mathbf{P_{CSLB}}$ and $\mathbf{P_{ESP}}$ is made every time perceptual input is included (so that the overall quantity of perceptual information is the same).

As shown in Figure 2 (left side), the embeddings learned by our model achieve a higher correlation with the USF data than simple concatenation, CCA and SVD regardless of perceptual input source. With the optimal perceptual source (ESP only), for instance, the correlation is 11% higher that the next best alternative method, CCA.

One possible factor behind this improvement is that, in our model, the learned representations fully integrate the two modalities, whereas for both CCA and the concatenation method each representation feature (whether of reduced dimension or not) corresponds to a particular modality. This deeper integration may help our architecture to overcome the challenges inherent in information combination such as inter-modality differences in information content and representation sparsity. It is also important to note that Bruni et al. (2014) applied their SVD method with comparatively dense perceptual representations extracted from images, whereas our dataset-based perceptual vectors were sparsely-encoded.

[3]CCA was implemented using the *CCA* package in R. SVD was implemented using *SVDLIBC* (`http://tedlab.mit.edu/~dr/SVDLIBC/`), with truncation factor $k = 1024$ as per (Bruni et al., 2014).

3.2 Propagating input to abstract concepts

To test the process of information propagation in our model, we evaluated the learned embeddings of more abstract concepts. We compared our approach with two recently-proposed alternative methods for inferring perceptual features when explicit perceptual information is unavailable.

Johns and Jones In the method of Johns and Jones (2012), pseudo-perceptual representations for target concepts without a perceptual representations (uni-modal concepts) are inferred as a weighted average of the perceptual representations of concepts that do have such a representation (bi-modal concepts).

In the first step of their two-step method, for each uni-modal concept $\mathbf{k}$, a quasi-perceptual representation is computed as an average of the perceptual representations of bi-modal concepts, weighted by the proximity between each of these concepts and $\mathbf{k}$

$$\mathbf{k}^p = \sum_{\mathbf{c} \in \bar{C}} S(\mathbf{k}^l, \mathbf{c}^l)^\lambda \cdot \mathbf{c}^p$$

where $\bar{C}$ is the set of bi-modal concepts, $\mathbf{c}^p$ and $\mathbf{k}^p$ are the perceptual representations for $\mathbf{c}$ and $\mathbf{k}$ respectively, and $\mathbf{c}^l$ and $\mathbf{k}^l$ the linguistic representations. The exponent parameter λ reflects the learning rate.

In step two, the initial quasi-perceptual representations are inferred for a second time, but with the weighted average calculated over the perceptual or initial quasi-perceptual representations of all other words, not just those that were originally bi-modal. As with Johns and Jones (2012), we set the learning rate parameter λ to be 3 in the first step and 13 in the second.

Ridge Regression An alternative, proposed for the present purpose by Hill et al. (2014), uses *ridge regression* (Myers, 1990). Ridge regression is a variant of least squares regression in which a regularization term is added to the training objective to favor solutions with certain properties.

For bi-modal concepts of dimension n_p, we apply ridge regression to learn n_p linear functions

$f_i : \mathbb{R}^{n_l} \rightarrow \mathbb{R}$ that map the linguistic representations (of dimension n_l) to a particular perceptual feature i. These functions are then applied together to map the linguistic representations of uni-modal concepts to full quasi-perceptual representations.

Following Hill et al. (2014), we take the Euclidian l_2 norm of the inferred parameter vector as the regularization term. This ensures that the regression favors lower coefficients and a smoother solution function, which should provide better generalization performance than simple linear regression. The objective for learning the f_i is then to minimize

$$\|\mathbf{a}X - Y_i\|_2^2 + \|\mathbf{a}\|_2^2$$

where $\mathbf{a}$ is the vector of regression coefficients, X is a matrix of linguistic representations and Y_i a vector of the perceptual feature i for the set of bi-modal concepts.

Comparisons We applied the Johns and Jones method and ridge regression starting from linguistic embeddings acquired by the Mikolov et al. (2013) model on the Text8 Corpus, and concatenated the resulting pseudo-perceptual and linguistic representations. As with the implementation of our model, the perceptual input for these alternative models was limited to concrete nouns (i.e. concrete nouns were the only bi-modal concepts in the models).

Figure 3 (right side) shows the propagation performance of the three models. While the correlations overall may seem somewhat low, this is a consequence of the difficulty of modelling the USF data. In fact, the performance of both the language-only model and our multi-modal extension across the concept types (from 0.18 to 0.36) is equal to or higher than previous models evaluated on the same data (Feng and Lapata, 2010; Silberer and Lapata, 2012; Silberer et al., 2013).

For learning representations of concrete verbs, our approach achieves a 69% increase in performance over the next best alternative. The performance of the model on abstract verbs is marginally inferior to Johns and Jones' method. Nevertheless, the clear advantage for concrete verbs makes our model the best choice for learning representations of verbs in general, as shown by performance on the set *all verbs*, which also includes mixed abstract-concrete pairs.

Our model is also marginally inferior to alternative approaches in learning representations of abstract nouns. However, in this case, no method improves on the linguistic-only baseline. It is possible that perceptual information is simply so removed from the core semantics of these concepts that they are best acquired via the linguistic medium alone, regardless of learning mechanism. The moderately inferior performance of our method in such cases is likely caused by its greater inherent inter-modal dependence compared with methods that simply concatenate uni-modal representations. When the perceptual signal is of low quality, this greater inter-modal dependence allows the linguistic signal to be obscured.

The trade-off, however, is generally higher-quality representations when the perceptual signal is stronger, exemplified by the fact that our proposed approach outperforms alternatives on pairs generated from both abstract and concrete nouns (*all nouns*). Indeed, the low performance of the Johns and Jones method on *all nouns* is striking given that: (a) It performs best on *abstract nouns* ($\rho = .282$), and (b) For concrete nouns it reverts to simple concatenation, which also performs comparatively well ($\rho = .249$). The poor performance of the Jobns and Jones method on *all nouns* must therefore derive its comparisons of mixed abstract-concrete or concrete-abstract pairs. This suggests that the pseudo-perceptual representations inferred by this method for abstract concepts method may not be compatible with the directly-encoded perceptual representations of concrete concepts, rendering the comparison computation between items of differing concreteness inaccurate.

3.3 Direct representation vs. propagation

Although property norm datasets such as the CSLB data typically consist of perceptual feature information for concrete nouns only, image-based datasets such as ESP do contain information on more abstract concepts, which was omitted from the previous experiments. Indeed, image banks such as Google Images contain millions of photographs portraying quite abstract concepts, such as *love* or *war*. On the other hand, encodings or descriptions of abstract concepts are generally more subjective and less reliable than those of concrete concepts (Wiemer-Hastings and Xu, 2005). We therefore investigated whether or not it is preferable to include this additional information as model input or to restrict perceptual input

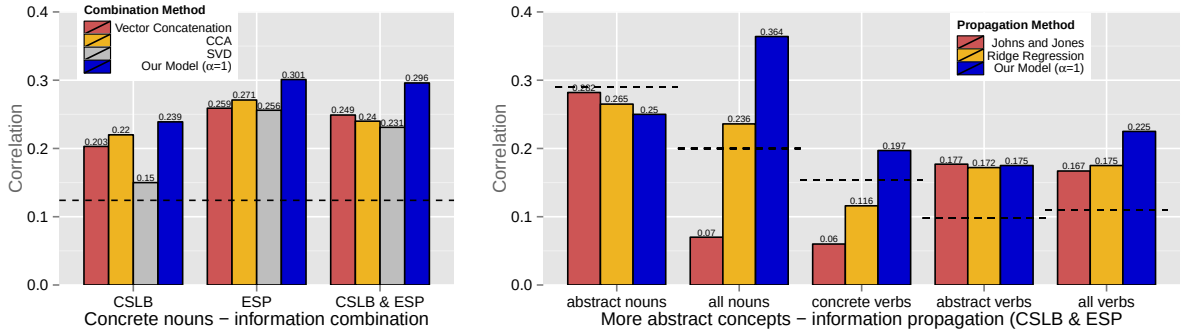


Figure 3: The proposed approach compared with other methods of information combination (left) and propagation. Dashed lines indicate language-only model baseline. For brevity we include both perceptual input sources ESP and CSLB when comparing means of propagation; results with individual information sources were similar.

to concrete nouns as previously.

Of our evaluation sets, it was possible to construct from ESP (and add to $\mathbf{P_{ESP}}$) representations for all of the concrete verbs, and for approximately half of the abstract verbs and abstract nouns. Figure 4 (top), shows the performance of a our model trained on all available perceptual input versus the model in which the perceptual input was restricted to concrete nouns.

The results reflect a clear manifestation of the abstract/concrete distinction. Concrete verbs behave similarly to concrete nouns, in that they can be effectively represented directly from perceptual information sources. The information encoded in these representations is beneficial to the model and increases performance. In contrast, constructing 'perceptual' representations of abstract verbs and abstract nouns directly from perceptual information sources is clearly counter-productive (to the extent that performance also degrades on the combined sets *all nouns* and *all verbs*). It appears in these cases that the perceptual input acts to obscure or contradict the otherwise useful signal inferred from the corpus.

As shown in the previous section, the inclusion of any form of perceptual input inhibits the learning of abstract nouns. However, this is not the case for abstract verbs. Our model learns higher quality representations of abstract verbs if perceptual input is restricted to concrete nouns than if no perceptual input is included at all *and* when perceptual input is included for both concrete nouns and abstract verbs. This supports the idea of a gradual scale of concreteness: The most concrete concepts can be effectively represented directly in the perceptual modality; somewhat more abstract concepts cannot be represented directly in the perceptual modality, but have representations that are improved by propagating perceptual input from concrete concepts via language; and the most abstract concepts are best acquired via language alone.

3.4 Source and quantity of perceptual input

For different concept types, we tested the effect of varying the proportion of perceptual to linguistic input (the parameter α). Perceptual input was restricted to concrete nouns as in Sections 3.1-3.2.

As shown in Figure 4, performance on concrete nouns improves (albeit to a decreasing degree) as α increases. When learning concrete noun representations, linguistic input is apparently redundant if perceptual input is of sufficient quality and quantity. For the other concept types, in each case there is an optimal value for α in the range .5–2, above which perceptual input obscures the linguistic signal and performance degrades. The proximity of these optima to 1 suggests that for optimal learning, when a concrete concept is experienced approximately equal weight should be given to available perceptual and linguistic information.

4 Conclusions

Motivated by the notable prevalence of abstract concepts in everyday language, and their likely importance to flexible, general-purpose representation learning, we have investigated how abstract and concrete representations can be acquired by multi-modal models. In doing so, we presented a simple and easy-to-implement architecture for acquiring semantic representations of both types of

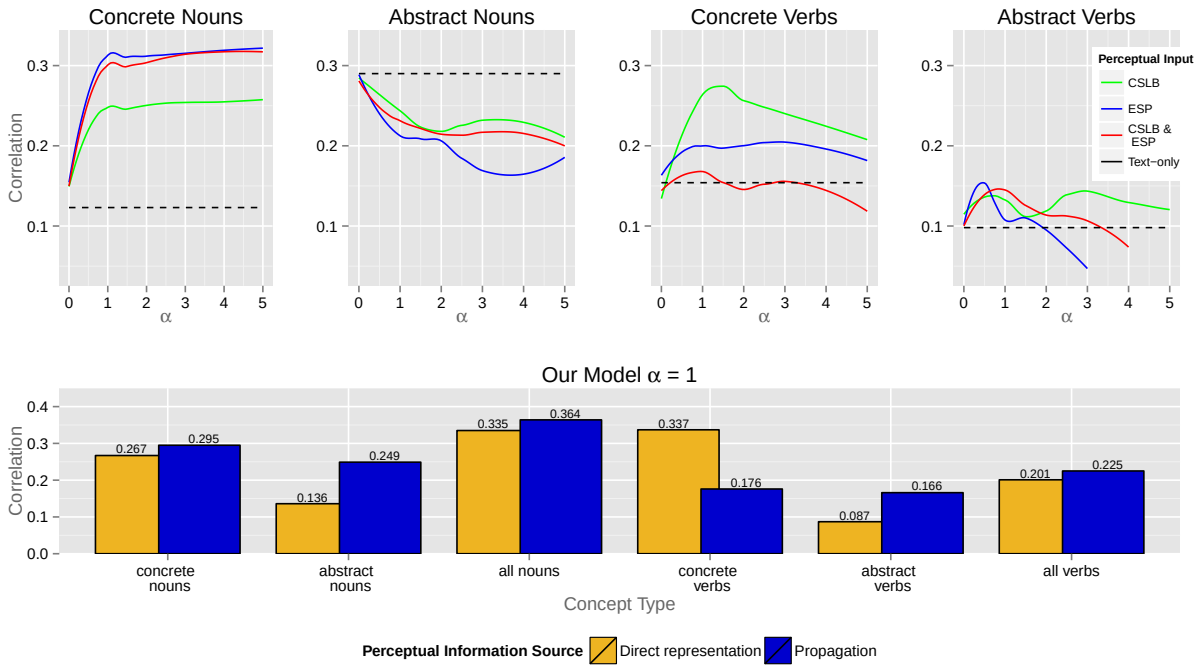


Figure 4: Top: Comparing the strategy of directly representing abstract concepts from perceptual information where available (yellow bars) vs. propagating via concrete concepts. Bottom: The effect of increasing α on correlation with USF pairs (Spearman ρ) for each concept type. Horizontal dashed lines indicate language-only model baseline.

concept from linguistic and perceptual input.

While neuro-probabilistic language models have been applied to the problem of multi-modal representation learning previously (Srivastava and Salakhutdinov, 2012; Wu et al., 2013; Silberer and Lapata, 2014) our model and experiments develop this work in several important ways. First, we address the problem of learning abstract concepts. By isolating concepts of different concreteness and part-of-speech in our evaluation sets, and separating the processes of information combination and propagation, we demonstrate that the multi-modal approach is indeed effective for some, but perhaps not all, abstract concepts. In addition, our model introduces a clear parallel with human language learning. Perceptual input is introduced precisely when concrete concepts are 'experienced' by the model in the corpus text, much like a language learner experiencing concrete entities via sensory perception.

Taken together, our findings indicate the utility of distinguishing three concept types when learning representations in the multi-modal setting.

Type I Concepts that can be effectively represented directly in the perceptual modality. For such concepts, generally concrete nouns or concrete verbs, our proposed approach provides a simple means of combining perceptual and linguistic input. The resulting multi-modal representations are of higher quality than those learned via other approaches, resulting in a performance improvement of over 10% in modelling free association.

Type II Concepts, including abstract verbs, that cannot be effectively represented directly in the perceptual modality, but whose representations can be improved by joint learning from linguistic input and perceptual information about related concepts. Our model can effectively propagate perceptual input (exploiting the relations inferred from the linguistic input) from Type I concepts to enhance the representations of Type II concepts above the language-only baseline. Because of the frequency of abstract concepts, such propagation extends the benefit of the multi-modal approach to a far wider range of language than models based solely in the concrete domain.

Type III Concepts that are more effectively learned via language-only models than multi-modal models, such as abstract nouns. Neither

our proposed approach nor alternative propagation methods achieve an improvement in representation quality for these concepts over the language-only baseline. Of course, it is an empirical question whether a multi-modal approach could ever enhance the representation learning of these concepts, one with potential implications for cognitive theories of grounding (a topic of much debate in psychology (Grafton, 2009; Barsalou, 2010)).

Additionally, we investigated the optimum type and quantity of perceptual input for learning concepts of different types. We showed that too much perceptual input can result in degraded representations. For concepts of type I and II, the optimal quantity resulted from setting $\alpha = 1$; i.e. whenever a concrete concept was encountered, the model learned from an equal number of language-based and perception-based examples. While we make no formal claims here, such observations may ultimately provide insight into human language learning and semantic memory.

In future we will address the question of whether Type III concepts can ever be enhanced via multi-modal learning, and investigate multi-modal models that optimally learn concepts of each type. This may involve filtering the perceptual input stream for concepts according to concreteness, and possibly more elaborate model architectures that facilitate distinct representational frameworks for abstract and concrete concepts.

Acknowledgements

Thanks to the Royal Society and St John's College for supporting this research, and to Yoshua Bengio and Diarmuid Ó Séaghdha for helpful discussions.

References

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

Lawrence W Barsalou and Katja Wiemer-Hastings. 2005. Situating abstract concepts. *Grounding Cognition: The Role of Perception and Action in Memory, Language, and Thought*, pages 129–163.

Lawrence W Barsalou. 2010. Grounded cognition: past, present, and future. *Topics in Cognitive Science*, 2(4):716–724.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

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

Joan L Bybee and Paul J Hopper. 2001. *Frequency and the Emergence of Linguistic Structure*, volume 45. John Benjamins Publishing.

Nick Chater and Christopher D Manning. 2006. Probabilistic models of language processing and acquisition. *Trends in Cognitive Sciences*, 10(7):335–344.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167. ACM.

Sebastian J Crutch and Elizabeth K Warrington. 2005. Abstract and concrete concepts have structurally different representational frameworks. *Brain*, 128(3):615–627.

Barry J Devereux, Lorraine K Tyler, Jeroen Geertzen, and Billi Randall. 2013. The centre for speech, language and the brain (cslb) concept property norms. *Behavior Research Methods*, pages 1–9.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Yansong Feng and Mirella Lapata. 2010. Visual information in semantic representation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 91–99. Association for Computational Linguistics.

Jerry A Fodor. 1983. *The modularity of mind: An essay on faculty psychology*. MIT press.

Scott T Grafton. 2009. Embodied cognition and the simulation of action to understand others. *Annals of the New York Academy of Sciences*, 1156(1):97–117.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664.

Felix Hill, Anna Korhonen, and Christian Bentz. 2013. A quantitative empirical analysis of the abstract/concrete distinction. *Cognitive Science*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Multi-modal models for abstract and concrete concept semantics. *Transactions of the Association for Computational Linguistics*.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

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

Douwe Kiela, Felix Hill, Anna Korhonen, and Stephen Clark. 2014. Improving multi-modal representations using image dispersion: Why less is sometimes more. In *Proceedings of the annual meeting of the Association for Computational Linguistics*. ACL.

Geoffrey Leech, Roger Garside, and Michael Bryant. 1994. Claws4: the tagging of the British National Corpus. In *Proceedings of the 15th conference on Computational linguistics-Volume 1*, pages 622–628. Association for Computational Linguistics.

Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior Research Methods*, 37(4):547–559.

Grégoire Mesnil, Yann Dauphin, Xavier Glorot, Salah Rifai, Yoshua Bengio, Ian J Goodfellow, Erick Lavoie, Xavier Muller, Guillaume Desjardins, David Warde-Farley, et al. 2012. Unsupervised and transfer learning challenge: a deep learning approach. *Journal of Machine Learning Research-Proceedings Track*, 27:97–110.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of International Conference of Learning Representations*, Scottsdale, Arizona, USA.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the international Workshop on Artificial Intelligence and Statistics*, pages 246–252.

Raymond H Myers. 1990. *Classical and Modern Regression with Applications*, volume 2. Duxbury Press Belmont, CA.

Douglas L Nelson, Cathy L McEvoy, and Thomas A Schreiber. 2004. The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407.

Allan Paivio. 1991. Dual coding theory: Retrospect and current status. *Canadian Journal of Psychology*, 45(3):255.

Stephen Roller and Sabine Schulte im Walde. 2013. A multimodal LDA model integrating textual, cognitive and visual modalities. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1146–1157, Seattle, Washington, USA, October. Association for Computational Linguistics.

Carina Silberer and Mirella Lapata. 2012. Grounded models of semantic representation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1423–1433. Association for Computational Linguistics.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of the annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Carina Silberer, Vittorio Ferrari, and Mirella Lapata. 2013. Models of semantic representation with visual attributes. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics, Sofia, Bulgaria, August*.

Nitish Srivastava and Ruslan Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In *NIPS*, pages 2231–2239.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Luis Von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 319–326. ACM.

Katja Wiemer-Hastings and Xu Xu. 2005. Content differences for abstract and concrete concepts. *Cognitive Science*, 29(5):719–736.

Pengcheng Wu, Steven CH Hoi, Hao Xia, Peilin Zhao, Dayong Wang, and Chunyan Miao. 2013. Online multimodal deep similarity learning with application to image retrieval. In *Proceedings of the 21st ACM International Conference on Multimedia*, pages 153–162. ACM.

Go Climb a Dependency Tree and Correct the Grammatical Errors

Longkai Zhang Houfeng Wang
Key Laboratory of Computational Linguistics (Peking University)
Ministry of Education, China
zhlongk@qq.com, wanghf@pku.edu.cn

Abstract

State-of-art systems for grammar error correction often correct errors based on word sequences or phrases. In this paper, we describe a grammar error correction system which corrects grammatical errors at tree level directly. We cluster all error into two groups and divide our system into two modules correspondingly: the general module and the special module. In the general module, we propose a TreeNode Language Model to correct errors related to verbs and nouns. The TreeNode Language Model is easy to train and the decoding is efficient. In the special module, two extra classification models are trained to correct errors related to determiners and prepositions. Experiments show that our system outperforms the state-of-art systems and improves the F_1 score.

1 Introduction

The task of grammar error correction is difficult yet important. An automatic grammar error correction system can help second language (L2) learners improve the quality of their writing. In recent years, there are various competitions devoted to grammar error correction, such as the HOO-2011(Dale and Kilgarriff, 2011), HOO-2012(Dale et al., 2012) and the CoNLL-2013 shared task (Ng et al., 2013). There has been a lot of work addressing errors made by L2 learners. A significant proportion of the systems for grammar error correction train individual statistical models to correct each special kind of error word by word and ignore error interactions. These methods assume no interactions between different kinds of grammatical errors. In real problem settings errors are correlated, which makes grammar error correction much more difficult.

Recent research begins to focus on the error interaction problem. For example, Wu and Ng (2013) decodes a global optimized result based on the individual correction confidence of each kind of errors. The individual correction confidence is still based on the noisy context. Rozovskaya and Roth (2013) uses a joint modeling approach, which considers corrections in phrase structures instead of words. For dependencies that are not covered by the joint learning model, Rozovskaya and Roth (2013) uses the results of Illinois system in the joint inference. These results are still at word level and are based on the noisy context. These systems can consider error interactions, however, the systems are complex and inefficient. In both Wu and Ng (2013) and Rozovskaya and Roth (2013), Integer Linear Programming (ILP) is used for decoding a global optimized result. In the worst case, the time complexity of ILP can be exponent.

In contrast, we think a better grammar error correction system should correct grammatical errors at sentence level directly and efficiently. The system should correct as many kinds of errors as possible in a generalized framework, while allowing special models for some kinds of errors that we need to take special care. We cluster all error into two groups and correspondingly divide our system into two modules: the general module and the special module. In the general module, our system views each parsed sentence as a dependency tree. The system generates correction candidates for each node on the dependency tree. The correction can be made on the dependency tree globally. In this module, nearly all replacement errors related to verb form, noun form and subject-verb agreement errors can be considered. In the special module, two extra classification models are used to correct the determiner errors and preposition errors . The classifiers are also trained at tree node level. We take special care of these two kinds

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 266–277,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

of errors because these errors not only include replacement errors, but also include insertion and deletion errors. A classification model is more suitable for handling insertion and deletion errors. Besides, they are the most common errors made by English as a Second Language (ESL) learners and are much easier to be incorporated into a classification framework.

We propose a TreeNode Language Model (TNLM) to efficiently measure the correctness of selecting a correction candidate of a node in the general module. Similar to the existing statistical language models which assign a probability to a linear chain of words, our TNLM assigns correctness scores directly on each node on the dependency tree. We select candidates for each node to maximize the global correctness score and use these candidates to form the corrected sentence. The global optimized inference can be tackled efficiently using dynamic programming. Because the decoding is based on the whole sentence, error interactions can be considered. Our TNLM only needs to use context words related to each node on the dependency tree. Training a TreeNode language model costs no more than training ordinary language models on the same corpus. Experiments show that our system can outperform the state-of-art systems.

The paper is structured as follows. Section 1 gives the introduction. In section 2 we describe the task and give an overview of the system. In section 3 we describe the general module and in section 4 we describe the special module. Experiments are described in section 5. In section 6 related works are introduced, and the paper is concluded in the last section.

2 Task and System Overview

2.1 Task Description

The task of grammar error correction aims to correct grammatical errors in sentences. There are various competitions devoted to the grammar error correction task for L2 learners. The CoNLL-2013 shared task is one of the most famous, which focuses on correcting five types of errors that are commonly made by non-native speakers of English, including determiner, preposition, noun number, subject-verb agreement and verb form errors. The training data released by the task organizers come from the NUCLE corpus(Dahlmeier et al., 2013). This corpus contains essays written by ESL learners, which are then corrected by English teachers. The test data are 50 student essays. Details of the corpus are described in Ng et al. (2013).

2.2 System Architecture

In our system, lists of correction candidates are first generated for each word. We generate candidates for nouns based on their plurality. We generate candidates for verbs based on their tenses. Then we select the correction candidates that maximize the overall correctness score. An example process of correcting figure 1(a) is shown in table 1.

Correcting grammatical errors using local statistical models on word sequence is insufficient. The local models can only consider the contexts in a fixed window. In the example of figure 1(a), the context of the verb "is" is "that boy is on the", which sounds reasonable at first glance but is incorrect when considering the whole sentence. The limitation of local classifiers is that long distance syntax information cannot be incorporated within the local context. In order to effectively use the syntax information to get a more accurate correcting result, we think a better way is to tackle the problem directly at tree level to view the sentence as a whole. From figure 1(a) we can see that the node "is" has two children on the dependency tree: "books" and "on". When we consider the node "is", its context is "books is on", which sounds incorrect. Therefore, we can make better corrections using such context information on nodes.

Therefore, our system corrects grammatical errors on dependency trees directly. Because the correlated of words are more linked on trees than in a word sequence, the errors are more easier to be corrected on the trees and the agreement of different error types is guaranteed by the edges. We follow the strategy of treating different kinds of errors differently, which is used by lots of grammar error correction systems. We cluster the five types of errors considered in CoNLL-2013 into two groups and divide our system into two modules correspondingly.

- **The general module**, which is responsible for the verb form errors, noun number errors and subject-verb agreement errors. These errors are all replacement errors, which can be corrected by replacing the wrongly used word with a reasonable candidate word.

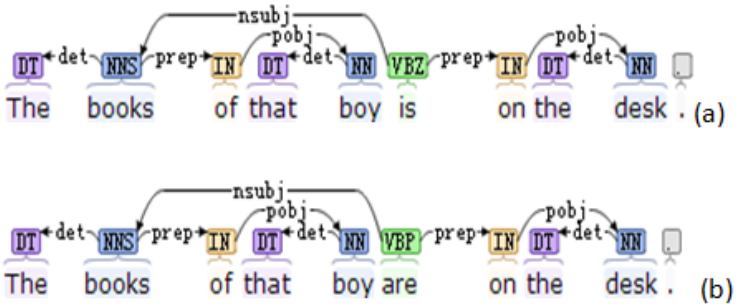


Figure 1: Dependency parsing results of (a) the original sentence "The books of that boy is on the desk ." (b) the corrected sentence.

Position	Original	Correction Candidates	Corrected
1	The	The	The
2	books	books, book	books
3	of	of	of
4	that	that	that
5	boy	boy, boys	boy
6	is	is,are,am,was,were,be,being,been	are
7	on	on	on
8	the	the	the
9	desk	desk, desks	desk
10	.	.	.

Table 1: An example of the "correction candidate generation and candidate selection" framework.

- **The special module**, where two classification models are used to correct the determiner errors and preposition errors at tree level. We take special care of these two kinds of errors because these errors include both replacement errors and insertion/deletion errors. Besides, they are the most common errors made by ESL learners and is much easier to be incorporated into a classification framework.

We should make it clear that we are not the first to use tree level correction models on ungrammatical sentences. Yoshimoto et al. (2013) uses a Treelet Language model (Pauls and Klein, 2012) to correct agreement errors. However, the performance of Treelet language model is not that good compared with the top-ranked system in CoNLL-2013. The reason is that the production rules in the Treelet language model are based on complex contexts, which will exacerbate the data sparseness problem. The "context" in Treelet language model also include words ahead of treelets, which are sometimes unrelated to the current node. In contrast, our TreeNode Language model only needs to consider useful context words related to each node on the dependency tree. To train a TreeNode language model costs no more than training ordinary language models on the same corpus.

2.3 Data Preparation

Our system corrects grammatical errors on dependency trees directly, therefore the sentences in training and testing data should have been parsed before being corrected. In our system, we use the Stanford parser[1] to parse the New York Times source of the Gigaword corpus[2], and use the parsed sentences as our training data. We use the original training data provided by CoNLL-2013 as the develop set to tune all parameters.

Some sentences in the news texts use a different writing style against the sentences written by ESL learners. For example, sentences written by ESL learners seldom include dialogues between people, while very often news texts include paragraphs such as "'I am frightened!' cried Tom". We use heuristic rules to eliminate the sentences in the Gigaword corpus that are less likely to appear in the ESL writing. The heuristic rules include delet-

[1]http://nlp.stanford.edu/software/lex-parser.shtml
[2]https://catalog.ldc.upenn.edu/LDC2003T05

ing sentences that are too short or too long[3], deleting sentences that contains certain punctuations such as quotation marks, or deleting sentences that are not ended with a period.

In total we select and parse 5 million sentences of the New York Times source of English newswire in the Gigaword corpus. We build the system and experiment based on these sentences.

3 The General Module

3.1 Overview

The general module aims to correct verb form errors, noun number errors and subject-verb agreement errors. Other replacement errors such as spelling errors can also be incorporated into the general module. Here we focus on verb form errors, noun number errors and subject-verb agreement errors only. Our general module views each sentence as a dependency tree. All words in the sentence form the nodes of the tree. Nodes are linked through directed edges, annotated with the dependency relations.

Before correcting the grammatical errors, the general module should generate correction candidates for each node first. For each node we use the word itself as its first candidate. Because the general module considers errors related to verbs and nouns, we generate extra correction candidates only for verbs and nouns. For verbs we use all its verb forms as its extra candidates. For example when considering the word "speaks", we use itself and {speak, spoke, spoken, speaking} as its correction candidates. For nouns we use its singular form and plural form as its extra correction candidates. For example when considering the word "dog", we use itself and "dogs" as its correction candidate. If the system selects the original word as the final correction, the sentence remains unchanged. But for convenience we still call the newly generated sentence "the corrected sentence".

In a dependency tree, the whole sentence s can be formulized as a list of production rules $r_1, ..., r_L$ of the form: $[r = head \rightarrow modifier_1, modifier_2 ...]$. An example of all production rules of figure 1(a) is shown in table 2. Because the production rules are made up of words, selecting a different correction candidate for only one node will result in a list of different production rules. For example, figure 1(b) selects the correction candidate "is" to replace the original "are". Therefore the production rules of figure 1(b) include [are → books, on], instead of [is → books, on] in figure 1(a).

books → The, of
of → boy
boy → that
is → books, on
on → desk
desk → the

Table 2: All the production rules in the example of figure 1(a)

The overall correctness score of s, which is $score(s)$, can be further decomposed into $\prod_{i=0}^{L} score(r_i)$. A reasonable score function should score the correct candidate higher than the incorrect one. Consider the node "is" in Figure 1(a), the production rule with head "is" is $[is \rightarrow books, on]$. Because the correction of "is" is "are", a reasonable scorer should have $score([is \rightarrow books, on]) < score([are \rightarrow books, on])$.

Given the formulation of sentence $s = [r_1, ..., r_L]$ and the candidates for each node, we are faced with two problems:

1. **Score Function**. Given a fixed selection of candidate for each node, how to compute the overall score of the dependency tree, i.e., $score(s)$. Because $score(s)$ is decomposed into $\prod_{i=0}^{L} score(r_i)$, the problem becomes finding a $score$ function to measure the correctness of each r given a fixed selection of candidates.

2. **Decoding**. Given each node a list of correction candidates and a reasonable score function $score(r)$ for the production rules, how to find the selection of candidates that maximize the overall score of the dependency tree.

For the first problem, we propose a TreeNode Language Model as the correctness measure of a fixed candidate selection. For the decoding problem, we use a dynamic programming method to efficiently find the correction candidates that maximize the overall score. We will describe the details in the following sections.

One concern is whether the automatically parsed trees are reliable for grammar error correction. We define "reliable" as follows. If we

[3] In our experiment, no less than 5 words and no more than 30 words.

change some words in original sentence into their reasonable correction candidates (e.g. change "is" to "are") but the structure of the dependency tree does not change (except the replaced word and its corresponding POS tag, which are definitely changed), then we say the dependency tree is reliable for this sentence. To verify this we randomly selected 1000 sentences parsed by the Stanford Parser. We randomly select the verbs and nouns and replace them with a wrong form. We parsed the modified sentences again and asked 2 annotators to examine whether the dependency trees are reliable for grammar error correction. We find that 99% of the dependency trees are reliable. Therefore we can see that the dependency tree can be used as the structure for grammar error correction directly.

3.2 TreeNode Language Model

In our system we use the score of TreeNode Language Model (TNLM) as the scoring function. Consider a node n on a dependency tree and assume n has K modifiers $C_1, ..., C_K$ as its child nodes. We define $Seq(n) = [C_1, ..., n, ..., C_K]$ as an ordered sub-sequence of nodes that includes the node n itself and all its child nodes. The order of the sub-sequence in $Seq(n)$ is sorted based on their position in the sentence. In this formulation, we can score the correctness of a production rule r by scoring the correctness of $Seq(n)$. Because $Seq(n)$ is a word sequence, we can use a language model to measure its correctness. The sub-sequences are not identical to the original text. Therefore instead of using ordinary language models, we should train special language models using the sub-sequences to measure the correctness of a production rule.

Take the sentence in figure 2 as an example. When considering the node "is" in the word sequence, it is likely to be corrected into "are" because it appear directly after the plural noun "parents". However, by the definition above, the subsequence corresponding to the node "damaged" is "car is damaged by ". In such context, the word "is" is less likely to be changed to "are". From the example we can see that the sub-sequence is suitable to be used to measure the correctness of a production rule. From this example we can also find that the sub-sequences are different with ordinary sentences, because ordinary sentences are less likely to end with "by".

Table 3 shows all the sub-sequences in the example of figure 2. If we collect all the subsequences in the corpus to form a new subsequence corpus, we can train a language model based on the new sub-sequence corpus. This is our TreeNode Language Model. One advantage of TLM is that once we have generated the subsequences, we can train the TLM in the same way as we train ordinary language models. Besides, the TLM is not limited to a fixed smoothing method. Any smoothing methods for ordinary language models are applicable for TLM.

Node	Sub-sentence
The	The
car	The car of
of	of parents
my	my
parents	my parents
is	is
damaged	car is damaged by
by	by storm
the	the
storm	the storm

Table 3: All the sub-sentences in the example of figure 2

In our system we train the TLM using the same way as training tri-gram language model. For a sub-sequence $S = w_0...w_L$, we calculate $P(S) = \prod_{i=0}^{L} P(w_i|w_{i-1}w_{i-2})$. The smoothing method we use is interpolation, which assumes the final $P'(w_i|w_{i-1}w_{i-2})$ of the tri-gram language model follows the following decomposition:

$$\begin{aligned} P'(w_i|w_{i-1}w_{i-2}) =& \lambda_1 P(w_i|w_{i-1}w_{i-2}) \\ &+ \lambda_2 P(w_i|w_{i-1}) \\ &+ \lambda_3 P(w_i) \end{aligned} \quad (1)$$

where λ_1, λ_2 and λ_3 are parameters sum to 1. The parameters λ_1, λ_2 and λ_3 are estimated through EM algorithm(Baum et al., 1970; Dempster et al., 1977; Jelinek, 1980).

3.3 Decoding

The decoding problem is to select one correction candidate for each node that maximizes the overall score of the corrected sentence. When the sentence is long and contains many verbs and nouns, enumerating all possible candidate selections is time-consuming. We use a bottom-up dynamic

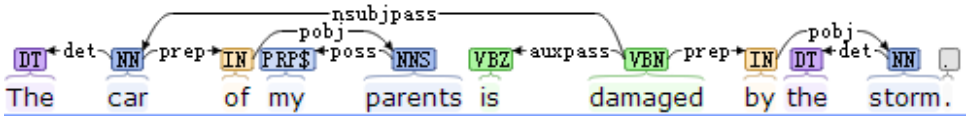


Figure 2: A illustrative sentence for TreeNode Language Model.

programming approach to find the maximized corrections within polynomial time complexity.

For a node n with L correction candidates $n_1, ...n_L$ and K child nodes $C_1, ..., C_K$, we define $n.scores[i]$ as the maximum score if we choose the ith candidate n_i for n. Because we decode from leaves to the root, $C_1.scores, ..., C_K.scores$ have already been calculated before we calculate $n.scores$.

We assume the sub-sequence $Seq(n_i) = [C_1, ..., C_M, n_i, C_{M+1}, ..., C_K]$ without loss of generality, where $C_1, .., C_M$ are the nodes before n_i and $C_{M+1}, ..., C_K$ are the nodes after n_i.

We define $c_{i,j}$ as the jth correction candidate of child node C_i. Given a selection of candidates for each child node $seq = [c_{1,j_1}, ..., c_{M,j_M}, n_i, c_{M+1,j_{M+1}}, ..., c_{K,j_K}]$, we can calculate $score(seq)$ as:

$$score(seq) = TNLM(seq) \prod_{i=1}^{K} C_i.scores[j_i] \quad (2)$$

where $TNLM(seq)$ is the TreeNode Language Model score of seq. Then, $n.scores[i]$ is calculated as:

$$n.scores[i] = \max_{\forall seq} score(seq) \quad (3)$$

Because seq is a word sequence, the maximization can be efficiently calculated using Viterbi algorithm (Forney Jr, 1973). To be specific, the Viterbi algorithm uses the transition scores and emission scores as its input. The transition scores in our model are the tri-gram probabilities from our tri-gram TNLM. The emission scores in our model are the candidate scores of each child: $C_1.scores, ..., C_K.scores$, which have already been calculated.

After the bottom-up calculation, we only need to look into the "ROOT" node to find the maximum score of the whole tree. Similar to the Viterbi algorithm, back pointers should be kept to find which candidate is selected for the final corrected sentence. Detailed decoding algorithm is shown in table 4.

```
Function decode(Node n)
  if n is leaf
    set n.scores uniformly
    return
  for each child c of n
    decode(c)
  calculating n.scores using Viterbi
End Function
BEGIN
  decode(ROOT)
  find the maximum score for the tree and backtrack all candidates
END
```

Table 4: The Decoding algorithm

In the real world implementations, we add a controlling parameter for the confidence of the correctness of the inputs. We multiply λ on $P(w_0|w_{-2}w_{-1})$ of the tri-gram TNLM if the correcting candidate w_0 is the same word in the original input. λ is larger than 1 to "emphasis" the confidence of the original word because the most of the words in the inputs are correct. The value of λ can be set using the development data.

3.4 The Special Module

The special module is designed for determiner errors and preposition errors. We take special care of these two kinds of errors because these errors include insertion and deletion errors, which cannot be corrected in the general module. Because there is a fixed number of prepositions and determiners, these two kinds of errors are much easier to be incorporated into a classification framework. Besides, they are the most common errors made by ESL learners and there are lots of previous works that leave valuable guidance for us to follow.

Similar to many previous state-of-art systems, we treat the correction of determiner errors and preposition errors as a classification problem. Although some previous works (e.g. Rozovskaya et al. (2013)) use NPs and the head of NPs as

features, they are basically local classifiers making predictions on word sequences. Difference to the local classifier approaches, we make predictions on the nodes of the dependency tree directly. In our system we correct determiner errors and preposition errors separately.

For the determiner errors, we consider the insertion, deletion and replacement of articles (i.e. 'a', 'an' and 'the'). Because the articles are used to modify nouns in the dependency trees, we can classify based on noun nodes. We give each noun node (node whose POS tag is noun) a label to indicate which article it should take. We use left position (LP) and right position (RP) to specify the position of the article. The article therefore locates between LP and RP. If a noun node already has an article as its modifier, then LP will be the position directly ahead of the article. In this case, $RP = LP + 2$. If an insertion is needed, the RP is the position of the first child node of the noun node. In this case $LP = RP - 1$. With this notation, detailed feature templates we use to correct determiner errors are listed in table 5. In our model we use 3 labels: 'a', 'the' and '$\emptyset$'. We use 'a', 'the' to represent a noun node should be modified with 'a' or ''the' correspondingly. We use ''$\emptyset$' to indicate that no article is needed for the noun node. We use rule-based method to distinguish between "a" and "an" as a post-process.

For the preposition errors, we only consider deletion and replacement of an existing preposition. The classification framework is similar to determiner errors. We consider classification on preposition nodes (nodes whose POS tag is preposition). We use prepositions as labels to indicate which preposition should be used. and use "$\emptyset$" to denote that the preposition should be deleted. We use the same definition of LP and RP as the correction of determiner errors. Detailed feature templates we use to correct preposition errors are listed in table 6. Similar to the previous work(Xing et al., 2013), we find that adding more prepositions will not improve the performance in our experiments. Thus we only consider a fixed set of prepositions: {in, for, to, of, on}.

Previous works such as Rozovskaya et al. (2013) show that Naive Bayes model and averaged perceptron model show better results than other classification models. These classifiers can give a reasonably good performance when there are limited amount of training data. In our system, we use large amount of automatically generated training data based on the parsed Gigaword corpus instead of the limited training data provided by CoNLL-2013.

Take generating training data for determiner errors as an example. We generate training data based on the parsed Gigaword corpus C described in section 2. Each sentence S in C is a dependency tree T. We use each noun node N on T as one training instance. If N is modified by "the", its label will be "THE". If N is modified by "a" or "an", its label will be "A". Otherwise its label will be "NULL". Then we just omit the determiner modifier and generate features based on table 5. Generating training data for preposition errors is the same, except we use preposition nodes instead of noun nodes.

By generating training instances in this way, we can get large amount of training data. Therefore we think it is a good time to try different classification models with enough training data. We experiment on Naive Bayes, Averaged Perceptron, SVM and Maximum Entropy models (ME) in a 5-fold cross validation on the training data. We find ME achieves the highest accuracy. Therefore we use ME as the classification model in our system.

4 Experiment

4.1 Experiment Settings

In the experiments, we use our parsed Gigaword corpus as the training data, use the training data provided by CoNLL-2013 as the develop data, and use the test data of CoNLL-2013 as test data directly. In the general module, the training data is used for the training of TreeNode Language Model. In the special module, the training data is used for training individual classification models.

We use the M2 scorer (Dahlmeier and Ng, 2012b) provided by the organizer of CoNLL-2013 for the evaluation of our system. The M2 scorer is widely used as a standard scorer in previous systems. Because we make comparison with the state-of-art systems on the CoNLL-2013 corpus, we use the same evaluation metric F_1 score of M2 scorer as the evaluation metric.

In reality, some sentences may have more than one kind of possible correction. As the example in "The books of that boy is on the desk.", the corresponding correction can be either "The books of that boy are on the desk." or "The book of that boy is on the desk.". The gold test data can only con-

Word Features	w_{LP} , w_{LP-1}, w_{LP-2}, w_{RP}, w_{RP+1}, w_{RP+2}, $w_{LP-2}w_{LP-1}$, $w_{LP-1}w_{LP}$, $w_{LP}w_{RP}$, $w_{RP}w_{RP+1}$, $w_{RP+1}w_{RP+2}$, $w_{LP-2}w_{LP-1}w_{LP}$, $w_{LP-1}w_{LP}w_{RP}$, $w_{LP}w_{RP}w_{RP+1}$, $w_{RP}w_{RP+1}w_{RP+2}$
Noun Node Features	NN, $w_{LP}NN$, $w_{LP-1}w_{LP}NN$, $w_{LP-2}w_{LP-1}w_{LP}NN$
Father/Child Node Features	Fa, $w_{RP}Fa$, $w_{RP}w_{RP+1}Fa$, $w_{RP}w_{RP+1}w_{RP+2}Fa$, $Fa\&Ch$

Table 5: Feature templates for the determiner errors. w_i is the word at the ith position. NN is the current noun node. Fa is the father node of the current noun node. Ch is a child node of the current noun node.

Word Features	w_{LP} , w_{LP-1}, w_{LP-2}, w_{RP}, w_{RP+1}, w_{RP+2}, $w_{LP-2}w_{LP-1}$, $w_{LP-1}w_{LP}$, $w_{LP}w_{RP}$, $w_{RP}w_{RP+1}$, $w_{RP+1}w_{RP+2}$, $w_{LP-2}w_{LP-1}w_{LP}$, $w_{LP-1}w_{LP}w_{RP}$, $w_{LP}w_{RP}w_{RP+1}$, $w_{RP}w_{RP+1}w_{RP+2}$
Father/Child Node Features	Fa, $w_{RP}Fa$, $w_{RP}w_{RP+1}Fa$, $w_{RP}w_{RP+1}w_{RP+2}Fa$, $Fa\&Ch$

Table 6: Feature templates for preposition errors. w_i is the word at the ith position. Fa is the father node of the current preposition node. Ch is a child node of the current preposition node.

sider a small portion of possible answers. To relieve this, the CoNLL-2013 shared task allows all participating teams to provide alterative answers if they believe their system outputs are also correct. These alterative answers form the "Revised Data" in the shared task, which indeed help evaluate the outputs of the participating systems. However, the revised data only include alterative corrections from the participating teams. Therefore the evaluation is not that fair for future systems. In our experiment we only use the original test data as the evaluation dataset.

4.2 Experiment Results

We first show the performance of each stage of our system. In our system, the general module and the special module correct grammar errors consequently. Therefore in table 7 we show the performance when each component is added to the system.

Method	**P**	**R**	**F1 score**
TNLM	33.96%	17.71%	23.28%
+Det	32.83%	38.28%	35.35%
+Prep	32.64%	39.20%	35.62%

Table 7: Results of each stage in our system. TNLM is the general module. "+Det" is the system containing the general module and determiner part of special module."+Prep" is the final system

We evaluate the effect of using TreeNode language model for the general module. We compare the TNLM with ordinary tri-gram language model. We use the same amount of training data and the same smoothing strategy (i.e. interpolation) for both of them. Table 8 shows the comparison. The TNLM can improve the F_1 by $+2.1\%$.

Method	**P**	**R**	**F1 score**
Ordinary LM	29.27%	16.68%	21.27%
Our TNLM	**33.96%**	**17.71%**	**23.28%**

Table 8: Comparison for the general module between TNLM and ordinary tri-gram language model on the test data.

Based on the result of the general module using TNLM, we compare our tree level special module against the local classification approach. The special module of our system makes predictions on the dependency tree directly, while local classification approaches make predictions on linear chain of words and decide the article of a noun Phrase or the preposition of a preposition phrase. We use the same word level features for the two approaches except for the local classifiers we do not add tree level features. Table 9 shows the comparison.

When using the parsed Gigaword texts as training data, the quality of the sentences we select will influence the result. For comparison, we randomly select the same amount of sentences from the same source of Gigaword and parse them as a alterative training set. Table 10 shows the comparison between random chosen training data and

Method	P	R	F1 score
Local Classifier	26.38%	39.14%	31.51%
Our Tree-based	**32.64%**	**39.20%**	**35.62%**

Table 9: Comparison for the special module on the test data. The input of the special module is the sentences corrected by the TNLM in the general module.

the selected training data of our system. We can see that the data selection (cleaning) procedure is important for the improvement of system $F1$.

Method	P	R	F1 score
Random	31.89%	35.85%	33.75%
Selected	**32.64%**	**39.20%**	**35.62%**

Table 10: Comparison of training using random chosen sentences and selected sentences.

Method	F1 score
Rozovskaya et al. (2013)	31.20%
Kao et al. (2013)	25.01%
Yoshimoto et al. (2013)	22.17%
Rozovskaya and Roth (2013)	35.20%
Our method	**35.62**%

Table 11: Comparison of $F1$ of different systems on the test data .

4.3 Comparison With Other Systems

We also compare our system with the state-of-art systems. The first two are the top-2 systems at CoNLL-2013 shared task : Rozovskaya et al. (2013) and Kao et al. (2013). The third one is the Treelet Language Model in Yoshimoto et al. (2013). The fourth one is Rozovskaya and Roth (2013), which until now shows the best performance. The comparison on the test data is shown in table 11.

In CoNLL-2013 only 5 kinds of errors are considered. Our system can be slightly modified to handle the case where other errors such as spelling errors should be considered. In that case, we can modify the candidate generation of the general module. We only need to let the generate correction candidates be any possible words that are similar to the original word, and run the same decoding algorithm to get the corrected sentence. As a comparison, the ILP systems should add extra scoring system to score extra kind of errors.

5 Related Works

Early grammatical error correction systems use the knowledge engineering approach (Murata and Nagao, 1994; Bond et al., 1996; Bond and Ikehara, 1996; Heine, 1998). However, manually designed rules usually have exceptions. Therefore, the machine learning approach has become the dominant approach recently. Previous machine learning approaches typically formulates the task as a classification problem. Of all the errors, determiner and preposition errors are the two main research topics (Knight and Chander, 1994; AEHAN et al., 2006; Tetreault and Chodorow, 2008; Dahlmeier and Ng, 2011). Features used in the classification models include the context words, POS tags, language model scores (Gamon, 2010), and tree level features (Tetreault et al., 2010). Models used include maximum entropy (AEHAN et al., 2006; Tetreault and Chodorow, 2008), averaged perceptron, Naive Bayes (Rozovskaya and Roth, 2011), etc. Other errors such as verb form and noun number errors also attract some attention recently (Liu et al., 2010; Tajiri et al., 2012).

Recent research efforts have started to deal with correcting different errors jointly (Gamon, 2011; Park and Levy, 2011; Dahlmeier and Ng, 2012a; Wu and Ng, 2013; Rozovskaya and Roth, 2013). Gamon (2011) uses a high-order sequential labeling model to detect various errors. Park and Levy (2011) models grammatical error correction using a noisy channel model. Dahlmeier and Ng (2012a) uses a beam search decoder, which iteratively corrects to produce the best corrected output. Wu and Ng (2013) and Rozovskaya and Roth (2013) use ILP to decode a global optimized result. The joint learning and joint inference are still at word/phrase level and are based on the noisy context. In the worst case, the time complexity of ILP can reach exponent. In contrast, our system corrects grammar errors at tree level directly, and the decoding is finished with polynomial time complexity.

6 Conclusion and Future work

In this paper we describe our grammar error correction system which corrects errors at tree level directly. We propose a TreeNode Language Model and use it in the general module to correct errors related to verbs and nouns. The TNLM is easy to train and the decoding of corrected sentence is efficient. In the special module, two extra classification models are trained to correct errors related to

determiners and prepositions at tree level directly. Because our current method depends on an automatically parsed corpus, future work may include applying some additional filtering (e.g. Mejer and Crammer (2012)) of the extended training set according to some confidence measure of parsing accuracy.

Acknowledgments

This research was partly supported by National Natural Science Foundation of China (No.61370117,61333018), Major National Social Science Fund of China (No.12&ZD227) and National High Technology Research and Development Program of China (863 Program) (No. 2012AA011101). The contact author of this paper, according to the meaning given to this role by Key Laboratory of Computational Linguistics, Ministry of Education, School of Electronics Engineering and Computer Science, Peking University, is Houfeng Wang. We thank Longyue Wang and the reviewers for their comments and suggestions.

References

AEHAN, N., Chodorow, M., and LEACOCK, C. L. (2006). Detecting errors in english article usage by non-native speakers.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, pages 164–171.

Bond, F. and Ikehara, S. (1996). When and how to disambiguate?—countability in machine translation—. In *International Seminar on Multimodal Interactive Disambiguation: MIDDIM-96*, pages 29–40. Citeseer.

Bond, F., Ogura, K., and Kawaoka, T. (1996). Noun phrase reference in japanese-to-english machine translation. *arXiv preprint cmp-lg/9601008*.

Dahlmeier, D. and Ng, H. T. (2011). Grammatical error correction with alternating structure optimization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 915–923. Association for Computational Linguistics.

Dahlmeier, D. and Ng, H. T. (2012a). A beam-search decoder for grammatical error correction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 568–578. Association for Computational Linguistics.

Dahlmeier, D. and Ng, H. T. (2012b). Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572. Association for Computational Linguistics.

Dahlmeier, D., Ng, H. T., and Wu, S. M. (2013). Building a large annotated corpus of learner english: The nus corpus of learner english. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31.

Dale, R., Anisimoff, I., and Narroway, G. (2012). Hoo 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 54–62. Association for Computational Linguistics.

Dale, R. and Kilgarriff, A. (2011). Helping our own: The hoo 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 242–249. Association for Computational Linguistics.

Dempster, A. P., Laird, N. M., Rubin, D. B., et al. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal statistical Society*, 39(1):1–38.

Forney Jr, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Gamon, M. (2010). Using mostly native data to correct errors in learners' writing: a meta-classifier approach. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 163–171. Association for Computational Linguistics.

Gamon, M. (2011). High-order sequence modeling for language learner error detection. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*,

pages 180–189. Association for Computational Linguistics.

Heine, J. E. (1998). Definiteness predictions for japanese noun phrases. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 519–525. Association for Computational Linguistics.

Jelinek, F. (1980). Interpolated estimation of markov source parameters from sparse data. *Pattern recognition in practice*.

Kao, T.-h., Chang, Y.-w., Chiu, H.-w., Yen, T.-H., Boisson, J., Wu, J.-c., and Chang, J. S. (2013). Conll-2013 shared task: Grammatical error correction nthu system description. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 20–25, Sofia, Bulgaria. Association for Computational Linguistics.

Knight, K. and Chander, I. (1994). Automated postediting of documents. In *AAAI*, volume 94, pages 779–784.

Liu, X., Han, B., Li, K., Stiller, S. H., and Zhou, M. (2010). Srl-based verb selection for esl. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1068–1076. Association for Computational Linguistics.

Mejer, A. and Crammer, K. (2012). Are you sure? confidence in prediction of dependency tree edges. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 573–576, Montréal, Canada. Association for Computational Linguistics.

Murata, M. and Nagao, M. (1994). Determination of referential property and number of nouns in japanese sentences for machine translation into english. *arXiv preprint cmp-lg/9405019*.

Ng, H. T., Wu, S. M., Wu, Y., Hadiwinoto, C., and Tetreault, J. (2013). The conll-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

Park, Y. A. and Levy, R. (2011). Automated whole sentence grammar correction using a noisy channel model. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 934–944. Association for Computational Linguistics.

Pauls, A. and Klein, D. (2012). Large-scale syntactic language modeling with treelets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 959–968. Association for Computational Linguistics.

Rozovskaya, A., Chang, K.-W., Sammons, M., and Roth, D. (2013). The university of illinois system in the conll-2013 shared task. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 13–19, Sofia, Bulgaria. Association for Computational Linguistics.

Rozovskaya, A. and Roth, D. (2011). Algorithm selection and model adaptation for esl correction tasks. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 924–933. Association for Computational Linguistics.

Rozovskaya, A. and Roth, D. (2013). Joint learning and inference for grammatical error correction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 791–802, Seattle, Washington, USA. Association for Computational Linguistics.

Tajiri, T., Komachi, M., and Matsumoto, Y. (2012). Tense and aspect error correction for esl learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 198–202. Association for Computational Linguistics.

Tetreault, J., Foster, J., and Chodorow, M. (2010). Using parse features for preposition selection and error detection. In *Proceedings of the acl 2010 conference short papers*, pages 353–358. Association for Computational Linguistics.

Tetreault, J. R. and Chodorow, M. (2008). The ups and downs of preposition error detection in esl writing. In *Proceedings of the 22nd International Conference on Computa-*

tional Linguistics-Volume 1, pages 865–872. Association for Computational Linguistics.

Wu, Y. and Ng, H. T. (2013). Grammatical error correction using integer linear programming. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1456–1465, Sofia, Bulgaria. Association for Computational Linguistics.

Xing, J., Wang, L., Wong, D. F., Chao, L. S., and Zeng, X. (2013). Um-checker: A hybrid system for english grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 34–42, Sofia, Bulgaria. Association for Computational Linguistics.

Yoshimoto, I., Kose, T., Mitsuzawa, K., Sakaguchi, K., Mizumoto, T., Hayashibe, Y., Komachi, M., and Matsumoto, Y. (2013). Naist at 2013 conll grammatical error correction shared task. *CoNLL-2013*, 26.

An Unsupervised Model for Instance Level Subcategorization Acquisition

Simon Baker
Computer Laboratory
University of Cambridge
sb895@cam.ac.uk

Roi Reichart
Technion, IIT
Haifa, Israel
roiri@ie.technion.ac.il

Anna Korhonen
Computer Laboratory
University of Cambridge
alk23@cam.ac.uk

Abstract

Most existing systems for subcategorization frame (SCF) acquisition rely on supervised parsing and infer SCF distributions at type, rather than instance level. These systems suffer from poor portability across domains and their benefit for NLP tasks that involve sentence-level processing is limited. We propose a new unsupervised, Markov Random Field-based model for SCF acquisition which is designed to address these problems. The system relies on supervised POS tagging rather than parsing, and is capable of learning SCFs at instance level. We perform evaluation against gold standard data which shows that our system outperforms several supervised and type-level SCF baselines. We also conduct task-based evaluation in the context of verb similarity prediction, demonstrating that a vector space model based on our SCFs substantially outperforms a lexical model and a model based on a supervised parser [1].

1 Introduction

Subcategorization frame (SCF) acquisition involves identifying the arguments of a predicate and generalizing about its syntactic frames, where each frame specifies the syntactic type and number of arguments permitted by the predicate. For example, in sentences (1)-(3) the verb *distinguish* takes three different frames, the difference between which is not evident when considering the phrase structure categorization:

(1) **Direct Transitive**: [They]NP [distinguished]VP [the mast]NP [of [ships on the horizon]NP]PP .

(2) **Indirect Transitive:** [They]NP [distinguished]VP [between [me and you]ADVP]PP .

(3) **Ditransitive**: [They]NP [distinguished]VP [him]NP [from [the other boys]NP]PP.

As SCFs describe the syntactic realization of the verbal predicate-argument structure, they are highly valuable for a variety of NLP tasks. For example, verb subcategorization information has proven useful for tasks such as parsing (Carroll and Fang, 2004; Arun and Keller, 2005; Cholakov and van Noord, 2010), semantic role labeling (Bharati et al., 2005; Moschitti and Basili, 2005), verb clustering, (Schulte im Walde, 2006; Sun and Korhonen, 2011) and machine translation (hye Han et al., 2000; Hajič et al., 2002; Weller et al., 2013).

SCF induction is challenging. The argument-adjunct distinction is difficult even for humans, and is further complicated by the fact that both arguments and adjuncts can appear frequently in potential argument head positions (Korhonen et al., 2000). SCFs are also highly sensitive to domain variation so that both the frames themselves and their probabilities vary depending on the meaning and behavior of predicates in the domain in question (e.g. (Roland and Jurafsky, 1998; Lippincott et al., 2010; Rimell et al., 2013), Section 4).

Because of the strong impact of domain variation, SCF information is best acquired automatically. Existing data-driven SCF induction systems, however, do not port well between domains. Most existing systems rely on hand-written rules (Briscoe and Carroll, 1997; Korhonen, 2002; Preiss et al., 2007) or simple co-occurrence statistics (O'Donovan et al., 2005; Chesley and Salmon-Alt, 2006; Ienco et al., 2008; Messiant et al., 2008; Lenci et al., 2008; Altamirano and Alonso i Alemany, 2010; Kawahara and Kurohashi, 2010) applied to the grammatical dependency output of supervised statistical parsers. Even the handful of recent systems

[1] The verb similarity dataset used for the evaluation of our model is publicly available at ie.technion.ac.il/∼roiri/.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 278–289,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

that use modern machine learning techniques (Debowski, 2009; Lippincott et al., 2012; Van de Cruys et al., 2012; Reichart and Korhonen, 2013) use supervised parsers to pre-process the data[2].

Supervised parsers are notoriously sensitive to domain variation (Lease and Charniak, 2005). As annotation of data for each new domain is unrealistic, current SCF systems suffer from poor portability. This problem is compounded for the many systems that employ manually developed SCF rules because rules are inherently ignorant to domain-specific preferences. The few SCF studies that focused on specific domains (e.g. biomedicine) have reported poor performance due to these reasons (Rimell et al., 2013).

Another limitation of most current SCF systems is that they produce a *type*-level SCF lexicon (i.e. a lexicon which lists, for a given predicate, different SCF types with their relative frequencies). Such a lexicon provides a useful high-level profile of the syntactic behavior of the predicate in question, but is less useful for downstream NLP tasks (e.g. information extraction, parsing, machine translation) that involve sentence processing and can therefore benefit from SCF information at instance level. Sentences (1)-(3) demonstrate this limitation - a prior distribution over the possible syntactic frames of *distinguish* provides only a weak signal to a sentence level NLP application that needs to infer the verbal argument structure of its input sentences.

We propose a new unsupervised model for SCF induction which addresses these problems with existing systems. Our model does not use a parser or hand-written rules, only a part-of-speech (POS) tagger is utilizes in order to produce features for machine learning. While POS taggers are also sensitive to domain variation, they can be adapted to domains more easily than parsers because they require much smaller amounts of annotated data (Lease and Charniak, 2005; Ringger et al., 2007). However, as we demonstrate in our experiments, domain adaptation of POS tagging may not even be necessary to obtain good results on the SCF acquisition task.

Our model, based on the Markov Random Field (MRF) framework, performs instance-based SCF learning. It encodes syntactic similarities among verb instances across different verb types (derived from a lexical and POS-based feature representation of verb instances) as well as prior beliefs on the tendencies of specific instances of the same verb type to take the same SCF.

We evaluate our model against corpora annotated with verb instance SCFs (Quochi et al., 2012). In addition, following the Levin verb clustering tradition (Levin, 1993) which ties verb meanings with their syntactic properties, we evaluate the semantic predictive power of our clusters. In the former evaluation, our model outperforms a number of strong baselines, including supervised and type-level ones, achieving an accuracy of up to 69.2%. In the latter evaluation a vector space model that utilized our induced SCFs substantially outperforms the output of a type-level SCF system that uses the fully trained Stanford parser.

2 Previous Work

Several SCF acquisition systems are available for English (O'Donovan et al., 2005; Preiss et al., 2007; Lippincott et al., 2012; Van de Cruys et al., 2012; Reichart and Korhonen, 2013) and other languages, including French (Messiant, 2008), Italian (Lenci et al., 2008), Turkish (Uzun et al., 2008), Japanese (Kawahara and Kurohashi, 2010) and Chinese (Han et al., 2008). The prominent input to these systems are grammatical relations (GRs) which express binary dependencies between words (e.g. direct and indirect objects, various types of complements and conjunctions). These are generated by some parsers (e.g. (Briscoe et al., 2006)) and can be extracted from the output of others (De-Marneffe et al., 2006).

Two representative systems for English are the Cambridge system (Preiss et al., 2007) and the BioLexicon system which was used to acquire a substantial lexicon for biomedicine (Venturi et al., 2009). These systems extract GRs at the verb instance level from the output of a parser: the RASP general-language unlexicalized parser[3] (Briscoe et al., 2006) and the lexicalized Enju parser tuned to the biomedical domain (Miyao and Tsujii, 2005), respectively. They generate potential SCFs by mapping GRs to a predefined SCF inventory using a set of manually developed rules (the Cambridge system) or by simply considering the sets of GRs including verbs in question as potential SCFs (BioLexicon). Finally, a type level lexicon

[2](Lippincott et al., 2012) does not use a parser, but the syntactic frames induced by the system do not capture *sets of* arguments for verbs, so are not SCFs in a traditional sense.

[3]A so-called unlexicalized parser is a parser trained without explicit SCF annotations.

is built through noisy frame filtering (based on frequencies or on external resources and annotations), which aims to remove errors from parsing and argument-adjunct distinction. Clearly, these systems require extensive manual work: a-priori definition of an SCF inventory and rules, manually annotated sentences for training a supervised parser, SCF annotations for parser lexicalization, and manually developed resources for optimal filtering.

A number of recent works have applied modern machine learning techniques to SCF induction, including point-wise co-occurrence of arguments (Debowski, 2009), a Bayesian network model (Lippincott et al., 2012), multi-way tensor factorization (Van de Cruys et al., 2012) and Determinantal Point Processes (DPPs) -based clustering (Reichart and Korhonen, 2013). However, all of these systems induce type-level SCF lexicons and, except from the system of (Lippincott et al., 2012) that is not capable of learning traditional SCFs, they all rely on supervised parsers.

Our new system differs from previous ones in a number of respects. First, in contrast to most previous systems, our system provides SCF analysis for each verb instance in its sentential context, yielding more precise SCF information for systems benefiting from instance-based analysis. Secondly, it addresses SCF induction as an unsupervised clustering problem, avoiding the use of supervised parsing or any of the sources of manual supervision used in previous works. Our system relies on POS tags - however, we show that it is not necessary to train a tagger with in-domain data to obtain good performance on this task, and therefore our approach provides a more domain-independent solution to SCF acquisition.

We employ POS-tagging instead of unsupervised parsing for two main reasons. First, while a major progress has been made on unsupervised parsing (e.g. (Cohen and Smith, 2009; Berg-Kirkpatrick et al., 2010)), the performance is still considerably behind that of supervised parsing. For example, the state-of-the-art discriminative model of (Berg-Kirkpatrick et al., 2010) achieves only 63% directed arc accuracy for WSJ sentences of up to 10 words, compared to more than 95% obtained with supervised parsers. Second, current unsupervised parsers produce unlabeled structures which are substantially less useful for SCF acquisition than labeled structures produced by supervised parsers (e.g. grammatical relations).

Finally, a number of recent works addressed related tasks such as argument role clustering for SRL (Lang and Lapata, 2011a; Lang and Lapata, 2011b; Titvo and Klementiev, 2012) in an unsupervised manner. While these works differ from ours in the task (clustering arguments rather than verbs) and the level of supervision (applying a supervised parser), like us they analyze the verb argument structure at the instance level.

3 Model

We address SCF induction as an unsupervised verb instance clustering problem. Given a set of plain sentences, our algorithm aims to cluster the *verb instances* in its input into syntactic clusters that strongly correlate with SCFs. In this section we introduce a Markov Random Field (MRF) model for this task: Section 3.1 describes our model's structure, components and objective; Section 3.2 describes the model potentials and the knowledge they encode; and Section 3.3 describes how clusters are induced from the model.

3.1 Model Structure

We implement our model in the MRF framework (Koller and Friedman, 2009). This enables us to encode the two main sources of information that govern SCF selection in verb instances: (1) At the sentential context, the verbal syntactic frame is encoded through syntactic features. Verb instances with similar feature representations should therefore take the same syntactic frame; and (2) At the global context, per verb type SCF distributions tend to be Zipfian (Korhonen et al., 2000). Instances of the same verb type should therefore be biased to take the same syntactic frame.

Given a collection of plain input sentences, we denote the number of verb instances in the collection with n, and the number of data-dependent equivalence classes (ECs) with K (see below for their definition), and define an undirected graphical model (MRF), $G = (V, E, L)$. We define the vertex set as $V = X \cup C$, with $X = \{x_1, \ldots, x_n\}$ consisting of one vertex for every verb instance in the input collection, and $C = \{c_1 \ldots c_K\}$ consisting of one vertex for each data-dependent EC. The set of labels used by the model, L, corresponds to the syntactic frames taken by the verbs in the input data. The edge set E is defined through the model's potentials that are described below.

We encode information in the model through three main sets of potentials: one set of *singleton potentials* - defined over individual model vertexes, and two sets of *pairwise potentials* - defined between pairs of vertexes. The first set consists of a singleton potential for each vertex in the model. Reflecting the Zipfian distribution of SCFs across the instances of the same verb type, these potentials encourage the model to assign such verb instances to the same frame (cluster). The information encoded in these potentials is induced via a pre-processing clustering step. The second set consists of a pairwise potential for each pair of vertexes $x_i, x_j \in X$ - that is, for each verb instance pair in the input, across verb types. These potentials encode the belief, computed as feature-based similarity (see below), that their verb instance arguments implement the same SCF.

Finally, potentials from the last set bias the model to assign the same SCF to high cardinality sets of cross-type verb instances based on their syntactic context. While these are pairwise potentials defined between verb instance vertexes (X) and EC vertexes (C), they are designed so that they bias the assignment of all verb instance vertexes that are connected to the same EC vertex towards the same frame assignment ($l \in L$). The two types of pairwise potentials complement each other by modeling syntactic similarities among verb instance pairs, as well as among higher cardinality verb instance sets.

The resulted maximum aposteriori problem (MAP) takes the following form:

$$MAP(V) = \arg\max_{x,c \in V} \sum_{i=1}^{n} \theta_i(x_i) + \sum_{i=1}^{n}\sum_{j=1}^{n} \theta_{i,j}(x_i, x_j) +$$

$$\sum_{i=1}^{n}\sum_{j=1}^{K} \phi_{i,j}(x_i, c_j) \cdot I(x_i \in EC_j) + \sum_{i=1}^{K}\sum_{j=1}^{K} \xi_{i,j}(c_i, c_j)$$

where the predicate $I(x_i \in EC_j)$ returns 1 if the i-th verb instance belongs the j-th equivalence class and 0 otherwise. The ξ pairwise potentials defined between EC vertexes are very simple potentials designed to promise different assignments for each pair of EC vertexes. They do so by assigning a $-\infty$ score to assignments where their argument vertexes take the same frame and a 0 otherwise. In the rest of this section we do not get back to this simple set of potentials.

A graphical illustration of the model is given in Figure 1. Note that we could have selected a richer model structure, for example, by defining a similarity potential over all verb instance vertexes that share an equivalence class. However, as the figure demonstrates, even the structure of the pruned version of our model (see Section 3.3) usually contains cycles, which makes inference NP-hard (Shimony, 1994). Our design choices aim to balance between the expressivity of the model and the complexity of inference. In Section 3.3 we describe the LP relaxation algorithm we use for inference.

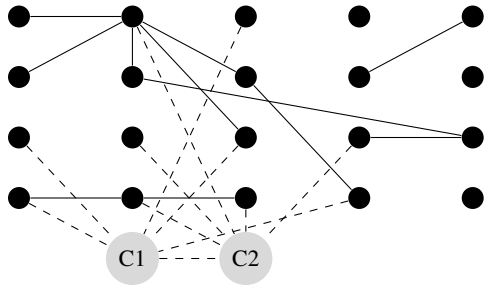


Figure 1: A graphical illustration of our model (after pruning, see Sec. 3.3) for twenty verb instances ($|X| = 20$), each represented with a black vertex, and two equivalence classes (ECs), each represented with a gray vertex ($|C| = 2$). Solid lines represent edges (and $\theta_{i,j}$ pairwise potentials) between verb instance vertexes. Dashed lines represent edges between verb instance vertexes and EC vertexes ($\phi_{i,j}$ pairwise potentials) or between EC vertexes ($\xi_{i,j}$ pairwise potentials) .

3.2 Potentials and Encoded Knowledge

Pairwise Syntactic Similarity Potentials. The pairwise syntactic similarity potentials are defined for each pair of verb instance vertexes, $x_i, x_j \in X$. They are designed to encourage the model to assign verb instances with similar fine-grained feature representations to the same frame ($l \in L$) and verb instances with dissimilar representations to different frames. For this aim, for every verb pair i, j with feature representation vectors v_i, v_j and verb instance vertexes $x_i, x_j \in X$, we define the following potential function:

$$\theta_{i,j}(x_i = l_1, x_j = l_2) = \left\{ \begin{array}{ll} \lambda(v_i, v_j) & \text{if } l_1 = l_2 \\ 0 & \text{otherwise} \end{array} \right\}$$

Where $l_1, l_2 \in L$ are label pairs and λ is a verb instance similarity function. Below we describe the feature representation and the λ function.

The verb instance feature representation is defined through the following process. For each

word instance in the input sentences we first build a basic feature representation (see below). Then, for each verb instance we construct a final feature representation defined to be the concatenation of that verb's basic feature representation with the basic representations of the words in a size 2 window around the represented verb. The final feature representation for the i-th verb instance in our dataset is therefore defined to be $v_i = [w_{-2}, w_{-1}, vb_i, w_{+1}, w_{+2}]$, where w_{-k} and w_{+k} are the basic feature representations of the words in distance $-k$ or $+k$ from the i-th verb instance in its sentence, and vb_i is the basic feature representation of that verb instance.

Our basic feature representation is inspired from the feature representation of the MST parser (McDonald et al., 2005) except that in the parser the features represent a directed edge in the complete directed graph defined over the words in a sentence that is to be parsed, while our features are generated for word n-grams. Particularly, our feature set is a concatenation of two sets derived from the MST set described in Table 1 of (McDonald et al., 2005) in the following way: (1) In both sets the parent word in the parser's set is replaced with the represented word; (2) In one set every child word in the parser's set is replaced by the word to the left of the represented word and in the other set it is replaced by the word to its right. This choice of features allows us to take advantage of a provably useful syntactic feature representation without the application of any parse tree annotation or parser.

We compute the similarity between the syntactic environments of two verb instances, i, j, using the following equation:

$$\lambda(v_i, v_j) = W \cdot \cos(v_i, v_j) - S$$

Where W is a hyperparameter designed to bias verb instances of the same verb type towards the same frame. Practically, W was tuned to be 3 for instances of the same type, and 1 otherwise [4].

While the cosine function is the standard measure of similarity between two vectors, its values are in the $[0, 1]$ range. In the MRF modeling framework, however, we must encode a negative pairwise potential value between two vertexes in order to encourage the model to assign different labels (frames) to them. We therefore added the positive hyperparameter S which was tuned, without access to gold standard manual annotations, so that there is an even number of negative and positive pairwise syntactic similarity potentials after the model is pruned (see Section 3.3) [5].

Type Level Singleton Potentials. The goal of these potentials is to bias verb instances of the same type to be assigned to the same syntactic frame while still keeping the instance based nature of our algorithm. For this aim, we applied Algorithm 1 for pre-clustering of the verb instances and encoded the induced clusters into the local potentials of the corresponding $x \in X$ vertexes. For every $x \in X$ the singleton potential is therefore defined to be:

$$\theta_i(x_i = l) = \left\{ \begin{array}{ll} F \cdot \max \lambda & \text{if } l \text{ is induced by Algorithm 1} \\ 0 & \text{otherwise} \end{array} \right\}$$

where $\max \lambda$ is the maximum λ score across all verb instance pairs in the model and $F = 0.2$ is a hyperparamter.

Algorithm 1 has two hyperparameters: T and M, the first is a similarity cut-off value used to determine the initial set of clusters, while the second is used to determine whether two clusters are similar enough to be merged. We tuned these hyperparameters, without manually annotated data, so that the number of clusters induced by this algorithm will be equal to the number of gold standard SCFs. T was tuned so that the first part of the algorithm generates an excessive number of clusters, and M was then tuned so that these clusters are merged to the desired number of clusters.

The λ function, used to measure the similarity between two verbs, is designed to bias the instances of the same verb type to have a higher similarity score. Algorithm 1 therefore tends to assign such instances to the same cluster. In our experiments that was always the case for this algorithm.

High Cardinality Verb Sets Potentials. This set of potentials aims to bias larger sets of verb instances to share the same SCF. It is inspired by (Rush et al., 2012) who demonstrated, that syntactic structures that appear at the same syntactic context, in terms of the surrounding POS tags, tend to manifest similar syntactic behavior. While they demonstrated the usefulness of their method for dependency parsing and POS tagging, we implement it for higher level SCFs.

We identified syntactic contexts that imply similar SCFs for verb instances appearing inside them.

[4] All hyperparameters that require gold-standard annotation for tuning, were tuned using held-out data (Section 4).

[5] The values in practice are $S = 0.43$ for labour legislation and $S = 0.38$ for environment.

Algorithm 1 Verb instance pre-clustering algorithm. $\hat{\lambda}$ is the average λ score between the members of its cluster arguments. T and M are hyperparametes tuned without access to gold standard data.

Require: $K = \emptyset$
  for all $x \in X$ **do**
    for all $k \in K$ **do**
      for all $u \in k$ **do**
        if $\lambda(v_x, v_u) > T$ **then**
          $k = k \cup \{x\}$
          Go to next x
        end if
      end for
    end for
    $k1 = \{x\}$
    $K = K \cup k1$
  end for
  for all $k_1, k_2 \in K$: $k_1 \neq k_2$ **do**
    if $\hat{\lambda}(k_1, k_2) > M$ **then**
      Merge (k_1, k_2)
    end if
  end for

Contexts are characterized by the coarse POS tag to the left and to the right of the verb instance. While the number of context sets is bounded only by the number of frames our model is designed to induce, in practice we found that defining two equivalence sets led to the best performance gain, and the sets we used are presented in Table 1.

In order to encode this information into our MRF, each set of syntactic contexts is associated with an equivalence class (EC) vertex $c \in C$ and the verb instance vertexes of all verbs that appear in a context from that set are connected with an edge to c. The pairwise potential between a vertex $x \in X$ and its equivalence class is defined to be:

$$\phi_{i,j}(x_i = l_1, c_j = l_2) = \left\{ \begin{array}{ll} U & \text{if } l_1 = l_2 \\ 0 & \text{otherwise} \end{array} \right\}$$

$U = 10$ is a hyperparameter that strongly biases x vertexes to get the same SCF as their EC vertex.

3.3 Verb Cluster Induction

In this section we describe how we induce verb instance clusters from our model. This process is based on the following three steps: (1) Graph pruning; (2) Induction of an Ensemble of approximate MAP inference solutions in the resulted graphical model; and, (3) Induction of a final clustering solution based on the ensemble created at step 2. Below we explain the necessity of each of these steps and provide the algorithmic details.

EC-1		EC-2	
Left	Right	Left	Right
,	D	V	T
N	D	R	T
V	.	N	D
R	D	R	N

Table 1: POS contexts indicative for the syntactic frame of the verb instance they surround. D: *determiner*, N: *noun*, V: *verb*, T: the preposition *'to'* (which has its own POS tag in the WSJ POS tag set which we use), R: *adverb*. EC-1 and EC-2 stand for the first and second equivalence class respectively. In addition, the following contexts where associated with both ECs: $(T, D), (T, N), (N, N)$ and (V, I) where I stands for a preposition.

Graph Pruning. The edge set of our model consists of an edge for every pair of verb instance vertexes and of the edges that connect verb instance vertexes and equivalence class vertexes. This results in a large tree-width graph which substantially complicates MRF inference. To alleviate this we prune all edges with a positive score lower than p_+ and all edges with a negative score higher than p_-, where p_+ and p_- are manually tuned hyperparametes [6].

MAP Inference. For most reasonable values of p_+ and p_- our graph still contains cycles even after it is pruned, which makes inference NP-hard (Shimony, 1994). Yet, thanks to our choice of an edge-factorized model, there are various approximate inference algorithms suitable for our case.

We applied the message passing algorithm for linear-programming (LP) relaxation of the MAP assignment (MPLP, (Sontag et al., 2008)). LP relaxation algorithms for the MAP problem define an upper bound on the original objective which takes the form of a linear program. Consequently, a minimum of this upper bound can be found using standard LP solvers or, more efficiently, using specialized message passing algorithms (Yanover et al., 2006). The MPLP algorithm described in (Sontag et al., 2008) is appealing in that it iteratively computes tighter upper bounds on the MAP objective (for details see their paper).

Cluster Ensemble Generation and a Final Solution. As our MAP objective is non-convex,

[6] The values used in practice are $p_+ = 0.28, p_- = -0.17$ for the labour legislation dataset, and $p_+ = 0.25$, $p_- = -0.20$ for the environment set.

the convergent point of an optimization algorithm applied to it is highly sensitive to its initialization. To avoid convergence to arbitrary local maxima which may be of poor quality, we turn to a perturbation protocol where we repeatedly introduce random noise to the MRF's potential functions and then compute the approximate MAP solution of the resulted model using the MPLP algorithm. Noising was done by adding an ϵ term to the $lambda$ values described in section 3.2 [7]. This protocol results in a set of cluster (label) assignments for the involved verb instances, which we treat as an ensemble of experts from which a final, high quality, solution is to be induced.

The basic idea in ensemble learning is that if several experts independently cluster together two verb instances, our belief that these verbs belong in the same cluster should increase. (Reichart et al., 2012) implemented this idea through the k-way normalized cut clustering algorithm (Yu and Shi, 2003). Its input is an undirected graph $\hat{G} = (\hat{V}, \hat{E}, \hat{W})$ where $\hat{V}$ is the set of vertexes, $\hat{E}$ is the set of edges and $\hat{W}$ is a non-negative and symmetric edge weight matrix. To apply this model to our task, we construct the input graph $\hat{G}$ from the labelings (frame assignments) contained in the ensemble. The graph vertexes $\hat{V}$ correspond to the verb instances and the (i, j)-th entry of the matrix $\hat{W}$ is the number of ensemble members that assign the same label to the i-th and j-th verb instances.

For $A, B \subseteq \hat{V}$ define:

$$links(A, B) = \sum_{i \in A, j \in B} \hat{W}(i, j)$$

Using this definition, the normalized link ratio of A and B is defined to be:

$$NormLinkRatio(A, B) = \frac{links(A, B)}{links(A, \hat{V})}$$

The k-way normalized cut problem is to minimize the links that leave a cluster relative to the total weight of the cluster. Denote the set of clusterings of $\hat{V}$ that consist of k clusters by $\hat{C} = \{\hat{c_1}, \ldots \hat{c_t}\}$ and the j-th cluster of the i-th clustering by $\hat{c_{ij}}$. Then

$$c^* = \underset{\hat{c_i} \in \hat{C}}{\operatorname{argmin}} \sum_{j=1}^{k} NormLinkRatio(\hat{c_{ij}}, \hat{V} - \hat{c_{ij}})$$

The algorithm of (Yu and Shi, 2003) solves this problem very efficiently as it avoids the heavy eigenvalues and eigenvectors computations required by traditional approaches.

4 Experiments and Results

Our model is unique compared to existing systems in two respects. First, it does not utilize supervision in the form of either a supervised syntactic parser and/or manually crafted SCF rules. Consequently, it induces unnamed frames (clusters) that are not directly comparable to the named frames induced by previous systems. Second, it induces syntactic frames at the verb instance, rather than type, level. Evaluation, and especially comparison to previous work, is therefore challenging.

We therefore evaluate our system in two ways. First, we compare its output, as well as the output of a number of clustering baselines, to the gold standard annotation of corpora from two different domains (the only publicly available ones with instance level SCF annotation, to the best of our knowledge). Second, in order to compare the output of our system to a rule-based SCF system that utilizes a supervised syntactic parser, we turn to a task-based evaluation. We aim to predict the degree of similarity between verb pairs and, following (Pado and Lapata, 2007) , we do so using a *syntactic-based* vector space model (VSM). We construct three VSMs - (a) one that derives features from our clusters; (b) one whose features come from the output of a state-of-the-art verb type level, rule based, SCF system (Reichart and Korhonen, 2013) that uses a modern parser (Klein and Manning, 2003); and (c) a standard lexical VSM. Below we show that our system compares favorably in both evaluations.

Data. We experimented with two datasets taken from different domains: labor legislation and environment (Quochi et al., 2012). These datasets were created through web crawling followed by domain filtering. Each sentence in both datasets may contain multiple verbs but only one target verb has been manually annotated with a SCF. The labour legislation domain dataset contains 4415 annotated verb instances (and hence also

[7] ϵ was accepted by first sampling a number in the $[0, 1]$ range using the Java psuodorandom generator and then scaling it to 1% of $\cos(v_i, v_j)$. This value was tuned, without access to gold standard manual annotations, so that there is an even number of negative and positive pairwise syntactic similarity potentials after the model is pruned (Section 3.3).

sentences) of 117 types, and the environmental domain dataset contains 4503 annotated verb instances of 116 types. In both datasets no verb type accounts for more than 4% of the instances and only up to 35 verb types account for 1% of the instances or more. The lexical difference between the corpora is substantial: they share only 42 annotated verb types in total, of which only 2 verb types (responsible for 4.1% and 5.2% of the instances in the environment and labor legislation domains respectively) belong to the 20 most frequent types (responsible for 37.9% and 46.85% of the verb instances in the respective domains) of each corpus.

The 29 members of the SCF inventory are detailed in (Quochi et al., 2012). Table 2, presenting the distribution of the 5 highest frequency frames in each corpus, demonstrates that, in addition to the significant lexical difference, the corpora differ to some extent in their syntactic properties. This is reflected by the substantially different frequencies of the "dobj:iobj-prep:su" and "dobj:su" frames.

As a pre-processing step we first POS tagged the datasets with the Stanford tagger (Toutanova et al., 2003) trained on the standard POS training sections of the WSJ PennTreebank corpus.

4.1 Evaluation Against SCF Gold Standard

Experimental Protocol The computational complexity of our algorithm does not allow us to run it on thousands of verb instances in a feasible time. We therefore repeatedly sampled 5% of the sentences from each dataset, ran our algorithm as well as the baselines (see below) and report the average performance of each method. The number of repetitions was 40 and samples were drawn from a uniform distribution while still promising that the distribution of gold standard SCFs in each sample is identical to their distribution in the entire dataset. Before running this protocol, 5% of each corpus was kept as held-out data on which hyperparameter tuning was performed.

Evaluation Measures and Baselines. We compare our system's output to instance-level gold standard annotation. We use standard measures for clustering evaluation, one measure from each of the two leading measure types: the V measure (Rosenberg and Hirschberg, 2007), which is an information theoretic measure, and greedy many-to-one accuracy, which is a mapping-based measure. For the latter, each induced cluster is first mapped to the gold SCF frame that annotates the highest number of verb instances this induced cluster also annotates and then a standard instance-level accuracy score is computed (see, e.g., (Reichart and Rappoport, 2009)). Both measures scale from 100 (perfect match with gold standard) to 0 (no match).

As mentioned above, comparing the performance of our system with respect to a gold standard to the performance of previous type-level systems that used hand-crafted rules and/or supervised syntactic parsers would be challenging. We therefore compare our model to the following baselines: (a) The *most frequent class (MFC)* baseline which assigns all verb instances with the SCF that is the most frequent one in the *gold standard annotation* of the data; (b) The *Random* baseline which simply assigns every verb instance with a randomly selected SCF; (c) Algorithm 1 of section 3.2 which generates unsupervised verb instance clustering such that verb instances of the same type are assigned to the same cluster; and (d) Finally, we also compare our model against versions where everything is kept fixed, except a subset of potentials which is omitted. This enables us to study the intricacies of our model and the relative importance of its components. For all models, the number of induced clusters is equal to the number of SCFs in the gold standard.

Results Table 3 presents the results, demonstrating that our full model substantially outperforms all baselines. For the first two simple heuristic baselines (*MFC* and *Random*) the margin is higher than 20% for both the greedy M-1 mapping measure and the V measure. Note tat the V score of the MFC baseline is 0 by definition, as it assigns all items to the same cluster. The poor performance of these simple baselines is an indication of the difficulty of our task.

Recall that the type level clustering induced by Algorithm 1 is the main source of type level information our model utilizes (through its singleton potentials). The comparison to the output of this algorithm (the *Type Pre-clustering* baseline) therefore shows the quality of the instance level refinement our model provides. As seen in table 3, our model outperforms this baseline by 6.9% for the M-1 measure and 5.2% for the V measure.

In order to compare our model to its components we exclude either the EC potentials (ϕ and ξ) only (Model - EC), or the EC and the singleton potentials (θ_i, Model - EC - Type pre-clustering). The results show that our model gains much more

Environment		Labour Legislation	
SCF	Frequency	SCF	Frequency
dobj:su	46%	dobj:su	39%
su	9%	dobj:iobj-prep:su	15%
iobj-prep:su	8%	su	10%
dobj:iobj-prep:su	6%	su:xcompto-vbare	8%
su:xcompto-vbare	6%	iobj-prep:su	7%

Table 2: Top 5 most frequent SCFs for the Environment and Labour Legislation datasets used in our experiments.

	Environment		Labour Legislation	
	M-1	V	M-1	V
Full Model	**66.4**	**57.3**	**69.2**	**55.6**
Baselines				
MFC	46.2	0	39.4	0
Random	34.6	28.1	36.5	27.8
Type Pre-clustering	60.1	52.1	62.3	51.4
Model Components				
Model - EC	64.9	56.2	67.4	54.6
Model - EC - Type pre-clustering	48.3	48.9	45.7	44.7

Table 3: Results for our full model, the baselines (*Type Pre-clustering*: the pre-clustering algorithm (Algorithm 1 of section 3.2), *MFC*: the most frequent class (SCF) in the gold standard annotation and *Random*: random SCF assignment) and the model components. The full model outperforms all other models across measures and datasets.

from the type level information encoded through the singleton potentials than from the EC potentials. Yet, EC potentials do lead to an improvement of up to 1.5% in M-1 and up to 1.1% in V and are therefore responsible for up to 26.1% and 21.2% of the improvement over the type pre-clustering baseline in terms of M-1 and V, respectively.

4.2 Task Based Evaluation

We next evaluate our model in the context of vector space modeling for verb similarity prediction (Turney and Pantel, 2010). Since most previous word similarity works used noun datasets, we constructed a new verb pair dataset, following the protocol used in the collection of the wordSimilarity-353 dataset (Finkelstein et al., 2002).

Our dataset consists of 143 verb pairs, constructed from 122 unique verb lemma types. The participating verbs appear ≥ 10 times in the concatenation of the labour legislation and the environment datasets. Only pairs of verbs that were considered at least remotely similar by human judges (independent of those that provided the similarity scores) were included. A similarity score between 1 and 10 was assigned to each pair by 10 native English speaking annotators and were then averaged in order to get a unique pair score.

Our first baseline is a standard VSM based on lexical collocations. In this model features correspond to the number of collocations inside a size 2 window of the represented verb with each of the 5000 most frequent nouns in the Google n-gram corpus (Goldberg and Orwant, 2013). Since our corpora are limited in size, we use the collocation counts from the Google corpus.

We used our model to generate a vector representation of each verb in the following way. We run the model 5000 times, each time over a set of verbs consisting of one instance of each of the 122 verb types participating in the verb similarity set. The output of each such run is transformed to a binary vector for each participating verb, where all coordinates are assigned the value of 0, except from the one that corresponds to the cluster to which the verb was assigned which has the value of 1. The final vector representation is a concatenation of the 5000 binary vectors. Note that for this task we did not use the graph cut algorithm to generate a final clustering from the multiple MRF

runs. Instead we concatenated the output of all these runs into one feature representation that facilitates similarity prediction. For our model we estimated the verb pair similarity using the Tanimato similarity score for binary vectors:

$$T(X,Y) = \frac{\sum_i X_i \wedge Y_i}{\sum_i x_i \vee Y_i}$$

For the baseline model, where the features are collocation counts, we used the standard cosine similarity.

Our second baseline is identical to our model, except that: (a) the data is parsed with the Stanford parser (version 3.3.0, (Klein and Manning, 2003)) which was trained with sections 2-21 of the WSJ corpus; (b) the phrase structure output of the parser is transformed to the CoNLL dependency format using the official CoNLL 2007 conversion script (Johansson and Nugues, 2007); and then (c) the SCF of each verb instance is inferred using the rule-based system used by (Reichart and Korhonen, 2013). The vector space representation for each verb is then created using the process we described for our model and the same holds for vector comparison. This baseline allows direct comparison of frames induced by our SCF model with those derived from a supervised parser's output.

We computed the Pearson correlation between the scores of each of the models and the human scores. The results demonstrate the superiority of our model in predicting verb similarity: the correlation of our model with the human scores is 0.642 while the correlation of the lexical collocation baseline is 0.522 and that of the supervised parser baseline is only 0.266. The results indicate that in addition to their good alignment with SCFs, our clusters are also highly useful for verb meaning representation. This is in line with the verb clustering theory of the Levin tradition (Levin, 1993) which ties verb meaning with their syntactic properties. We consider this an intriguing direction of future work.

5 Conclusions

We presented an MRF-based unsupervised model for SCF acquisition which produces verb instance level SCFs as output. As opposed to previous systems for the task, our model uses only a POS tagger, avoiding the need for a statistical parser or manually crafted rules. The model is particularly valuable for NLP tasks benefiting from SCFs that are applied across text domains, and for the many tasks that involve sentence-level processing.

Our results show that the accuracy of the model is promising, both when compared against gold standard annotations and when evaluated in the context of a task. In the future we intend to improve our model by encoding additional information in it. We will also adapt it to a multilingual setup, aiming to model a wide range of languages.

Acknowledgments

The first author is supported by the Commonwealth Scholarship Commission (CSC) and the Cambridge Trust.

References

Ivana Romina Altamirano and Laura Alonso i Alemany. 2010. IRASubcat, a highly customizable, language independent tool for the acquisition of verbal subcategorization information from corpus. In *Proceedings of the NAACL 2010 Workshop on Computational Approaches to Languages of the Americas*.

Abhishek Arun and Frank Keller. 2005. Lexicalization in crosslinguistic probabilistic parsing: The case of french. In *Proceedings of ACL-05*.

Taylor Berg-Kirkpatrick, Alexander Bouchard-Cote, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of NAACL-HLT-10*.

Akshar Bharati, Sriram Venkatapathy, and Prashanth Reddy. 2005. Inferring semantic roles using subcategorization frames and maximum entropy model. In *Proceedings of CoNLL-05*.

Ted Briscoe and John Carroll. 1997. Automatic extraction of subcategorization from corpora. In *Proceedings of ANLP-97*.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the rasp system. In *Proceedings of ACL-COLING-06*.

John Carroll and Alex Fang. 2004. The automatic acquisition of verb subcategorisations and their impact on the performance of an HPSG parser. In *Proceedings of IJCNLP-04*.

Paula Chesley and Susanne Salmon-Alt. 2006. Automatic extraction of subcategorization frames for french. In *Proceedings of LREC-06*.

Kostadin Cholakov and Gertjan van Noord. 2010. Using unknown word techniques to learn known words. In *Proceedings of EMNLP-10*.

Shay Cohen and Noah Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of NAACL-HLT-09*.

Marie-Catherine De-Marneffe, Bill Maccartney, and Christopher Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC-06*.

Lukasz Debowski. 2009. Valence extraction using EM selection and co-occurrence matrices. *Proceedins of LREC-09*.

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

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. In *Proceedings of (*SEM)-13*. Association for Computational Linguistics.

Jan Hajič, Martin mejrek, Bonnie Dorr, Yuan Ding, Jason Eisner, Daniel Gildea, Terry Koo, Kristen Parton, Gerald Penn, Dragomir Radev, and Owen Rambow. 2002. Natural language generation in the context of machine translation. Technical report, Center for Language and Speech Processing, Johns Hopkins University, Baltimore. Summer Workshop Final Report.

Chung hye Han, Benoit Lavoie, Martha Palmer, Owen Rambow, Richard Kittredge, Tanya Korelsky, and Myunghee Kim. 2000. Handling structural divergences and recovering dropped arguments in a korean/english machine translation system. In *Proceedings of the AMTA-00*.

Dino Ienco, Serena Villata, and Cristina Bosco. 2008. Automatic extraction of subcategorization frames for italian. In *Proceedings of LREC-08*.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for english. In *Proceedings of NODALIDA-07*.

Daisuke Kawahara and Sadao Kurohashi. 2010. Acquiring reliable predicate-argument structures from raw corpora for case frame compilation. In *Proceedings of LREC-10*.

Dan Klein and Christopher Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL-03*.

Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. The MIT Press.

Anna Korhonen, Genevieve Gorrell, and Diana McCarthy. 2000. Statistical filtering and subcategorization frame acquisition. In *Proceedings of EMNLP-00*.

Anna Korhonen. 2002. Semantically motivated subcategorization acquisition. In *Proceedings of the ACL-02 workshop on Unsupervised lexical acquisition*.

Joel Lang and Mirella Lapata. 2011a. Unsupervised semantic role induction via split-merge clustering. In *Proceedings of ACL-11*.

Joel Lang and Mirella Lapata. 2011b. Unsupervised semantic role induction with graph partitioning. In *Proceedings of EMNLP-11*.

Matthew Lease and Eugene Charniak. 2005. Parsing biomedical literature. In *Proceedings of IJCNLP-05*.

Alessandro Lenci, Barbara Mcgillivray, Simonetta Montemagni, and Vito Pirrelli. 2008. Unsupervised acquisition of verb subcategorization frames from shallow-parsed corpora. In *Proceedings of LREC-08*.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. Chicago, IL.

Tom Lippincott, Anna Korhonen, and Diarmuid Oseaghdha. 2010. Exploring subdomain variation in biomedical language. *BMC Bioinformatics*.

Tom Lippincott, Aanna Korhonen, and Diarmuid Oseaghdha. 2012. Learning syntactic verb frames using graphical models. In *Proceedings of ACL-12*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL-05*.

Cedric Messiant, Anna Korhonen, and Thierry Poibeau. 2008. LexSchem: A large subcategorization lexicon for French verbs. In *Proceedings of LREC-08*.

Cedric Messiant. 2008. A subcategorization acquistion system for french verbs. In *Proceedings of ACL08-SRW*.

Yusuke Miyao and Junichi Tsujii. 2005. Probabilistic disambiguaton models for wide-coverage hpsg parsing. In *Proceedings of ACL-05*.

Alessandro Moschitti and Roberto Basili. 2005. Verb subcategorization kernels for automatic semantic labeling. In *Proceedings of the ACL-SIGLEX Workshop on Deep Lexical Acquisition*.

Ruth O'Donovan, Michael Burke, Aoife Cahill, Josef van Genabith, and Andy Way. 2005. Large-scale induction and evaluation of lexical resources from the penn-ii and penn-iii treebanks. *Computational Linguistics*, 31:328–365.

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

Judita Preiss, Ted Briscoe, and Anna Korhonen. 2007. A system for large-scale acquisition of verbal, nominal and adjectival subcategorization frames from corpora. In *Proceedings of ACL-07*.

Valeria Quochi, Francesca Frontini, Roberto Bartolini, Olivier Hamon, Marc Poch, Muntsa Padr, Nuria Bel, Gregor Thurmair, Antonio Toral, and Amir Kamram. 2012. Third evaluation report. evaluation of panacea v3 and produced resources. Technical report.

Roi Reichart and Anna Korhonen. 2013. Improved lexical acquisition through dpp-based verb clustering. In *Proceedings of ACL-13*.

Roi Reichart and Ari Rappoport. 2009. The nvi clustering evaluation measure. In *Proceedings of CoNLL-09*.

Roi Reichart, Gal Elidan, and Ari Rappoport. 2012. A diverse dirichlet process ensemble for unsupervised induction of syntactic categories. In *Proceedings of COLING-12*.

Laura Rimell, Thomas Lippincott, Karin Verspoor, Helen Johnson, and Anna Korhonen. 2013. Acquisition and evaluation of verb subcategorization resources for biomedicine. *Journal of Biomedical Informatics*, 46:228–237.

Eric Ringger, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin Seppi, and Deryle Lonsdale. 2007. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proceedings of the ACL-07 Linguistic Annotation Workshop*.

Douglas Roland and Daniel Jurafsky. 1998. subcategorization frequencies are affected by corpus choice. In *Proceedings of ACL-98*.

Andrew Rosenberg and Julia Hirschberg. 2007. V measure: a conditional entropybased external cluster evaluation measure. In *Proceedings of EMNLP-07*.

Alexander Rush, Roi Reichart, Michael Collins, and Amir Globerson. 2012. Improved parsing and pos tagging using inter-sentence consistency constraints. In *Proceedings of EMNLP-12*.

Sabine Schulte im Walde. 2006. Experiments on the automatic induction of german semantic verb classes. *Computational Linguistics*, 32(2):159–194.

Solomon Shimony. 1994. Finding the maps for belief networks is np-hard. *Artificial Intelligence*, 68:399–310.

David Sontag, Talya Meltzer, Amir Globerson, Tommi Jaakkola, and Yair Weiss. 2008. Tightening lp relaxations for map using message passing. In *Proceedings of UAI-08*.

Lin Sun and Anna Korhonen. 2011. Hierarchical verb clustering using graph factorization. In *Proceedings of EMNLP-11*.

Ivan Titvo and Alexandre Klementiev. 2012. A bayesian approach to unsupervised semantic role induction. In *Proceedings of EMNLP-12*.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of NAACL-03*.

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

Tim Van de Cruys, Laura Rimell, Thierry Poibeau, and Anna Korhonen. 2012. Multi-way tensor factorization for unsupervised lexical acquisition. In *Proceedings of COLING-12*.

Giulia Venturi, Simonetta Montemagni, Simone Marchi, Yutaka Sasaki, Paul Thompson, John McNaught, and Sophia Ananiadou. 2009. Bootstrapping a verb lexicon for biomedical information extraction. *Computational Linguistics and Intelligent Text Processing*, 5449:137–148.

Marion Weller, Alexander Fraser, and Sabine Schulte im Walde. 2013. Using subcategorization knowledge to improve case prediction for translation to german. In *Proceedings of ACL-13*.

Chen Yanover, Talya Meltzer, and Yair Weiss. 2006. Linear programming relazations and belief propogataion an empitical study. *JMLR Special Issue on Machine Learning and Large Scale Optimization*.

Stella Yu and Jianbo Shi. 2003. Multiclass spectral clustering. In *Proceedings of ICCV-13*.

Parsing low-resource languages using Gibbs sampling for PCFGs with latent annotations

Liang Sun[1] **Jason Mielens**[2] **Jason Baldridge**[2]

[1]Department of Mechanical Engineering
The University of Texas at Austin
sally722@utexas.edu

[2]Department of Linguistics
The University of Texas at Austin
{jmielens,jbaldrid}@utexas.edu

Abstract

PCFGs with latent annotations have been shown to be a very effective model for phrase structure parsing. We present a Bayesian model and algorithms based on a Gibbs sampler for parsing with a grammar with latent annotations. For PCFG-LA, we present an additional Gibbs sampler algorithm to learn annotations from training data, which are parse trees with coarse (unannotated) symbols. We show that a Gibbs sampling technique is capable of parsing sentences in a wide variety of languages and producing results that are on-par with or surpass previous approaches. Our results for Kinyarwanda and Malagasy in particular demonstrate that low-resource language parsing can benefit substantially from a Bayesian approach.

1 Introduction

Despite great progress over the past two decades on parsing, relatively little work has considered the problem of creating accurate parsers for low-resource languages. Existing work in this area focuses primarily on approaches that use some form of cross-lingual bootstrapping to improve performance. For instance, Hwa et al. (2005) use a parallel Chinese/English corpus and an English dependency grammar to induce an annotated Chinese corpus in order to train a Chinese dependency grammar. Kuhn (2004b) also considers the benefits of using multiple languages to induce a monolingual grammar, making use of a measure for data reliability in order to weight training data based on confidence of annotation. Bootstrapping approaches such as these achieve markedly improved results, but they are dependent on the existence of a parallel bilingual corpus. Very few such corpora are readily available, particularly for low-resource languages, and creating such corpora obviously presents a challenge for many practical applications. Kuhn (2004a) shows some of the difficulty in handling low-resource languages by examining various tasks using Q'anjob'al as an example. Another approach is that of Bender et al. (2002), who take a more linguistically-motivated approach by making use of linguistic universals to seed newly developed grammars. This substantially reduces the effort by making it unnecessary to learn the basic parameters of a language, but it lacks the robustness of grammars learned from data.

Recent work on Probabilistic Context-Free Grammars with latent annotations (PCFG-LA) (Matsuzaki et al., 2005; Petrov et al., 2006) have shown them to be effective models for syntactic parsing, especially when less training material is available (Liang et al., 2009; Shindo et al., 2012). The coarse nonterminal symbols found in vanilla PCFGs are refined by latent variables; these latent annotations can model subtypes of grammar symbols that result in better grammars and enable better estimates of grammar productions. In this paper, we provide a Gibbs sampler for learning PCFG-LA models and show its effectiveness for parsing low-resource languages such as Malagasy and Kinyawanda.

Previous PCFG-LA work focuses on the problem of parameter estimation, including expectation-maximization (EM) (Matsuzaki et al., 2005; Petrov et al., 2006), spectral learning (Cohen et al., 2012; Cohen et al., 2013), and variational inference (Liang et al., 2009; Wang and Blunsom, 2013). Regardless of inference method, previous work has used the same method to parse new sentences: a Viterbi parse under a new sentence-specific PCFG obtained from an approximation of the original grammar (Matsuzaki et al., 2005). Here, we provide an alternative approach to parsing new sentences: an extension of the Gibbs sampling algorithm of Johnson et al. (2007), which learns rule probabilities in an unsupervised PCFG.

We use a Gibbs sampler to collect sampled trees theoretically distributed from the true posterior distribution in order to parse. Priors in a Bayesian model can control the sparsity of grammars (which the inside-outside algorithm fails to do), while naturally incorporating smoothing into the model (Johnson et al., 2007; Liang et al., 2009). We also build a Bayesian model for parsing with a treebank, and incorporate information from training data as a prior. Moreover, we extend the Gibbs sampler to learn and parse PCFGs with latent annotations. Learning the latent annotations is a compute-intensive process. We show how a small amount of training data can be used to bootstrap: after running a large number of sampling iterations on a small set, the resulting parameters are used to seed a smaller number of iterations on the full training data.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 290–300,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

This allows us to employ more latent annotations while maintaining reasonable training times and still making full use of the available training data.

To determine the cross-linguistic applicability of these methods, we evaluate on a wide variety of languages with varying amounts of available training data. We use English and Chinese as examples of languages with high data availability, while Italian, Malagasy, and Kinyarwanda provide examples of languages with little available data.

We find that our technique comes near state of the art results on large datasets, such as those for Chinese and English, and it provides excellent results on limited datasets – both artificially limited in the case of English, and naturally limited in the case of Italian, Malagasy, and Kinyarwanda. This, combined with its ability to run off-the-shelf on new languages without any supporting materials such as parallel corpora, make it a valuable technique for the parsing of low-resource languages.

2 Gibbs sampling for PCFGs

Our starting point is a Gibbs Sampling algorithm for vanilla PCFGs introduced by Johnson et al. (2007) for estimating rule probabilities in an unsupervised PCFG. We focus instead on using this algorithm for parsing new sentences and then extending it to learn PCFGs with latent annotations. We begin by summarizing the Bayesian PCFG and Gibbs sampler defined by Johnson et al. (2007).

Bayesian PCFG For a grammar G, each rule r in the set of rules R has an associated probability θ_r. The probabilities for all the rules that expand the same nonterminal A must sum to one: $\sum_{A\to\beta\in R}\theta_{A\to\beta}=1$.

Given an input corpus $\boldsymbol{w}$=($\boldsymbol{w^{(1)}},\cdots,\boldsymbol{w^{(n)}}$), we introduce a latent variable $\boldsymbol{t}$=($t^{(1)},\cdots,t^{(n)}$) for trees generated by G for each sentence. The joint posterior distribution of $\boldsymbol{t}$ and θ conditioned on $\boldsymbol{w}$ is:

$$\begin{aligned} p(\boldsymbol{t},\theta \mid \boldsymbol{w}) &\propto p(\theta)p(\boldsymbol{w}\mid\boldsymbol{t})p(\boldsymbol{t}\mid\theta) \\ &= p(\theta)(\prod_{i=1}^{n} p(w^{(i)}\mid t^{(i)})p(t^{(i)}\mid\theta)) \\ &= p(\theta)(\prod_{i=1}^{n} p(w^{(i)}\mid t^{(i)})\prod_{r\in R}\theta_r^{f_r(t^{(i)})}) \quad (1)\end{aligned}$$

Here $f_r(t)$ is the number of occurrences of rule r in the derivation of $\boldsymbol{t}$; $p(w^{(i)}\mid t^{(i)})=1$ if the yield of $t^{(i)}$ is the sequence $w^{(i)}$, and 0 otherwise.

We use a Dirichlet distribution parametrized by α_A: $Dir(\alpha_A)$ as the prior of the probability distribution for all rules expanding non-terminal A ($p(\theta_A)$). The prior for all θ, $p(\theta)$, is the product of all Dirichlet distributions over all non-terminals $A\in N$: $p(\theta\mid\alpha)=\prod_{A\in N}p(\theta_A\mid\alpha_A)$.

Since the Dirichlet distribution is conjugate to the Multinomial distribution, which we use to model the likelihood of trees, the conditional posterior of θ_A can be updated as follows:

$$\begin{aligned} p_G(\theta\mid\boldsymbol{t},\alpha) &\propto p_G(\boldsymbol{t}\mid\theta)p(\theta\mid\alpha) \\ &\propto (\prod_{r\in R}\theta_r^{f_r(\boldsymbol{t})})(\prod_{r\in R}\theta_r^{\alpha_r-1}) \\ &= \prod_{r\in R}\theta_r^{f_r(\boldsymbol{t})+\alpha_r-1} \quad (2)\end{aligned}$$

which is still a Dirichlet distribution with updated parameter $f_r(\boldsymbol{t})+\alpha_r$ for each rule $r\in R$.

Gibbs sampler The parameters of the PCFG model can be learned from an annotated corpus by simply counting rules. However, parsing cannot be done directly with standard CKY as with standard PCFGs, so we use the Gibbs sampling algorithm presented in Johnson et al. (2007). An additional motivation for using this algorithm is that Johnson et al. use it for learning without annotated structures, and in future work we seek to learn from fewer, and at times partial, annotations.

An advantage of using Gibbs sampling for Bayesian inference, as opposed to other approximation algorithms such as Variational Bayesian inference (VB) and Collapsed Variational Bayesian inference (CVB), is that Markov Chain Monte Carlo (MCMC) algorithms are guaranteed to converge to a sample from the true posterior under appropriate conditions (Taddy, 2011). Both VB and CVB converge to inaccurate and locally optimal solutions, like EM. In some models, CVB can achieve more accurate results due to weaker assumptions (Wang and Blunsom, 2013). Another advantage of Gibbs sampling is that the sampler allows for parallel computation by allowing each sentence to be sampled entirely independently of the others. After each parallel sampling stage, all model parameters are updated in a single step, and the process then repeats (see §2).

To sample the joint posterior $p(\boldsymbol{t},\theta\mid\boldsymbol{w})$, we sample production probabilities θ and then trees $\boldsymbol{t}$ from these conditional distributions:

$$p(\boldsymbol{t}\mid\theta,\boldsymbol{w},\alpha)=\prod_{i=1}^{n}p(t_i\mid w_i,\theta) \quad (3)$$

$$p(\theta\mid\boldsymbol{t},\boldsymbol{w},\alpha)=\prod_{A\in N}Dir(\theta_A\mid f_A(\boldsymbol{t})+\alpha_A) \quad (4)$$

Step 1: Sample Rule Probabilities. Given trees $\boldsymbol{t}$ and prior α, the production probabilities θ_A for each nonterminal $A\in N$ are sampled from a Dirichlet distribution with parameters $f_A(\boldsymbol{t})+\alpha_A$. $f_A(\boldsymbol{t})$ is a vector, and each component of $f_A(\boldsymbol{t})$, is the number of occurrences of one rule expanding nonterminal A.

Step 2: Sample Tree Structures. To sample trees from $p(t_i\mid\boldsymbol{w_i},\theta)$, we use the efficient sampling scheme used in previous work (Goodman, 1998; Finkel et al., 2006; Johnson et al., 2007). There are two parts to this algorithm. The first constructs an inside table as in the Inside-Outside algorithm for PCFGs (Lary and Young, 1990). The second selects the tree by recursively sampling productions from top to bottom.

Require: A is parent node of binary rule; $w_{i,k}$ is a span of words: $i+1<k$
function TREESAMPLER(A, i, k)
for $i < j < k$ and pair of child nodes of A:B, C **do**
$P(j, B, C) = \frac{\theta_{A\to BC}\cdot p_{B,i,j}\cdot p_{C,j,k}}{\cdot} p_{A,i,k}$
end for
Sample $j*, B*, C*$ from multinomial distribution for (j, B, C) with probabilities calculated above
return $j*, B*, C*$
end function

Algorithm 1: Sampling split position and rule to expand parent node

Consider a sentence $\boldsymbol{w}$, with sub-spans $w_{i,k} = (w_{i+1}, \cdots, w_k)$. Given $\boldsymbol{\theta}$, we construct the inside table with entries $p_{A,i,k}$ for each nonterminal and each word span $w_{i,k} : 0 \le i < k \le l$, where $p_{A,i,k} = P_{G_A}(w_{i,k}|\theta)$ is the probability that words i through k were produced by the non-terminal A. The table is computed recursively by

$$p_{A,k-1,k} = \theta_{A\to w_k} \quad (5)$$

$$p_{A,i,k} = \sum_{A\to BC\in R}\sum_{i<j<k} \theta_{A\to BC}\cdot p_{B,i,j}\cdot p_{C,j,k} \quad (6)$$

for all $A, B, C \in N$ and $0 \le i < j < k \le l$.

The resulting inside probabilities are then used to generate trees from the distribution of all valid trees of the sentence. The tree is generated from top to bottom recursively with the function $TreeSampler$ defined in Algorithm 1.

In unsupervised PCFG learning, the rule probabilities can be resampled using the sampled trees, then used to reparse the corpus, and so on. We use this property to refine latent annotations for the PCFG-LA model described in the next section.

3 PCFG with latent annotations

When labeled trees are available, rule frequencies can be directly extracted and used as priors for a PCFG. However, when learning PCFG-LAs, we must learn the fine-grained rules from the coarse trees, so we extend the Gibbs sampler to assign latent annotations to unannotated trees. The resulting learned PCFG-LA parser outputs samples of annotated trees so that we can obtain unannotated trees after marginalizing.

3.1 Model

With the PCFG-LA model (Matsuzaki et al., 2005; Petrov et al., 2006) fine-grained CFG rules are automatically induced from training, effectively providing a form of feature engineering without human intervention. Given $H = \{1, \cdots, K\}$, a set of latent annotation symbols, and $x \in H$:

- $\theta_{A[x]\to U}$ is the probability of rule $A[x] \to U$, where $U \in N\times N \cup T$. The probabilities for all rules that expand the same annotated non-terminal must sum to one.
- $\beta_{A[x],B,C\to y,z}$ is the probability of assigning latent annotation y, z to child nodes B, C of $A[x]$. $\sum_{y,z\in H\times H}\beta_{A[x],B,C\to y,z} = 1$.

The inputs to the PCFG-LA are a CFG G with finite number of latent annotations for each non-terminal, an initial guess of probabilities of grammar rule θ_0, and a prior α_θ is learned from training.

The joint posterior distribution of $\boldsymbol{t}$ and θ, β conditioned on $\boldsymbol{w}$ is:

$$\begin{aligned} p(\boldsymbol{t}, \theta, \beta \mid \boldsymbol{w}) &\propto p(\theta, \beta)p(\boldsymbol{w} \mid \boldsymbol{t})p(\boldsymbol{t} \mid \theta, \beta) \\ &= p(\theta)p(\beta)(\prod_{i=1}^{n} p(w_i \mid t_i)p(t_i \mid \theta, \beta)) \end{aligned} \quad (7)$$

We assume that θ and β are independent to get $P(\theta, \beta) = P(\theta)P(\beta)$.

To learn parameters θ, β, we use a Dirichlet distribution as a prior for both θ and β. The distribution for all rules expanding $A[x]$ is:

$$P(\theta \mid \alpha^\theta) = \prod_{A\in N, x\in H} P(\theta_{A[x]} \mid \alpha^\theta_{A[x]}) \quad (8)$$

The distribution for latent annotations associated with child nodes of $A[x] \to BC$ is:

$$P(\beta \mid \alpha^\beta) = \prod_{y,z\in H\times H} P(\beta_{A[x],B,C} \mid \alpha^\beta_{A[x],B,C}). \quad (9)$$

With this setting, the conditional posterior of $\theta_{A[x]}$ and $\beta_{A[x],B,C}$ can be updated, as in §2. For all unary and binary rules r expanding $A[x]$:

$$\theta_{A[x]} \mid \boldsymbol{t}, \alpha^\theta \sim Dir(f_r(\boldsymbol{t}) + \alpha^\theta_r) \quad (10)$$

Here, $f_r(\boldsymbol{t})$ is the number of occurrence of annotated rule r in $\boldsymbol{t}$. Also, for combination of latent annotations $y, z \in H \times H$ assigned to B, C in rule $A[x] \to B, C$:

$$\beta_{A[x],B,C} \mid \boldsymbol{t}, \alpha^\beta \sim Dir(f_d(\boldsymbol{t}) + \alpha^\beta_d) \quad (11)$$

Here, $f_d(\boldsymbol{t})$ is the number of occurrences of combination d in $\boldsymbol{t}$.

3.2 Learning PCFG-LAs from raw text

To learn from raw text, we extend the sampler in §2 to PCFG-LA. Given priors $\alpha^\theta, \alpha^\beta$ and raw text, the algorithm alternates between two steps. The first samples trees for the entire corpus; the second samples θ and β from Dirichlet distributions with updated parameters, combining priors and counts from sampled trees. The algorithm then alternates between these steps until convergence. The outputs are samples of θ, β and annotated trees.

The parsing process is specified in Algorithm 2. The first step assigns a tree to a sentence, say $w_{0,l}$. We first

Require: $w_1, \cdots, w_n$ are raw sentences; θ_0, β_0 are initial values; $\alpha^\theta, \alpha^\beta$ are priors; M is the number of iterations
function PARSE($w_1, .., w_n, \theta_0, \beta_0, \alpha^\theta, \alpha^\beta, M$)
for iteration $i = 1$ to M **do**
for sentence $s = 1$ to n **do**
Calculate Inside Table
Sample tree nodes and associated latent annotations, get tree structure $t_s^{(i)}$
end for
Sample $\theta^{(i)}$, $\beta^{(i)}$
end for
for sentence $s = 1$ to n **do**
Marginalize the latent annotations to get unannotated trees $T_s^{(1)}, \cdots, T_s^{(M)}$
Find the mode of $T_s^{(1)}, \cdots, T_s^{(M)}$: T_s
end for
return $T_1, \cdots, T_n$
end function

Algorithm 2: Parsing new sentences

construct an inside table (see §2). Each entry in the table stores the probability that a word span is produced by a given annotated nonterminal. For root node S, with θ, β and inside table $p_{A[x],i,k}$, we sample one annotation based on all $p_{S[x],0,l}$, $x \in H$. Assume that we sampled x for S, we further sample a rule to expand $S[x]$ and possible splits of the span $w_{0,l}$ jointly. Assume that we sampled nonterminals B, C to expand $S[x]$, where B is responsible for $w_{0,j}$ and C is responsible for $w_{j,l}$. We further sample annotations for B, C together, say y, z. Then we sample rules and split positions to expand $B[y]$ and $C[z]$, and continue until reaching the terminals.

This algorithm alone could be used for unsupervised learning of PCFG-LA if we input a non-informed or weakly-informed prior α^θ and α^β. With access to unannotated trees for training, we only need to assign latent annotations to them and then use the frequencies of these annotated rules as the prior when parsing. The details of training when trees are available are illustrated in §3.3.

Once we have trees (with latent annotations), the step of sampling θ and β from a Dirichlet distribution is direct. We need to count the number of occurrences $f_r(t)$ for each rule r like $A[x] \to U, U \in N \times N \cup T$ in updated annotated trees $\boldsymbol{t}$, and draw $\theta_{A[x]}$ from the updated Dirichlet distribution $Dir(f_{A[x]}(\boldsymbol{t}) + \alpha^\theta_{A[x]})$. We also need to count the number of occurrences of $f_d(t)$ for each combination of $yz \in H \times H$ assigned to B, C given $A[x] \to B, C$ in $\boldsymbol{t}$, and draw $\beta_{A[x],B,C}$ from the updated Dirichlet distribution $Dir(f_{A[x],B,C}(\boldsymbol{t}) + \alpha^\beta_{A[x],B,C})$ similarly.

To parse a sentence, we first calculate the inside table and then sample the tree.

Calculate the inside table. Given $\boldsymbol{\theta}, \boldsymbol{\beta}$ and a string $\boldsymbol{w}$=$w_{0,l}$, we construct a table with entries $p_{A[x],i,k}$ for each $A \in N$, $x \in H$ and $0 \le i < k \le l$, where $p_{A[x],i,k} = P_{G_{A[x]}}(w_{i,k}|\theta, \beta)$ is the probability that words i through k were produced by the annotated nonterminal $A[x]$. The table can be computed recursively, for all $A \in N$, $x \in H$, by

$$p_{A[x],k-1,k} = \theta_{A[x] \to w_k} \tag{12}$$

$$p_{A[x],i,k} = \sum_{A[x] \to BC : BC \in N \times N} \sum_{j : i<j<k} \sum_{yz \in H \times H} \theta_{A[x] \to BC} \beta_{A[x]BC \to yz} p_{B[y],i,j} p_{C[z],j,k} \tag{13}$$

Sample the tree, top to bottom. First, from start symbol S, sample latent annotation from multinomial with probability $\pi_{S[x]} p_{S[x],0,l}$ for each $x \in H$. Next, given annotated non-terminal $A[x]$ and i, k, sample possible child nodes and split positions from multinomial with probability:

$$p(B, C, j) = \frac{1}{p_{A[x],i,k}} \cdot \sum_{y,z \in H} \theta_{A[x] \to BC} \beta_{A[x]BC \to yz} p_{B[y],i,j} p_{C[z],j,k} \tag{14}$$

Here the probability is calculated by marginalizing all possible latent annotations for B, C, and $\theta_{A[x] \to BC} \beta_{A[x]BC \to yz}$ is the probability of choosing $B[y], C[z]$ to expand $A[x]$, and $p_{B[y],i,j} p_{C[z],j,k}$ are the probabilities for $B[y]$ and $C[z]$ to be responsible for word span $w_{i,j}$ and $w_{j,k}$ respectively. And $p_{A[x],i,k}$ is the normalizing term.

Third, given $A[x], B, C, i, j, k$, sample annotations for B, C from multinomial with probability:

$$p(y, z) = \frac{\beta_{A[x]BC \to yz} p_{B[y],i,j} p_{C[z],j,k}}{\sum_{y,z} \beta_{A[x]BC \to yz} p_{B[y],i,j} p_{C[z],j,k}} \tag{15}$$

A crucial aspect of this procedure is that all trees can be sampled independently. This parallel process produces a substantial speed gain that is important particularly when using more latent annotations. After all trees have been sampled (independently), the counts from each individual tree are combined prior to the next sampling iteration.

3.3 Learning from coarse training trees

In training, we need to learn the probabilities of fine-grained rules given coarsely-labeled trees. We perform Gibbs sampling on the training data by first iteratively sampling probabilities and then assigning annotations to tree nodes. We use the average counts of annotated production rules from sampled trees to produce the prior α^θ and α^β incorporated into parsing raw sentences.

We first index the non-terminal nodes of each tree T by $1, 2, \cdots$ from top to bottom, and left to right. Then the sampler iterates between two steps. The first samples θ, β given annotated trees (as in §3.2). The second samples latent annotations for nonterminal nodes

Require: $T_1, \cdots, T_n$ are fully parsed trees; θ_0, β_0 are initial values; $\alpha^{\theta_0}, \alpha^{\beta_0}$ are priors; M is the number of iterations
function ANNO($T_1, \cdots, T_n, \theta_0, \beta_0, \alpha^{\theta_0}, \alpha^{\beta_0}, M$)
for iteration $i = 1$ to M **do**
for sentence $s = 1$ to n **do**
Calculate inside probability
Sample latent annotations for each node in the tree, get tree with latent annotations $t_s^{(i)}$
end for
Sample $\theta^{(i)}$, $\beta^{(i)}$
end for
return Mean of number of occurrences of production rules and associated latent annotations from all sampled annotated trees
end function

Algorithm 3: Learning prior from training

in parsed trees, which also takes two steps. The first step is to, for each node in the tree, calculate and store the probability that the node is annotated by x. The second step is to jointly sample latent annotations for child nodes of root nodes, and then continue this process from top to bottom until reaching the pre-terminal nodes.

Step one: inside probabilities. Given tree T, compute $b_T^i[x]$ for each non-terminal i recursively:

1. If node N_i is a pre-terminal node above terminal symbol w, then for $x \in H$

$$b_T^i[x] = \theta_{N_i[x] \to w} \tag{16}$$

2. Otherwise, let j, k be two child nodes of i, then for $x \in H$

$$b_T^i[x] = \sum_{y,z \in H} \theta_{N_i[x] \to N_j N_k} \beta_{N_i[x] N_j N_k \to y,z} b_T^j[y] b_T^k[z] \tag{17}$$

Step two: outside sampling. Given inside probability $b_T^i[x]$ for every non-terminal i and all latent annotations $x \in H$, we sample the latent annotations from top to bottom:

1. If node i is the root node ($i = 1$), then sample $x \in H$ from a multinomial distribution with $f_T^i[x] = \pi(N_i[x])$.

2. For a parent node with sampled latent annotation $N_i[x]$ with children N_j, N_k, sample latent annotations for these two nodes from a multinomial distribution with

$$f_T^i[y,z] = \frac{1}{b_T^i[x]} \cdot \theta_{N_i[x] \to N_j N_k} \beta_{N_i[x] N_j N_k \to y,z} b_T^j[y] b_T^k[z] \tag{18}$$

After training, we take the average counts of sampled annotated rules and combinations of latent annotations as priors to parse raw sentences.

4 Experiments[1]

Our goal is to understand parsing efficacy using sampling and latent annotations for low-resource languages, so we perform experiments on five languages with varying amount of training data. We compare our results to a number of previously established baselines. First, for all languages, we use both a standard unsmoothed PCFG and the Bikel parser, trained on the training corpus. Additionally, we compare to state-of-the-art results for both English and Chinese, which have an existing body of work in PCFGs using a Bayesian framework. For Chinese, we compare to Huang & Harper (2009), using their results that only use the Chinese Treebank (CTB). For English, we compare to Liang et al. (2009). Prior results for parsing the constituency version of the Italian data are available from Alicante et al. (2012), but as they make use of a different version of the treebank including extra sentences, and additionally use the extensive functional tags present in the corpus, we do not directly compare our results to theirs.[2]

4.1 Data

English (ENG) and Chinese (CHI) are the two main languages used for this work; they are commonly used in parser evaluation and have previous examples of statistical parsers using a Bayesian framework. And since we primarily are interested in parsing low-resource languages, we include results for Kinyarwanda (KIN) and Malagasy (MLG) as examples of languages without substantial existing treebanks. Finally, as a middle-ground language, we use Italian (ITL).

For English, we use the Wall-Street Journal section of the Penn Treebank (WSJ) (Marcus et al., 1993). The data split is sections 02-21 for training, section 22 for development, and section 23 for testing. For Chinese, the Chinese Treebank (CTB5) (Xue et al., 2005) was used. The data split is files 81-899 for training, files 41-80 for development, and files 1-40/900-931 for testing.

The ITL data is from the Turin University Treebank (TUT) (Bosco et al., 2000) and consists of 2,860 Italian sentences from a variety of domains. It was split into training, development, and test sets with a 70/15/15 percentage split.

The KIN texts are transcripts of testimonies by survivors of the Rwandan genocide provided by the Kigali Genocide Memorial Center, along with a few BBC news articles. The MLG texts are articles from the websites Lakroa and La Gazette and Malagasy Global Voices. Both datasets are described in Garrette and Baldridge (2013). The KIN and MLG data is very small compared to ENG and CHI: the KIN dataset con-

[1] Code available at github.com/jmielens/gibbs-pcfg-2014, along with instructions for replicating experiments when possible

[2] As part of a standardized pre-processing step, we strip functional tags, which makes a direct comparison to their results inappropriate.

tains 677 sentences, while the MLG dataset has only 113. Also, we simulated a small training set for ENG data by using only section 02 of the WSJ for training.

4.2 Experimental Setup

As a preprocessing step, all trees are converted into Chomsky Normal-Form such that all non-terminal productions are binary and all unary chains are removed.

Additional standard normalization is performed. Functional tags (e.g. the *SBJ* part of NP-SBJ), empty nodes (traces), and indices are removed. Our binarization is simple: given a parent, select the rightmost child as the head and add a stand-in node that contains the remainder of the original children; the process then recurses. This simple technique uses no explicit head-finding rules, which eases cross-linguistic applicability.

From this normalized data, we train latent PCFGs with K=1,2,4,8,16,32 (where K=1 is equivalent to the plain PCFG described in section 2).

4.3 Practical refinements

Unknown word handling. We use a similar unknown word handling procedure to Liang et al. (2009). From our raw corpus we extract features associated with every word, these features include surrounding context words as well as substring suffix/prefix features. Using these features we produce fifty clusters using k-means. Then, as a pre-parsing step, we replace all words occurring less than five times with their cluster label - this simulates unknown words for training. Finally, during evaluation, any word not seen in training was also replaced with its corresponding cluster label. This final step is simple because there are no 'unknown unknowns' in our corpus, as the clustering has been performed over the entire corpus prior to training. This approach is similar to methods for unsupervised POS-tag induction that also utilize clusters in this manner (Dasgupta & Ng, 2007).

We compare this unknown word handling method to one in which the clustering and a classifier is trained not on the corpus under consideration, but rather on a separate corpus of unrelated data. This comparison was made to understand the effects of including the evaluation set in the training data (without labels) versus training on out-of-domain texts. This is a more realistic measurement of out-of-the-box performance of a trained model.

Jump-starting sampling. In the basic setup, training high K-value models takes a prohibitively long time, so we also consider a jump-start technique that allows larger annotation values (such as K=16) to be run in less time. We train these high-K value models first on a highly reduced training set (5% of the full training set) for a large number of iterations, and then use the found θ values as the starting point for training on the full training set for a small number of iterations. Although many of the estimated parameters on the reduced set will be zero, the prior allows us to eventually recover this information in the full set. This allows us to train on the full training set, which is desirable relative to training on a reduced set, while still allowing the model sufficient iterations to burn in. The fact that we are likely starting in a fairly good position within the search space (due to estimating θ from the corpus) also likely helps enable these lower iteration counts.

System	K=1	K=2	K=4	K=8	K=16
Unsmoothed PCFG	40.2	—	—	—	—
Bikel Parser	57.9	—	—	—	—
Liang et al. 07	60.5	71.1	77.2	**79.2**	78.2
Berkeley Parser	60.8	**74.4**	**78.4**	79.1	**78.7**
Gibbs PCFG	**61.0**	71.3	76.6	78.7	78.0

Table 1: F1 scores for small English training data experiments. 'K' is the number of latent annotations – K=1 represents a vanilla, unannotated PCFG.

5 Results

We start with Tables 1 and 2, which show performance when training on section 02 of the WSJ (pretending English is a "low-resource" language). The results show that the basic Gibbs PCFG (where K=1), with an F-score of 61.0, substantially outperforms not only an unsmoothed PCFG (the simplest baseline), but also the Bikel parser (Bikel, 2004b) trained on the same amount of data. Table 1 also shows further large gains are obtained from using latent annotations—from 60.5 for K=1 to 78.7 for K=8.

The Gibbs PCFG also compares quite favorably to the PCFG-LA of Liang et al. (2009)—slightly better for K=1 and K=2 and slightly worse for K=4 and K=8. Table 2 shows that the Gibbs PCFG is able to produce results with a smaller amount of variance relative to the Berkeley Parser, even at low training sizes. This trend is repeated in Table 3, which shows that the Gibbs PCFG also produces less variance when training on different single sections of the WSJ relative to the Berkeley Parser, although it again produces slightly lower F1 scores.

We also use the small English corpus to determine the effects of weighting the prior when sampling annotations, varying α between 0.1 and 10.0. Though performance is not sensitive to varying α for larger corpora, Figure 1 shows it can make a substantial difference for smaller corpora (with an optimal value was obtained with an α value of 5 for this small training set). This seems to indicate that the lower counts associated with the smaller training sets should be compensated for by weighting those counts more heavily when processing the evaluation set, as we had anticipated.

System	WSJ Sec. 02	KIN	MLG
Berkeley Parser	**78.3** $\pm$ **0.93**	60.6 $\pm$ 1.1	52.2 $\pm$ 2.0
Gibbs PCFG	76.7 $\pm$ 0.63	**67.2** $\pm$ **0.92**	**57.5** $\pm$ **1.1**

Table 2: F1 scores with standard deviation over ten runs of small training data, K=4.

System	F1 / StDev
Berkeley Parser	**77.5** ± **2.1**
Gibbs PCFG	77.0 ± 1.4

Table 3: F1 scores with standard deviations over twenty runs, training on individual WSJ sections (02-21).

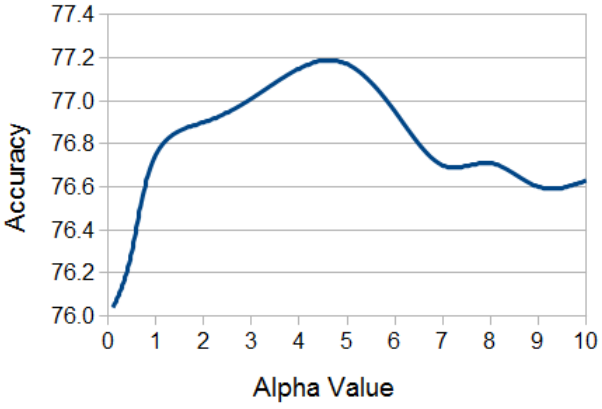


Figure 1: Accuracy by varying α levels for small English data.

To evaluate the effectiveness of the jump-start technique, we ran the full ENG data set with K=4 to compare the results from the full training setup to jump-starting. For this, we performed 100 training iterations on the reduced training set (WSJ section 02) and then used the resulting θ values to seed training on the full training set. Those training runs varied between three and nine iterations, and the results are shown in Figure 2. The full ENG K=4 F-score is 86.2, so these results represent a slight step back. Nonetheless, the technique is still valuable in that it allows for inferring latent annotations for higher K-values than would typically be available to us in a reasonable timeframe.

Table 4 shows the results for the main experiments. Sampling a vanilla PCFG (K=1) produces results that are not state-of-the-art, but still good overall and always better than an unsmoothed PCFG. The benefits of the latent annotations are further shown in the increase

Condition	ENG	CHI	ITA	KIN	MLG
Unsmoothed PCFG	69.9	66.8	62.1	45.9	49.2
Liang et al. 07	87.1	—	—	—	—
Huang & Harper09	—	**84.1**	—	—	—
Bikel Parser	86.9	81.1	**74.5**	55.7	49.5
Berkeley Parser	**90.1**	83.4	71.6	61.4	51.8
Gibbs PCFG,K=1	79.3	75.4	66.3	58.5	55.1
Gibbs PCFG,K=2	82.6	79.8	69.3	65.0	57.0
Gibbs PCFG,K=4	86.0	82.3	71.9	67.2	57.8
Gibbs PCFG,K=16	87.2	83.2	72.4	**68.1**	**58.2**
Gibbs PCFG,K=32	87.4	83.4	71.0	66.8	55.3

Table 4: F1 scores for experiments on sampled PCFGs. Note that Wang and Blunsom (2013) obtain an ENG F-score of 77.9% using collapsed VB for K=2. Though they do not give exact numbers, their Fig. 7 indicates an F-score of about 87% for K=16.

Figure 2: F-Score for K=4, varying full-set training iterations (with and without 100x jump start).

of F1 score in all languages, as compared to the vanilla PCFG. Experiments were run up to K=32 primarily due to time constraint. Although previous literature results report increases up to the equivalent of K=64, it may be the case that higher K values with no merge step more easily lead to overfitting in our model – reducing the effectiveness of those high values, as shown by the overall poorer performance on several languages at K=32 when compared to K=16 as well as the general levelling-off seen at the high K values.

For English and Chinese, the previous Bayesian framework parsers outperform our own, but only by around two points. Additionally, our parsing of Chinese improves on the Bikel parser (trained on our training data) despite the fact that the Bikel parser makes use of language specific optimizations. Our parser needs no changes to switch languages.

The Gibbs PCFG with K=16 is superior to the strong Bikel and Berkeley Parser benchmarks for both KIN and MLG, a promising result for future work on parsing low-resource languages in general. Note also that our parser exhibits less variance than Berkeley Parser especially for KIN and MLG, which supports the fact that the variance of Berkeley Parser is higher for models with few subcategories (Petrov et al., 2006).

Examples of the improvement across latent annotations for a given tree are shown in Figure 3. The details of the noun phrase 'no major bond offerings' were the same for each tree, and are thus abstracted here. The low K-value tree (K=2) is shown in 3a, and primarily suffers from issues related to the prepositional phrase, 'in Europe friday'. In particular, the low K-value tree incorrectly groups 'Europe friday' as a noun phrase object of 'in'.

The higher K-value tree (K=8) is shown in 3b. This tree manages to correctly analyze the prepositional phrase, accurately separately the temporal locative 'Friday' from the actual prepositional phrase of 'in Europe'. However, the high K-value tree makes a

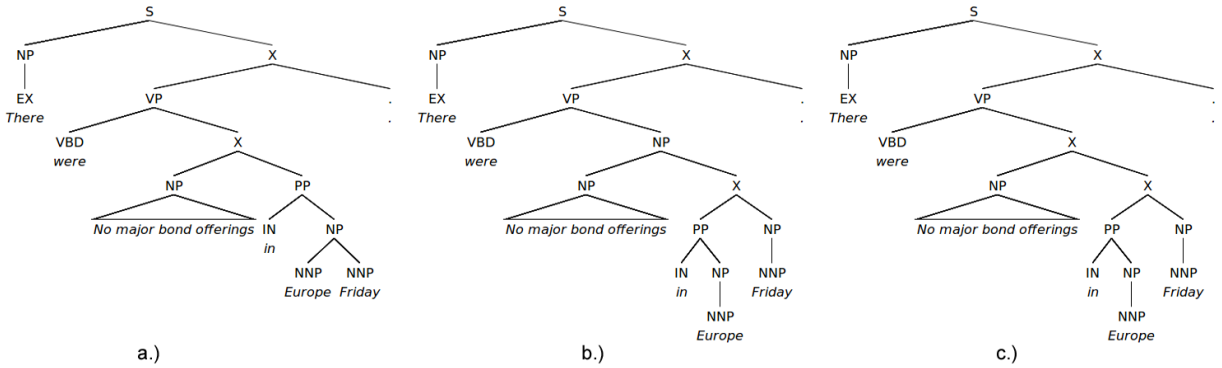


Figure 3: Examples of tree progression in the Gibbs PCFG with a) K=2, b) K=8, and c) gold tree.

different mistake that the low K-value tree did not; it groups 'no major bond offerings in Europe Friday' as a noun phrase, when it should be three separate phrases (two noun phrases and a prepositional phrase). This error may be related to the additional latent annotations. With more available noun phrase subtypes, it may be the case that a more unusual noun phrase could be permitted that would have been too low probability with only a few subtypes.

To determine whether the substantial range in F1 scores across languages are primarily the result of the much larger training corpora available for certain languages, two extreme training set reduction experiments were conducted. The training sets for all languages were reduced to a total of either 100 or 500 sentences. This process was repeated 10 times in a cross-validation setup, where 10 separate sets of sentences were selected for each language. The results of these experiments are shown in Table 5.

We conclude that while data availability is a major factor in the higher performance of English and Chinese in our original experiments, it is not the only issue. Clearly, either the linguistic facts of particular languages or perhaps choices of formalism and annotation conventions in the corpora make some of the languages more difficult to parse than others. The primary questions is why Gibbs-PCFG is able to achieve higher relative performance on the KIN/MLG datasets when compared to the other parsers, and why this advantage does not necessarily transfer to the extreme small-scale versions of the ENG/CHI/ITL data. Preliminary investigation into the properties of the corpora have revealed a number of potential answers. For instance, the POS tagsets for KIN/MLG are substantially reduced compared to the other corpora, and there are differences in the branching factor of the native versions of the corpora as well: a typical maximum branching factor for a tree in ENG/CHI/ITL is around 4-5, while for KIN/MLG it is almost always 2 (natively binary). Branching factors above 5 essentially never occur in KIN/MLG, while they are not rare in ENG/CHI/ITL. The question of exactly why the Gibbs-PCFG seems to perform well on these corpora remains an open question, but these differences could provide a starting point for future analysis.

Condition	In-Domain	Out-of-Domain
Full English (K=4)	**86.0**	83.3
Small English (K=4)	**76.6**	75.7
Kinyarwanda (K=4)	**67.2**	65.1
Malagasy (K=4)	**57.8**	55.4

Table 6: Effect of differing regimes for handling unknown words.

In addition to the actual F1 scores, the relative uniformity of the standard deviation results indicates that the individual parsers are not that much different in terms of their ability to provide consistent results at these small data extremes, as opposed to the slightly higher training levels where the Gibbs-PCFG generated smaller variances.

Considering the effects of unknown word handling, Table 6 shows that using the evaluation set when creating the unknown word classifier does improve overall parsing accuracy when compared to an unknown word handler that is trained on out-of-domain texts. This shows that results reported in previous work somewhat overstate the accuracy of these parsers when used in the wild—which matters greatly in the low-resource setting.

6 Conclusion

Our experiments demonstrate that sampling vanilla PCFGs, as well as PCFGs with latent annotations, is feasible with the use of a Gibbs sampler technique and produces results that are in line with previous parsers on controlled test sets. Our results also show that our methods are effective on a wide variety of languages—including two low-resource languages—with no language-specific model modifications needed.

Additionally, although not a uniform winner, the Gibbs-PCFG shows a propensity for performing well on naturally small corpora (here, KIN/MLG). The exact reason for this remains slightly unclear, but the fact that a similar advantage is not found for extremely small versions of large corpora indicates that our approach may be particularly well-suited for application in real low-resource environments as opposed to a sim-

Parser	Size	ENG	CHI	ITL	KIN	MLG
Bikel	100	54.7 ± 2.2	51.4 ± 3.0	51 ± 2.4	47.1 ± 2.3	44.4 ± 2.0
Berkeley	100	55.2 ± 2.6	53.9 ± 2.9	50 ± 2.8	47.8 ± 2.1	44.5 ± 2.3
Gibbs-PCFG	100	54.5 ± 2.0	51.7 ± 2.4	49.5 ± 3.6	50.3 ± 2.3	45.8 ± 1.8
Bikel	500	56.2 ± 2.0	54.1 ± 2.7	54.2 ± 2.4	—	—
Berkeley	500	58.9 ± 2.2	56.4 ± 2.7	52.5 ± 2.7	—	—
Gibbs-PCFG	500	58.1 ± 2.0	55.7 ± 2.3	51.1 ± 3.2	—	—

Table 5: 100/500 sentence training set results, including st.dev over 10 runs. KIN/MLG did not have enough data to run the 500 sentence version.

ulated environment.

Having established this procedure and its relative tolerance for low amounts of data, we would like to extend the model to make use of partial bracketing information instead of complete trees, perhaps in the form of Fragmentary Unlabeled Dependency Grammar annotations (Schneider et al., 2013). This would allow the sampling procedure to potentially operate using corpora with lighter annotations than full trees, making initial annotation effort not quite as heavy and potentially increasing the amount of available data for low-resource languages. Additionally, using the expert partial annotations to help restrict the sample space could provide good gains in terms of training time.

Acknowledgments

Supported by the U.S. Army Research Office under grant number W911NF-10-1-0533. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the U.S. Army Research Office.

References

Anita Alicante, Cristina Bosco, Anna Corazza, and Alberto Lavelli. 2012. A treebank-based study on the influence of Italian word order on parsing performance. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uur Doan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of LREC'12*, Istanbul, Turkey. European Language Resources Association (ELRA).

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2002. The Grammar Matrix: An Open-Source Starter-Kit for the Rapid Development of Cross-Linguistically Consistent Broad-Coverage Precision Grammars. In John Carroll, Nelleke Oostdijk, and Richard Sutcliffe, editors, *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan.

Dan Bikel. 2004a. *On The Parameter Space of Generative Lexicalized Statistical Parsing Models*. Ph.D. thesis, University of Pennsylvania.

Daniel M Bikel. 2004b. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Ezra Black, Fred Jelinek, John Lafferty, David M Magerman, Robert Mercer, and Salim Roukos. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the workshop on Speech and Natural Language*, pages 134–139. Association for Computational Linguistics.

Taylor L Booth and Richard A Thompson. 1973. Applying probability measures to abstract languages. *Computers, IEEE Transactions on*, 100(5):442–450.

Cristina Bosco, Vincenzo Lombardo, Daniela Vassallo, and Leonardo Lesmo. 2000. Building a Treebank for Italian: a Data-driven Annotation Schema. In *In Proceedings of the Second International Conference on Language Resources and Evaluation LREC-2000 (pp. 99*, pages 99–105.

Glenn Carroll and Eugene Charniak. 1992. *Two experiments on learning probabilistic dependency grammars from corpora*. Department of Computer Science, Univ.

Eugene Charniak. 1996. Tree-bank grammars. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1031–1036.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

Noam Chomsky. 1956. Three models for the description of language. *Information Theory, IRE Transactions on*, 2(3):113–124.

Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. 2012. Spectral learning of latent-variable PCFGs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 223–231. Association for Computational Linguistics.

Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL-HLT*, pages 148–157.

Michael John Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 184–191. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23. Association for Computational Linguistics.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637.

Jenny Rose Finkel, Christopher D Manning, and Andrew Y Ng. 2006. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626. Association for Computational Linguistics.

Dan Garrette and Jason Baldridge. 2013. Learning a Part-of-Speech Tagger from Two Hours of Annotation. In *Proceedings of NAACL*, Atlanta, Georgia.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-World Semi-Supervised Learning of POS-Taggers for Low-Resource Languages. In *Proceedings of the 51th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741.

Joshua T Goodman. 1998. *Parsing Inside-Out*. Ph.D. thesis, Harvard University Cambridge, Massachusetts.

Zhongqiang Huang and Mary Harper. 2009. Self-Training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 832–841. Association for Computational Linguistics.

Rebecca Hwa, Philip Resnik, and Amy Weinberg. Breaking the Resource Bottleneck for Multilingual Parsing. In *The Proceedings of the Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*. Conference on Language Resources and Evaluation.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov Chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Jonas Kuhn. 2004a. Applying computational linguistic techniques in a documentary project for Qanjobal (Mayan, Guatemala). In *In Proceedings of LREC 2004*. Citeseer.

Jonas Kuhn. 2004b. Experiments in parallel-text based grammar induction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 470. Association for Computational Linguistics.

Karim Lary and Steve J Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algrithm. *Computer, Speech and Language*, 4:35–56.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Probabilistic grammars and hierarchical Dirichlet processes. *The handbook of applied Bayesian analysis*.

David M Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 276–283. Association for Computational Linguistics.

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

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 75–82. Association for Computational Linguistics.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 128–135. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *HLT-NAACL*, pages 404–411.

Slav Petrov and Dan Klein. 2008. Sparse multi-scale grammars for discriminative latent variable parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 867–876. Association for Computational Linguistics.

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

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple Unsupervised Grammar Induction from Raw Text with Cascaded Finite State Models. In *ACL*, pages 1077–1086.

Nathan Schneider, Brendan O'Connor, Naomi Saphra, David Bamman, Manaal Faruqui, Noah A Smith, Chris Dyer, and Jason Baldridge. 2013. A framework for (under) specifying dependency syntax without overloading annotators. *arXiv preprint arXiv:1306.2091*.

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

Matthew A Taddy. 2011. On estimation and selection for topic models. *arXiv preprint arXiv:1109.4518*.

Pengyu Wang and Phil Blunsom. 2013. Collapsed Variational Bayesian Inference for PCFGs. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 173–182, Sofia, Bulgaria, August. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.*, 11(2):207–238, June.

Incremental Semantic Role Labeling with Tree Adjoining Grammar

Ioannis Konstas[⋆], **Frank Keller**[⋆], **Vera Demberg**[†] and **Mirella Lapata**[⋆]
⋆: Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
{ikonstas,keller,mlap}@inf.ed.ac.uk
†: Cluster of Excellence Multimodal Computing and Interaction,
Saarland University
vera@coli.uni-saarland.de

Abstract

We introduce the task of incremental semantic role labeling (iSRL), in which semantic roles are assigned to incomplete input (sentence prefixes). iSRL is the semantic equivalent of incremental parsing, and is useful for language modeling, sentence completion, machine translation, and psycholinguistic modeling. We propose an iSRL system that combines an incremental TAG parser with a semantically enriched lexicon, a role propagation algorithm, and a cascade of classifiers. Our approach achieves an SRL F-score of 78.38% on the standard CoNLL 2009 dataset. It substantially outperforms a strong baseline that combines gold-standard syntactic dependencies with heuristic role assignment, as well as a baseline based on Nivre's incremental dependency parser.

1 Introduction

Humans are able to assign semantic roles such as agent, patient, and theme to an incoming sentence before it is complete, i.e., they incrementally build up a partial semantic representation of a sentence prefix. As an example, consider:

(1) The athlete realized [her goals]$_{\text{PATIENT/THEME}}$ were out of reach.

When reaching the noun phrase *her goals*, the human language processor is faced with a semantic role ambiguity: *her goals* can either be the PATIENT of the verb *realize*, or it can be the THEME of a subsequent verb that has not been encountered yet. Experimental evidence shows that the human language processor initially prefers the PATIENT role, but switches its preference to the *theme* role when it reaches the subordinate verb *were*. Such semantic garden paths occur because human language processing occurs word-by-word, and are well attested in the psycholinguistic literature (e.g., Pickering et al., 2000).

Computational systems for performing semantic role labeling (SRL), on the other hand, proceed non-incrementally. They require the whole sentence (typically together with its complete syntactic structure) as input and assign all semantic roles at once. The reason for this is that most features used by current SRL systems are defined globally, and cannot be computed on sentence prefixes.

In this paper, we propose incremental SRL (iSRL) as a new computational task that mimics human semantic role assignment. The aim of an iSRL system is to determine semantic roles while the input unfolds: given a sentence prefix and its partial syntactic structure (typically generated by an incremental parser), we need to (a) identify which words in the input participate in the semantic roles as arguments and predicates (the task of **role identification**), and (b) assign correct semantic labels to these predicate/argument pairs (the task of **role labeling**). Performing these two tasks incrementally is substantially harder than doing it non-incrementally, as the processor needs to commit to a role assignment on the basis of incomplete syntactic and semantic information. As an example, take (1): on reaching *athlete*, the processor should assign this word the AGENT role, even though it has not seen the corresponding predicate yet. Similarly, upon reaching *realized*, the processor can complete the AGENT role, but it should also predict that this verb also has a PATIENT role, even though it has not yet encountered the argument that fills this role. A system that performs SRL in a fully incremental fashion therefore needs to be able to assign **incomplete semantic roles,** unlike existing full-sentence SRL models.

The uses of incremental SRL mirror the applications of incremental parsing: iSRL models can be used in language modeling to assign better string probabilities, in sentence completion systems to

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 301–312,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

provide semantically informed completions, in any real time application systems, such as dialog processing, and to incrementalize applications such as machine translation (e.g., in speech-to-speech MT). Crucially, any comprehensive model of human language understanding needs to combine an incremental parser with an incremental semantic processor (Padó et al., 2009; Keller, 2010).

The present work takes inspiration from the psycholinguistic modeling literature by proposing an iSRL system that is built on top of a cognitively motivated incremental parser, viz., the Psycholinguistically Motivated Tree Adjoining Grammar parser of Demberg et al. (2013). This parser includes a predictive component, i.e., it predicts syntactic structure for upcoming input during incremental processing. This makes PLTAG particularly suitable for iSRL, allowing it to predict incomplete semantic roles as the input string unfolds. Competing approaches, such as iSRL based on an incremental dependency parser, do not share this advantage, as we will discuss in Section 4.3.

2 Related Work

Most SRL systems to date conceptualize semantic role labeling as a supervised learning problem and rely on role-annotated data for model training. Existing models often implement a two-stage architecture in which role identification and role labeling are performed in sequence. Supervised methods deliver reasonably good performance with F-scores in the low eighties on standard test collections for English (Màrquez et al., 2008; Björkelund et al., 2009).

Current approaches rely primarily on syntactic features (such as path features) in order to identify and label roles. This has been a mixed blessing as the path from an argument to the predicate can be very informative but is often quite complicated, and depends on the syntactic formalism used. Many paths through the parse tree are likely to occur infrequently (or not at all), resulting in very sparse information for the classifier to learn from. Moreover, as we will discuss in Section 4.4, such path information is not always available when the input is processed incrementally. There is previous SRL work employing Tree Adjoining Grammar, albeit in a non-incremental setting, as a means to reduce the sparsity of syntax-based features. Liu and Sarkar (2007) extract a rich feature set from TAG derivations and demonstrate that this improves SRL performance.

In contrast to incremental parsing, incremental semantic role labeling is a novel task. Our model builds on an incremental Tree Adjoining Grammar parser (Demberg et al., 2013) which predicts the syntactic structure of upcoming input. This allows us to perform incremental parsing and incremental SRL in tandem, exploiting the predictive component of the parser to assign (potentially incomplete) semantic roles on a word-by-word basis. Similar to work on incremental parsing that evaluates incomplete trees (Sangati and Keller, 2013), we evaluate the incomplete semantic structures produced by our model.

3 Psycholinguistically Motivated TAG

Demberg et al. (2013) introduce Psycholinguistically Motivated Tree Adjoining Grammar (PLTAG), a grammar formalism that extends standard TAG (Joshi and Schabes, 1992) in order to enable incremental parsing. Standard TAG assumes a lexicon of **elementary trees**, each of which contains at least one lexical item as an **anchor** and at most one leaf node as a **foot node,** marked with $A*$. All other leaves are marked with $A\downarrow$ and are called **substitution nodes.** Elementary trees that contain a foot node are called **auxiliary trees**; those that do not are called **initial trees**. Examples for TAG elementary trees are given in Figure 1a–c.

To derive a TAG parse for a sentence, we start with the elementary tree of the head of the sentence and integrate the elementary trees of the other lexical items of the sentence using two operations: **adjunction** at an internal node and **substitution** at a substitution node (the node at which the operation applies is the **integration point**). Standard TAG derivations are not guaranteed to be incremental, as adjunction can happen anywhere in a sentence, possibly violating left-to-right processing order. PLTAG addresses this limitation by introducing **prediction trees,** elementary trees without a lexical anchor. These can be used to predict syntactic structure anchored by words that appear later in an incremental derivation. The use of prediction trees ensures that fully connected **prefix trees** can be built for every prefix of the input sentence.

Each node in a prediction tree carries **markers** to indicate that this node was predicted, rather than being anchored by the current sentence prefix. An example is Figure 1d, which contains a prediction tree with marker "1". In PLTAG, markers are eliminated through a new operation called **verification**, which matches them with the nodes

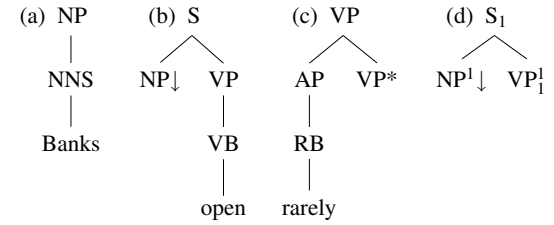


Figure 1: PLTAG lexicon entries: (a) and (b) initial trees, (c) auxiliary tree, (d) prediction tree.

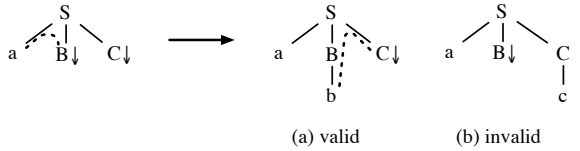


Figure 3: The current fringe (dashed line) indicates where valid substitutions can occur. Other substitutions result in an invalid prefix tree.

of non-predictive elementary trees. An example of a PLTAG derivation is given in Figure 2. In step 1, a prediction tree is introduced through substitution, which then allows the adjunction of an adverb in step 2. Step 3 involves the verification of the marker introduced by the prediction tree against the elementary tree for *open*.

In order to efficiently parse PLTAG, Demberg et al. (2013) introduce the concept of **fringes**. Fringes capture the fact that in an incremental derivation, a prefix tree can only be combined with an elementary tree at a limited set of nodes. For instance, the prefix tree in Figure 3 has two substitution nodes, for *B* and *C*. However, only substitution into *B* leads to a valid new prefix tree; if we substitute into *C*, we obtain the tree in Figure 3b, which is not a valid prefix tree (i.e., it represents a non-incremental derivation).

The parsing algorithm proposed by Demberg et al. (2013) exploits fringes to tabulate intermediate results. It manipulates a chart in which each cell (i, f) contains all the prefix trees whose first i leaves are the first i words and whose current fringe is f. To extend the prefix trees for i to the prefix trees for $i+1$, the algorithm retrieves all current fringes f such that the chart has entries in the cell (i, f). For each such fringe, it needs to determine the elementary trees in the lexicon that can be combined with f using substitution or adjunction. In spite of the large size of a typical TAG lexicon, this can be done efficiently, as it only requires matching the current fringes. For each match, the parser then computes the new prefix trees and its new current fringe f' and enters it into cell $(i+1, f')$.

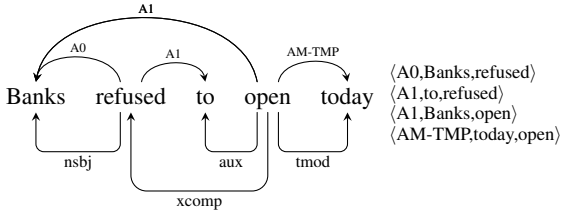


Figure 4: Syntactic dependency graph with semantic role annotation and the accompanying semantic triples, for *Banks refused to open today*.

Demberg et al. (2013) convert the Penn Treebank (Marcus et al., 1993) into TAG format by enriching it with head information and argument/modifier information from Propbank (Palmer et al., 2005). This makes it possible to decompose the Treebank trees into elementary trees as proposed by Xia et al. (2000). Prediction trees can be learned from the converted Treebank by calculating the connection path (Mazzei et al., 2007) at each word in a tree. Intuitively, a prediction tree for word w_n contains the structure that is necessary to connect w_n to the prefix tree $w_1 \ldots w_{n-1}$, but is not part of any of the elementary trees of $w_1 \ldots w_{n-1}$. Using this lexicon, a probabilistic model over PLTAG operations can be estimated following Chiang (2000).

4 Model

4.1 Problem Formulation

In a typical semantic role labeling scenario, the goal is to first identify words that are predicates in the sentence and then identify and label all the arguments for each predicate. This translates into spotting specific words in a sentence that represent the predicate's arguments, and assigning predefined semantic role labels to them. Note that in this work we focus on verb predicates only. The output of a semantic role labeler is a set of semantic dependency triples $\langle l, a, p \rangle$, with $l \in \mathcal{R}$, and $a, p \in \mathbf{w}$, where $\mathcal{R}$ is a set of semantic role labels denoting a specific relationship between a predicate and an argument (e.g., ARG0, ARG1, ARGM in Propbank), $\mathbf{w}$ is the list of words in the sentence, l denotes a specific role label, a the argument, and p the predicate. An example is shown in Figure 4.

As discussed in the introduction, standard semantic role labelers make their decisions based on evidence from the whole sentence. In contrast, our aim is to assign semantic roles incrementally, i.e.,

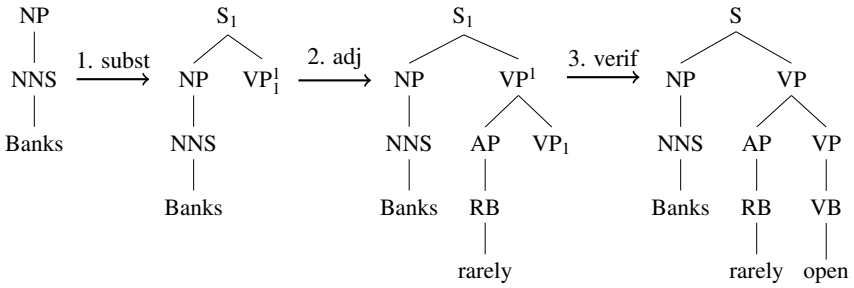


Figure 2: Incremental parse for *Banks rarely open* using the operations substitution (with a prediction tree), adjunction, and verification.

we want to produce a set of (potentially incomplete) semantic dependency triples for each prefix of the input sentence. Note that not every word is an argument to a predicate, therefore the set of triples will not necessarily change at every input word. Furthermore, the triples themselves may be incomplete, as either the predicate or the argument may not have been observed yet (**predicate-incomplete** or **argument-incomplete** triples).

Our iSRL system relies on PLTAG, using a semantically augmented lexicon. We parse an input sentence incrementally, applying a novel incremental role propagation algorithm (IRPA) that creates or updates existing semantic triple candidates whenever an elementary (or prediction) tree containing role information is attached to the existing prefix tree. As soon as a triple is completed we apply a two-stage classification process, that first identifies whether the predicate/argument pair is a good candidate, and then disambiguates role labels in case there is more than one candidate.

4.2 Semantic Role Lexicon

Recall that Propbank is used to construct the PLTAG treebank, in order to distinguish between arguments and modifiers, which result in elementary trees with substitution nodes, and auxiliary trees, i.e., trees with a foot node, respectively (see Figure 1). Conveniently, we can use the same information to also enrich the extracted lexicon with the semantic role annotations, following the process described by Sayeed and Demberg (2013).[1] For arguments, annotations are retained on the substitution node in the parental tree, while for modifiers, the role annotation is displayed on the foot node of the auxiliary tree. Note that we display role annotation on traces that are leaf nodes, which enables us to recover long-range dependencies (third and fifth tree in Figure 5a). Likewise, we annotate prediction trees with semantic roles, which enables our system to predict upcoming incomplete triples.

Our annotation procedure unavoidably introduces some role ambiguity, especially for frequently occurring trees. This can give rise to two problems when we generate semantic triples incrementally: IRPA tends to create many spurious candidate semantic triples for elementary trees that correspond to high frequency words (e.g., prepositions or modals). Secondly, a semantic triple may be identified correctly but is assigned several role labels. (See the elementary tree for *refuse* in Figure 5a.) We address these issues by applying classifiers for role label disambiguation at every parsing operation (substitution, adjunction, or verification), as detailed in Section 4.4.

4.3 Incremental Role Propagation Algorithm

The main idea behind IRPA is to create or update existing semantic triples as soon as there is available role information during parsing. Our algorithm (lines 1–6 in Algorithm 1) is applied after every PLTAG parsing operation, i.e., when an elementary or prediction tree $\mathcal{T}$ is adjoined to a particular integration point node π_{ip} of the prefix tree of the sentence, via substitution or adjunction (lines 3–4).[2] In case an elementary tree $\mathcal{T}_v$ verifies a prediction tree $\mathcal{T}_{pr}$ (lines 5–6), the same methodology applies, the only difference being that we have to tackle multiple integration point nodes $\mathcal{T}_{pr,ip}$, one for each prediction marker of $\mathcal{T}_{pr}$ that matches the corresponding nodes in $\mathcal{T}_v$.

For simplicity of presentation, we will use a concrete example, see Figure 5. Figure 5a shows the lexicon entries for the words of the sentence

[1] Contrary to Sayeed and Demberg (2013) we put role label annotations for PPs on the preposition rather than their NP child, following of the CoNLL 2005 shared task (Carreras and Màrquez, 2005).

[2] Prediction tree $\mathcal{T}_{pr}$ in our algorithm is only used during verification, so it set to nil for substitution and adjunction operations.

Banks refused to open. Naturally, some nodes in the lexicon trees might have multiple candidate role labels. For example, the substitution NP node of the second tree takes two labels, namely A0 and A1. These stem from different role *signatures* when the same elementary tree occurs in different contexts during training (A1 only on the NP; A0 on the NP and A1 on S). For simplicity's sake, we collapse different signatures, and let a classifier labeller to disambiguate such cases (see Section 4.4).

Algorithm 1 Incremental Role Propagation Alg.

```
1:  procedure IRPA(π_ip, T, T_pr)
2:      Σ ← ∅        ▷ Σ is a dictionary of (π_ip, ⟨l,a,p⟩) pairs
3:      if parser operation is substitution or adjunction then
4:          CREATE-TRIPLES(π_ip, T)
5:      else if parser operation is verification then
6:          CREATE-TRIPLES-VERIF(π_ip, T, T_pr)
        return set of triples ⟨l,a,p⟩ for prefix tree π
7:  procedure CREATE-TRIPLES(π_ip, T)
8:      if HAS-ROLES(π_ip) then
9:          UPDATE-TRIPLE(π_ip, T)
10:     else if HAS-ROLES(T) then
11:         T_ip ← substitution or foot node of T
12:         ADD-TRIPLE(π_ip, T_ip, T)
13:     for all remaining nodes n ∈ T with roles do
14:         ADD-TRIPLE(π_ip, n, T)        ▷ incomplete triples
15: procedure CREATE-TRIPLES-VERIF(π_ip, T_v, T_pr)
16:     if HAS-ROLES(T_v) then
17:         anchor ← lexeme of T_v
18:         for all T_ip ← node in T_v with role do
19:             T_pr,ip ← matching node of T_ip in T_pr
20:             CREATE-TRIPLES(T_pr,ip, T_v)
        ▷ Process the rest of covered nodes in T_pr with roles
21:     for all remaining T_pr,ip ← node in T_pr with role do
22:         UPDATE-TRIPLE(T_pr,ip, T_pr)
23: function UPDATE-TRIPLE(π_ip, T)
24:     dep ← FIND-INCOMPLETE(Σ, T_ip)
25:     anchor ← lexeme of T
26:     if anchor of T is predicate then
27:         SET-PREDICATE(dep, anchor)
28:     else if anchor of T is argument then
29:         SET-ARGUMENT(dep, anchor)
        return dep
30: procedure ADD-TRIPLE(π_ip, T_ip, T)
31:     dep ← ⟨[roles of T_ip], nil, nil⟩
32:     anchor ← lexeme of T
33:     if anchor of T is predicate then
34:         SET-PREDICATE(dep, anchor)
35:         SET-ARGUMENT(dep, head of π_ip)
36:     else if anchor of T is argument then
37:         if T is auxiliary then                ▷ adjunction
38:             SET-ARGUMENT(dep, anchor)
39:         else        ▷ substitution: arg is head of prefix tree
40:             SET-ARGUMENT(dep, head of T_ip)
41:             pred ← find dep ∈ Σ with matching π_ip
42:             SET-PREDICATE(dep, pred)
43:     Σ ← (π_ip, dep)
```

Once we process *Banks*, the prefix tree becomes the lexical entry for this word, see the first column of Figure 5b. Next, we process *refused*: the parser substitutes the prefix tree into the elementary tree $\mathcal{T}$ of *refused*;[3] the integration point π_{ip} on the prefix tree is the topmost NP. Since the operation is a substitution (line 3), we create triples between $\mathcal{T}$ and π_{ip} via CREATE-TRIPLES (lines 7–12). π_{ip} does not have any role information (line 8), so we proceed to add a new semantic triple between the role-labeled integration point $\mathcal{T}_{ip}$, i.e., substitution NP node of $\mathcal{T}$, and π_{ip}, via ADD-TRIPLE (lines 30–43). First, we create an incomplete semantic triple with all roles from $\mathcal{T}_{ip}$ (line 31). Then we set the predicate to the anchor of $\mathcal{T}$ to be the word *refused*, and the argument to be the head word of the prefix tree, *Banks* (lines 34–35). Note that predicate identification is a trivial task based on part-of-speech information in the elementary tree.[4]

Then, we add the pair (NP $\rightarrow$ ⟨{A0,A1},Banks, refused⟩) to a dictionary (line 43). Storing the integration point along with the semantic triple is essential, to be able to recover incomplete triples in later stages of the algorithm. Finally, we repeat this process for all remaining nodes on $\mathcal{T}$ that have roles, in our example the substitution node S (lines 13–14). This outputs an incomplete triple, ⟨{A1},nil,refused⟩.

Next, the parser decides to substitute a prediction tree (third tree in Figure 5a) into the substitution node S of the prefix tree. Since the integration point is on the prefix tree and has role information (line 8), the corresponding triple should already be present in our dictionary. Upon retrieving it, we set the nil argument to the anchor of the incoming tree. Since it is a prediction tree, we set it to the root of the tree, namely S_2 (phrase labels in triples are denoted by italics), but mark the triple as yet incomplete. This distinction allows us to fill in the correct lexical information once it becomes available, i.e, when the tree gets verified. We also add an incomplete triple for the trace t in the subject position of the prediction tree, as described above. Note that this triple contains multiple roles; this is expected given that prediction trees are unlexicalized and occur in a wide variety of contexts.

When the next verb arrives, the parser successfully verifies it against the embedded prediction

[3] PLTAG parsing operations can occur in two ways: An elementary tree can be substituted into the substitution node of the prefix tree, or the prefix tree can be substituted into a node of an elementary tree. The same holds for adjunction.

[4] Most predicates can be identified as anchors of non-modifier auxiliary trees. However, there are exceptions to this rule, i.e., modifier auxiliary trees and non-modifier non-auxiliary trees being also verbs in our lexicon, hence the use of the more reliable POS tags.

	IRPA	MaltParser
Banks	–	–
refused	⟨{A0,A1},Banks,refused⟩, ⟨A1,S_2,refused⟩, ⟨{A0,A1,A2},*t*,nil⟩	⟨A0,Banks,refused⟩
to	–	–
open	⟨A1,to,refused⟩, ⟨A1,Banks,open⟩	⟨A1,to,refused⟩, ⟨A0,Banks,open⟩
today	⟨AM-TMP,today,open⟩	⟨AM-TMP,today,open⟩

Table 1: Complete and incomplete semantic triple generation, comparing IRPA and a system that maps gold-standard role labels onto MaltParser incremental dependencies for Figure 4.

tree within the prefix tree (last step of Figure 5b). Our algorithm first cycles through all nodes that match between the verification tree $\mathcal{T}_v$ and the prediction tree $\mathcal{T}_{pr}$ and will complete or create new triples via CREATE-TRIPLES (lines 18–20). In our example, the second semantic triple gets completed by replacing S_2 with the head of the subtree rooted in S. Normally, this would be the verb *open*, but in this case the verb is followed by the infinitive marker *to*, hence we heuristically set it to be the argument of the triple instead, following Carreras and Màrquez (2005). For the last triple, we set the predicate to the anchor of $\mathcal{T}_v$ *open*, and now are able to remove the excess role labels A0 and A2. This illustrated how the lexicalized verification tree disambiguates the semantic information stored in the prediction tree. Finally, trace *t* is set to the closest NP head that is below the same phrase subtree, in this case *Banks*. Note that *Banks* is part of two triples as shown in the last tree of Figure 5b: it is either an A0 or an A1 for *refused* and an A1 for *open*.

We are able to create incomplete semantic triples after the prediction of the upcoming verb at step 2, as shown in Figure 5b. This is not possible using an incremental dependency parser such as MaltParser (Nivre et al., 2007) that lacks a predictive component. Table 1 illustrates this by comparing the output of IRPA for Figure 5b with the output of a baseline system that maps role labels onto the syntactic dependencies in Figure 4, generated incrementally by MaltParser (see Section 5.3 for a description of the MaltParser baseline). MaltParser has to wait for the verb *open* before outputting the relevant semantic triples. In contrast, IRPA outputs incomplete triples as soon as the information is available, and later on updates its decision. (MaltParser also incorrectly assigns A0 for the *Banks–open* pair.)

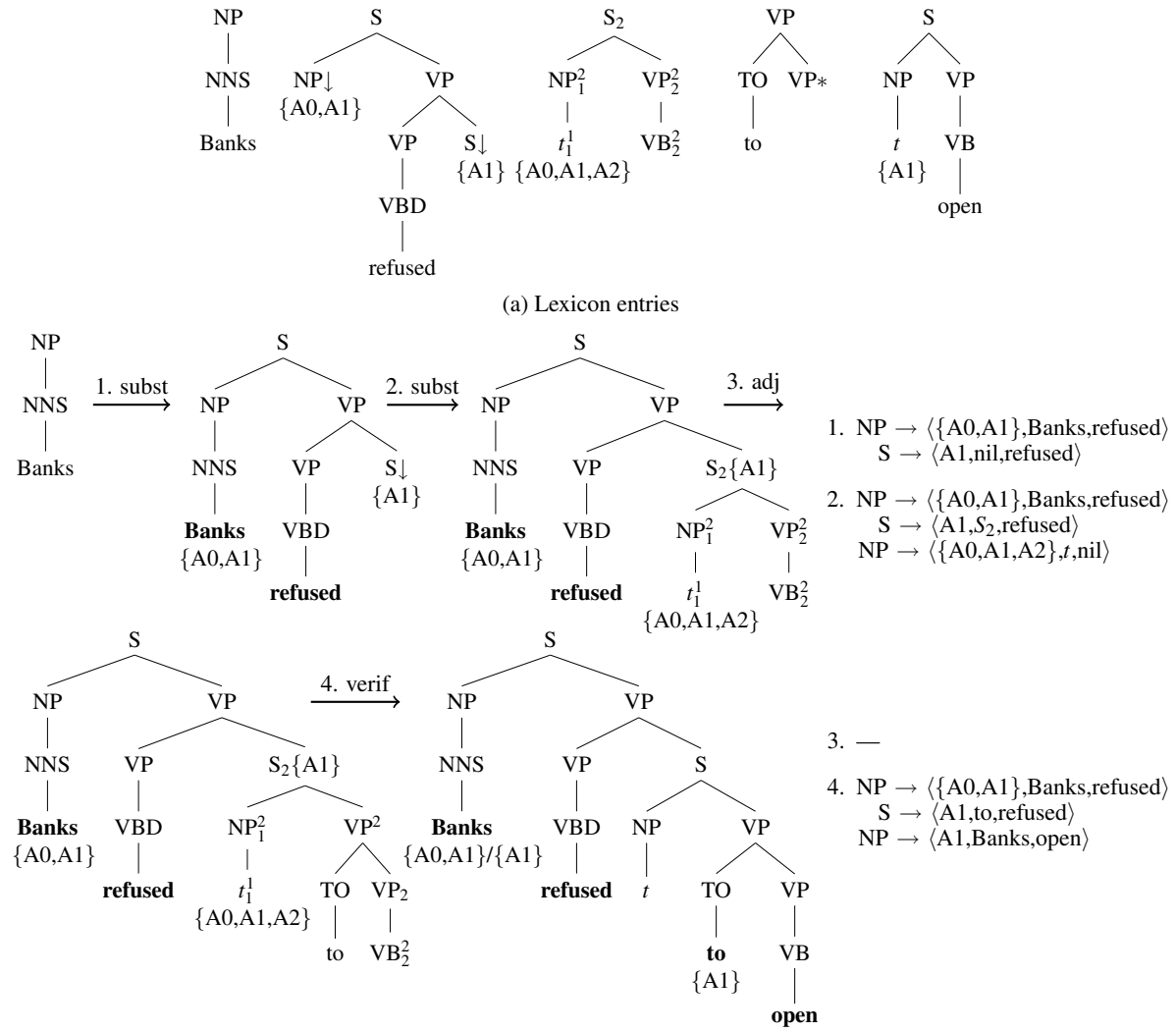
4.4 Argument Identification and Role Label Disambiguation

IRPA produces semantic triples for every role annotation present in the lexicon entries, which will often overgenerate role information. Furthermore, some triples have more than one role label attached to them. During verification, we are able to filter out the majority of labels in the corresponding prediction trees; However, most triples are created via substitution and adjunction.

In order to address these problems we adhere to the following classification and ranking strategy: after each semantic triple gets completed, we perform a binary classification that evaluates its suitability as a whole, given bilexical and syntactic information. If the triple is identified as a good candidate, then we perform multi-class classification over role labels: we feed the same bilexical and syntactic information to a logistic classifier, and get a ranked list of labels. We then use this list to re-rank the existing ambiguous role labels in the semantic triple, and output the top scoring ones.

The identifier is a binary L2-loss support vector classifier, and the role disambiguator an L2-regularized logistic regression classifier, both implemented using the efficient LIBLINEAR framework of Fan et al. (2008). The features used are based on Björkelund et al. (2009) and Liu and Sarkar (2007), and are listed in Table 2.

The bilexical features are: predicate POS tag, predicate lemma, argument word form, argument POS tag, and position. The latter indicates the position of the argument relative to the predicate, i.e., before, on, or after. The syntactic features are: the predicate and argument elementary trees without the anchors (to avoid sparsity), the category of the integration point node on the prefix tree where the elementary tree of the argument attaches to, an alphabetically ordered set of the categories of the fringe nodes of the prefix tree after attaching the argument tree, and the path of PLTAG operations applied between the argument and the predicate. Note that most of the original features used by Björkelund et al. (2009) and others are not applicable in our context, as they exploit information that is not accessible incrementally. For example, sibling information to the right of the word is not available. Furthermore, our PLTAG parser does not compute syntactic dependencies, hence these cannot serve as features (and in any case not all dependencies are available incrementally, see Figure 4). To counterbalance this, we use local syntactic information stored in the fringe of the pre-



(a) Lexicon entries

(b) Incremental parsing using PLTAG and incremental propagation of roles

Figure 5: Incremental Role Propagation Algorithm application for the sentence *Banks refused to open*.

Bilexical	Syntactic
PredPOS	PredElemTree
PredLemma	ArgElemTree
ArgWord	IntegrationPoint
ArgPOS	PrefixFringe
Position	OperationPath

Table 2: Features for argument identification and role label disambiguation.

fix tree. We also store the series of operations applied by our parser between argument and predicate, in an effort to emulate the effect of recovering longer-range patterns.

5 Experimental Design

5.1 PLTAG and Classifier Training

We extracted the semantically-enriched lexicon and trained the PLTAG parser by converting the Wall Street Journal part of Penn Treebank to PLTAG format. We used Propbank to retrieve semantic role annotation, as described in Section 4.2. We trained the PLTAG parser according to Demberg et al. (2013) and evaluated the parser on section 23, on sentences with 40 words or less, given gold POS tags for each word, and achieved a labeled bracket F_1 score of 79.41.

In order to train the argument identification and role label disambiguation classifiers, we used the English portion of the CoNLL 2009 Shared Task (Hajič et al., 2009; Surdeanu et al., 2008). It consists of the Penn Treebank, automatically converted to dependencies following Johansson and

Nugues (2007), accompanied by semantic role label annotation for every argument pair. The latter is converted from Propbank based on Carreras and Màrquez (2005). We extracted the bilexical features for the classifiers directly from the gold standard annotation of the training set. The syntactic features were obtained as follows: for every sentence in the training set we applied IRPA using the trained PLTAG parser, with gold standard lexicon entries for each word of the input sentence. This ensures near perfect parsing accuracy. Then for each semantic triple predicted incrementally, we extracted the relevant syntactic information in order to construct training vectors. If the identified predicate-argument pair was in the gold standard then we assigned a positive label for the identification classifier, otherwise we flagged it as negative. For those pairs that are not identified by IRPA but exist in the gold standard (false negatives), we extracted syntactic information from already identified similar triples, as follows: We first look for correctly identified arguments, wrongly attached to a different predicate and re-create the triple with correct predicate/argument information. If no argument is found, we then pick the argument in the list of identified arguments for a correct predicate with the same POS-tag as the gold-standard argument. In the case of the role label disambiguation classifier we just assign the gold label for every correctly identified pair, and ignore the (possibly ambiguous) predicted one. After tuning on the development set, the argument identifier achieved an accuracy of 92.18%, and the role label disambiguation classifier, 82.37%.

5.2 Evaluation

The focus of this paper is to build a system that is able to output semantic role labels for predicate-argument pairs incrementally, as soon as they become available. In order to properly evaluate such a system, we need to measure its performance incrementally. We propose two different cumulative scores for assessing the (possibly incomplete) semantic triples that have been created so far, as the input is processed from left to right, *per word.* The first metric is called Unlabeled Prediction Score (UPS) and gets updated for every identified argument or predicate, even if the corresponding semantic triple is incomplete. Note that UPS does not take into account the role label, it only measures predicate and argument identification. In this respect it is analogous to unlabeled dependency accuracy reported in the parsing literature. We expect a model that is able to predict semantic roles to achieve an improved UPS result compared to a system that does not do prediction, as illustrated in Table 1. Our second score, Combined Incremental SRL Score (CISS), measures the identification of complete semantic role triples (i.e., correct predicate, predicate sense, argument, and role label) per word; by the end of the sentence, CISS coincides with standard combined SRL accuracy, as reported in CoNLL 2009 SRL-only task. This score is analogous to labeled dependency accuracy in parsing.

Note that conventional SRL systems such as Björkelund et al. (2009) typically assume gold standard syntactic information. In order to emulate this, we give our parser gold standard lexicon entries for each word in the test set; these contain all possible roles observed in the training set for a given elementary tree (and all possible senses for each predicate). This way the parser achieves a syntactic parsing F_1 score of 94.24, thus ensuring the errors of our system can be attributed to IRPA and the classifiers. Also note that we evaluate on verb predicates only, therefore trivially reducing the task of predicate identification to the simple heuristic of looking for words in the sentence with a verb-related POS tag and excluding auxiliaries and modals. Likewise, predicate sense disambiguation on verbs presumably is trivial, as we observed almost no ambiguity of senses among lexicon entries of the same verb (we adhered to a simple majority baseline, by picking the most frequent sense, given the lexeme of the verb, in the few ambiguous cases). It seems that the syntactic information held in the elementary trees discriminates well among different senses.

5.3 System Comparison

We evaluated three configurations of our system. The first configuration (iSRL) uses all semantic roles for each PLTAG lexicon entry, applies the PLTAG parser, IRPA, and both classifiers to perform identification and disambiguation, as described in Section 4. The second one (Majority-Baseline), solves the problem of argument identification and role disambiguation without the classifiers. For the former we employ a set of heuristics according to Lang and Lapata (2014), that rely on gold syntactic dependency information, sourced from CoNLL input. For the latter, we choose the most frequent role given the gold standard dependency relation label for the particular argument. Note that dependencies have been produced in view of the whole sentence and not incrementally.

System	Prec	Rec	F1
iSRL-Oracle	91.00	80.26	85.29
iSRL	81.48	75.51	78.38
Majority-Baseline	71.05	58.10	63.92
Malt-Baseline	60.90	46.14	52.50

Table 3: Full-sentence combined SRL score

This gives the baseline a considerable advantage especially in case of longer range dependencies. The third configuration (iSRL-Oracle), is identical to iSRL, but uses the gold standard roles for each PLTAG lexicon entry, and thus provides an upper-bound for our methodology. Finally, we evaluated against Malt-Baseline, a variant of Majority-Baseline that uses the MaltParser of Nivre et al. (2007) to provide labeled syntactic dependencies MaltParser is a state-of-the-art shift-reduce dependency parser which uses an incremental algorithm. Following Beuck et al. (2011), we modified the parser to provide intermediate output at each word by emitting the current state of the dependency graph before each shift step. We trained MaltParser using the arc-eager algorithm (which outperformed the other parsing algorithms available with MaltParser) on the CoNLL dataset, achieving a labeled dependency accuracy of 89.66% on section 23.

6 Results

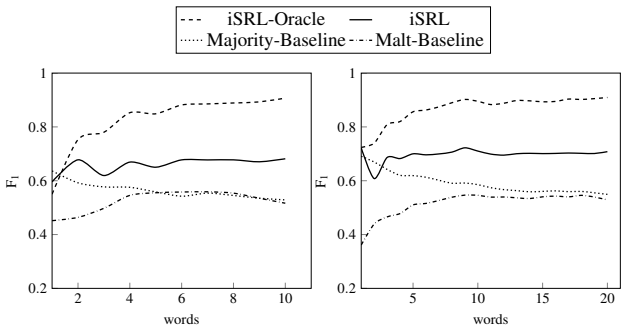
Figures 6 and 7 show the results on the incremental SRL task. We plot the F_1 for Unlabeled Prediction Score (UPS) and Combined Incremental SRL Score (CISS) per word, separately for sentences of lengths 10, 20, 30, and 40 words. The task gets harder with increasing sentence length, hence we can only meaningfully compare the average scores for sentence of the same length. (This approach was proposed by Sangati and Keller 2013 for evaluating the performance of incremental parsers.)

The UPS results in Figure 6 clearly show that our system (iSRL) outperforms both baselines on unlabeled argument and predicate prediction, across all four sentence lengths. Furthermore, we note that the iSRL system achieves a near-constant performance for all sentence prefixes. Our PLTAG-based prediction/verification architecture allows us to correctly predict incomplete semantic role triples, even at the beginning of the sentence. Both baselines perform worse than the iSRL system in general. Moreover, the Malt-Baseline performs badly on the initial sentence prefixes (up to word 10), presumably as it does not benefit from syntactic prediction, and thus cannot generate incomplete triples early in the sentence, as illustrated in Table 1. The Majority-Baseline also does not do prediction, but it has access to gold-standard syntactic dependencies, and thus outperforms the Malt-Baseline on initial sentence prefixes. Note that due to prediction, our system tends to over-generate incomplete triples in the beginning of sentences, compared to non-incremental output, which may inflate UPS for the first words. However, this cancels out later in the sentence if triples are correctly completed; failure to do so would decrease UPS. The near-constant performance of our output illustrates this phenomenon. Finally, the iSRL-Oracle outperforms all other systems, as it benefits from correct role labels and correct PLTAG syntax, thus providing an upper limit on performance.

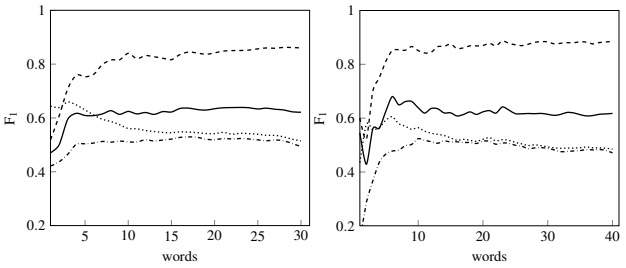
The CISS results in Figure 7 present a similar picture. Again, the iSRL system outperforms both baselines at all sentence lengths. In addition, it shows particularly strong performance (almost at the level of the iSRL-Oracle) at the beginning of the sentence. This presumably is due to the fact that our system uses prediction and is able to identify correct semantic role triples earlier in the sentence. The baselines also show higher performance early in the sentence, but to a lesser degree.

Table 3 reports traditional combined SRL scores for full sentences over all sentence lengths, as defined for the CoNLL task. Our iSRL system outperforms the Majority-Baseline by almost 15 points, and the Malt-Baseline by 25 points. It remains seven points below the iSRL-Oracle upper limit.

Finally, in order to test the effect of syntactic parsing on our system, we also experimented with a variant of our iSRL system that utilizes all lexicon entries for each word in the test set. This is similar to performing the CoNLL 2009 joint task, which is designed for systems that carry out both syntactic parsing and semantic role labeling. This variant achieved a full sentence F-score of 68.0%, i.e., around 10 points lower than our iSRL system. This drop in score correlates with the difference in syntactic parsing F-score between the two versions of PLTAG parser (94.24 versus 79.41), and is expected given the high ambiguity of the lexicon entries for each word. Note, however, that the full-parsing version of our system still outperforms Malt-Baseline by 15 points.

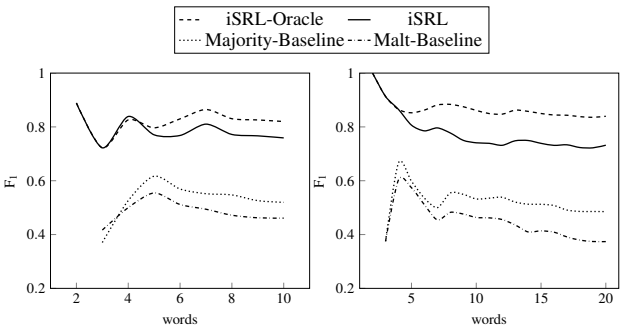


(a) 10 words (b) 20 words

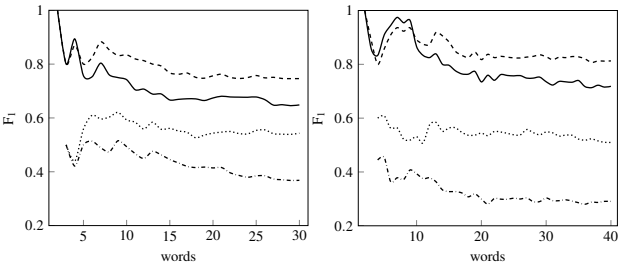


(c) 30 words (d) 40 words

Figure 6: Unlabeled Prediction Score (UPS)



(a) 10 words (b) 20 words



(c) 30 words (d) 40 words

Figure 7: Combined iSRL Score (CISS)

7 Conclusions

In this paper, we introduced the new task of incremental semantic role labeling and proposed a system that solves this task by combining an incremental TAG parser with a semantically enriched lexicon, a role propagation algorithm, and a cascade of classifiers. This system achieved a full-sentence SRL F-score of 78.38% on the standard CoNLL dataset. Not only is the full-sentence score considerably higher than the Majority-Baseline (which is a strong baseline, as it uses gold-standard syntactic dependencies), but we also observe that our iSRL system performs well incrementally, i.e., it predicts both complete and incomplete semantic role triples correctly early on in the sentence. We attributed this to the fact that our TAG-based architecture makes it possible to predict upcoming syntactic structure together with the corresponding semantic roles.

Acknowledgments

EPSRC support through grant EP/I032916/1 “An integrated model of syntactic and semantic prediction in human language processing” to FK and ML is gratefully acknowledged.

References

Beuck, Niels, Arne Khn, and Wolfgang Menzel. 2011. Incremental parsing and the evaluation of partial dependency analyses. In *Proceedings of the 1st International Conference on Dependency Linguistics*. Depling 2011.

Björkelund, Anders, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, Stroudsburg, PA, USA, CoNLL ’09, pages 43–48.

Carreras, Xavier and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL ’05, pages 152–164.

Chiang, David. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. pages 456–463.

Demberg, Vera, Frank Keller, and Alexander Koller. 2013. Incremental, predictive parsing with psycholinguistically motivated tree-adjoining grammar. *Computational Linguistics* 39(4):1025–1066.

Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Li-

blinear: A library for large linear classification. *Journal of Machine Learning Research* 9:1871–1874.

Hajič, Jan, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), June 4-5*. Boulder, Colorado, USA.

Johansson, Richard and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for english. In Joakim Nivre, Heiki-Jaan Kalep, Kadri Muischnek, and Mare Koit, editors, *NODALIDA 2007 Proceedings*. University of Tartu, pages 105–112.

Joshi, Aravind K. and Yves Schabes. 1992. Tree adjoining grammars and lexicalized grammars. In Maurice Nivat and Andreas Podelski, editors, *Tree Automata and Languages*, North-Holland, Amsterdam, pages 409–432.

Keller, Frank. 2010. Cognitively plausible models of human language processing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Companion Volume: Short Papers*. Uppsala, pages 60–67.

Lang, Joel and Mirella Lapata. 2014. Similarity-driven semantic role induction via graph partitioning. *Computational Linguistics Accepted* pages 1–62. To appear.

Liu, Yudong and Anoop Sarkar. 2007. Experimental evaluation of LTAG-based features for semantic role labeling. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic, pages 590–599.

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

Màrquez, Lluís, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. 2008. Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics* 34(2):145–159.

Mazzei, Alessandro, Vincenzo Lombardo, and Patrick Sturt. 2007. Dynamic TAG and lexical dependencies. *Research on Language and Computation* 5:309–332.

Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering* 13:95–135.

Padó, Ulrike, Matthew W. Crocker, and Frank Keller. 2009. A probabilistic model of semantic plausibility in sentence processing. *Cognitive Science* 33(5):794–838.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics* 31(1):71–106.

Pickering, Martin J., Matthew J. Traxler, and Matthew W. Crocker. 2000. Ambiguity resolution in sentence processing: Evidence against frequency-based accounts. *Journal of Memory and Language* 43(3):447–475.

Sangati, Federico and Frank Keller. 2013. Incremental tree substitution grammar for parsing and word prediction. *Transactions of the Association for Computational Linguistics* 1(May):111–124.

Sayeed, Asad and Vera Demberg. 2013. The semantic augmentation of a psycholinguistically-motivated syntactic formalism. In *Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 57–65.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.

Witten, Ian H. and Timothy C. Bell. 1991. The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. *Information Theory, IEEE Transactions on* 37(4):1085–1094.

Xia, Fei, Martha Palmer, and Aravind Joshi. 2000. A uniform method of grammar extraction and its applications. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in*

Natural Language Processing and Very Large Corpora. pages 53–62.

A Graph-based Approach for Contextual Text Normalization

Çağıl Sönmez and Arzucan Özgür
Department of Computer Engineering
Bogazici University
Bebek, 34342 Istanbul, Turkey
{cagil.ulusahin,arzucan.ozgur}@boun.edu.tr

Abstract

The informal nature of social media text renders it very difficult to be automatically processed by natural language processing tools. Text normalization, which corresponds to restoring the non-standard words to their canonical forms, provides a solution to this challenge. We introduce an unsupervised text normalization approach that utilizes not only lexical, but also contextual and grammatical features of social text. The contextual and grammatical features are extracted from a word association graph built by using a large unlabeled social media text corpus. The graph encodes the relative positions of the words with respect to each other, as well as their part-of-speech tags. The lexical features are obtained by using the longest common subsequence ratio and edit distance measures to encode the surface similarity among words, and the double metaphone algorithm to represent the phonetic similarity. Unlike most of the recent approaches that are based on generating normalization dictionaries, the proposed approach performs normalization by considering the context of the non-standard words in the input text. Our results show that it achieves state-of-the-art F-score performance on standard datasets. In addition, the system can be tuned to achieve very high precision without sacrificing much from recall.

1 Introduction

Social text, which has been growing and evolving steadily, has its own lexical and grammatical features (Choudhury et al., 2007; Eisenstein, 2013). *lol* meaning *laughing out loud*, *xoxo* meaning *kissing*, *4u* meaning *for you* are among the most commonly used examples of this jargon. In addition, these informal expressions in social text usually take many different lexical forms when generated by different individuals (Eisenstein, 2013). The limited accuracies of the Speech-to-Text (STT) tools in mobile devices, which are increasingly being used to post messages on social media platforms, along with the scarcity of attention of the users result in additional divergence of social text from more standard text such as from the newswire domain. Tools such as spellchecker and slang dictionaries have been shown to be insufficient to cope with this challenge long time ago (Sproat et al., 2001). In addition, most Natural Language Processing (NLP) tools including named entity recognizers and dependency parsers generally perform poorly on social text (Ritter et al., 2010).

Text normalization is a preprocessing step to restore non-standard words in text to their original (canonical) forms to make use in NLP applications or more broadly to understand the digitized text better (Han and Baldwin, 2011). For example, *talk 2 u later* can be normalized as *talk to you later* or similarly *enormoooos, enrmss* and *enourmos* can be normalized as *enormous*. Other examples of text messages from Twitter and their corresponding normalized forms are shown in Table 1.

The non-standard words in text are referred to as Out of Vocabulary (OOV) words. The normalization task restores the OOV words to their In Vocabulary (IV) forms. Social text is continuously evolving with new words and named entities that are not in the vocabularies of the systems (Hassan and Menezes, 2013). Therefore, not every OOV word (e.g. *iPhone, WikiLeaks or tok-*

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 313–324,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Hav guts to say wat u desire.. Dnt beat behind da bush!! And 1 mre thng no mre say ***y*** *r people's man!!*	Have guts to say what you desire.. Don't beat behind the bush!! And one more thing no more say **you** are people's man!!
There r sm songs u don't want 2 listen 2 yl walking cos when u start dancing ppl won't knw ***y****.*	There are some songs you don't want to listen to while walking because when you start dancing people won't know **why**.

Table 1: Sample tweets and their normalized forms.

enizing) should be considered for normalization. The OOV tokens that should be considered for normalization are referred to as ill-formed words. Ill-formed words can be normalized to different canonical words depending on the context of the text. For example, let's consider the two examples in Table 1. "y" is normalized as "you" in the first one and as "why" in the second one.

In this paper, we propose a graph-based text normalization method that utilizes both contextual and grammatical features of social text. The contextual information of words is modeled by a word association graph that is created from a large social media text corpus. The graph represents the relative positions of the words in the social media text messages and their Part-of-Speech (POS) tags. The lexical similarity features among the words are modeled using the longest common subsequence ratio and edit distance that encode the surface similarity and the double metaphone algorithm that encodes the phonetic similarity. The proposed approach is unsupervised, which is an important advantage over supervised systems, given the continuously evolving language in the social media domain. The same OOV word may have different appropriate normalizations depending on the context of the input text message. Recently proposed dictionary-based text normalization systems perform dictionary look-up and always normalize the same OOV word to the same IV word regardless of the context of the input text (Han et al., 2012; Hassan and Menezes, 2013). On the other hand, the proposed approach does not only make use of the general context information in a large corpus of social media text, but it also makes use of the context of the OOV word in the input text message. Thus, an OOV word can be normalized to different IV words depending on the context of the input text.

2 Related Work

Early work on text normalization mostly made use of the noisy channel model. The first work that had a significant performance improvement over the previous research was by Brill and Moore (2000). They proposed a novel noisy channel model for spell checking based on string to string edits. Their model depended on probabilistic modeling of sub-string transformations.

Toutanova and Moore (2002) improved this approach by extending the error model with phonetic similarities over words. Their approach is based on learning rules to predict the pronunciation of a single letter in the word depending on the neighbouring letters in the word.

Choudhury et al. (2007) developed a supervised Hidden Markov Model based approach for normalizing Short Message Service (SMS) texts. They proposed a word for word decoding approach and used a dictionary based method to normalize commonly used abbreviations and non-standard usage (e.g. "howz" to "how are" or "aint" to "are not"). Cook and Stevenson (2009) extended this model by introducing an unsupervised noisy channel model. Rather than using one generic model for all word formations as in (Choudhury et al., 2007), they used a mixture model in which each different word formation type is modeled explicitly.

The limitations of these methods were that they did not consider contextual features and assumed that tokens have unique normalizations. In the text normalization task several OOV tokens are ambiguous and without contextual information it is not possible to build models that can disambiguate transformations correctly.

Aw et al. (2006) proposed a phrase-based statistical machine translation (MT) model for the text normalization task. They defined the problem as translating the SMS language to the English language and based their model on two submodels: a word based language model and a phrase based lexical mapping model (channel model). Their system also benefits from the input context and they argue that the strength of their model is in its ability to disambiguate mapping as in "2" → "two" *or* "to", and "w" → "with" *or* "who". Making use of the whole conversation, this is the closest approach to ours in the sense of utilizing contextual sensitivity and coverage.

Pennell and Liu (2011) on the other hand, proposed a character level MT system, that is robust to new abbreviations. In their two phased system, a character level trained MT model is used to produce word hypotheses and a trigram LM is used to choose a hypothesis that fits into the input context.

The MT based models are supervised models, a drawback of which is that they require annotated data. Annotated training data is not readily available and is difficult to create especially for the rapidly evolving social media text (Yang and Eisenstein, 2013).

More recent approaches handled the text normalization task by building normalization lexicons. Han and Baldwin (2011) developed a two phased model, where they only consider the ill-formed OOV words for normalization. First, a confusion set is generated using the lexical and phonetic distance features. Later, the candidates in the confusion set are ranked using a mixture of dictionary look up, word similarity based on lexical edit distance, phonemic edit distance, prefix sub-string, suffix sub-string and longest common subsequence (LCS), as well as context support metrics. Chrupala (2014) on the other hand achieved lower word error rates without using any lexical resources.

Gouws et al. (2011) investigated the distinct contributions of features that are highly depended on user-centric information such as the geological location of the users and the twitter client that the tweet is received from. Using such user-based contextual metrics they modelled the transformation distributions across populations.

Liu et al. (2012) proposed a broad coverage normalization system, which integrates an extended noisy channel model, that is based on enhanced letter transformations, visual priming, string and phonetic similarity. They try to improve the performance of the top n normalization candidates by integrating human perspective modeling.

Yang and Eisenstein (2013) introduced an unsupervised log linear model for text normalization. Their joint statistical approach uses local context based on language modeling and surface similarity. Along with dictionary based models, Yang and Eisenstein's model have obtained a significant improvement on the performance of text normalization systems.

Another relevant study is conducted by Hassan and Menezes (2013), who generated a normalization lexicon using Markov random walks on a contextual similarity lattice that they created using 5-gram sequences of words. The best normalization candidates are chosen using the average hitting time and lexical similarity features. Context of a word in the center of a 5-gram sequence is defined by the other words in the 5-gram. Even if one word is not the same, the context is considered to be different. This is a relatively conservative way for modeling the prior contexts of words. In our model, we filtered candidate words based on their grammatical properties and let each neighbouring token to contribute to the prior context of a word, which leads to both a higher recall and a higher precision.

3 Methodology

In this paper, we propose a graph-based approach that models both contextual and lexical similarity features among an ill-formed OOV word and candidate IV words. An input text is first preprocessed by tokenizing and Part-Of-Speech (POS) tagging. If the text contains an OOV word, the normalization candidates are chosen by making use of the contextual features, which are extracted from a pre-generated directed word association graph, as well as lexical similarity features. Lexical similarity features are based on edit distance, longest common subsequence ratio, and double metaphone distance. In addition, a slang dictionary[1] is used as an external resource to enrich the normalization candidate set. The details of the approach are explained in the following subsections.

3.1 Preprocessing

After tokenization, the next step in the pipeline is POS tagging each token using a POS tagger specifically designed for social media text. Unlike the regular POS taggers designed for well-written newswire-like text, social media POS taggers provide a broader set of tags specific to the peculiarities of social text (Owoputi et al., 2013; Gimpel et al., 2011). Using this extended set of tags we can identify tokens such as discourse markers (e.g. rt for retweets, cont. for a tweet whose content follows up in the coming tweet) or URLs. This enables us to model better the context of the words in social media text. A sample preprocessed sentence is shown in Table 3.

[1] http://www.noslang.com

As shown in Table 2, after preprocessing, each token is assigned a POS tag with a confidence score between 0 and 1[2]. Later, we use these confidence scores in calculating the edge weights in our context graph. Note that even though the words *w* and *beatiful* are misspelled, they are tagged correctly by the tagger, with lower confidence scores though.

Token	*POS tag*	Tag confidence
with	*Preposition*	0.9963
a	*Determiner*	0.9980
beautiful	*Adjective*	0.9971
smile	*Noun*	0.9712
w	*Preposition*	0.7486
a	*Determiner*	0.9920
beatiful	*Adjective*	0.9733
smile	*Noun*	0.9806

Table 2: Sample POS tagger output

3.2 Graph construction

Contextual information of words is modeled through a word association graph created by using a large corpus of social media text. The graph encodes the relative positions of the POS tagged words in the text with respect to each other. After preprocessing, each text message in the corpus is traversed in order to extract the nodes and the edges of the graph. A node is defined with four properties: *id, oov, freq* and *tag*. The token itself is the *id* field. The *freq* property indicates the node's frequency count in the dataset. The *oov* field is set to True if the token is an OOV word. Following the prior work by Han and Baldwin, (2011) we used the GNU Aspell dictionary (v0.60.6) to determine whether a word is OOV or not. We also edited the output of Aspell dictionary to accept letters other than "a" and "i" as OOV words. A portion of the graph that covers parts of the sample sentence in Table 3 is shown in Figure 1.

In the created word association graph, each node is a unique set of a token and its POS tag. This helps us to identify the candidate IV words for a given OOV word by considering not only lexical and contextual similarity, but also grammatical similarity in terms of POS tags. For example, if the token *smile* has been frequently seen as a Noun or a Verb, and not in other forms in the dataset (e.g. Table 4), this provides evidence that it is not a good normalization candidate for an OOV token that has been tagged as a Pronoun. On the other hand, *smile* can be a good candidate for a Noun or a Verb OOV token, if it is lexically and contextually similar to it.

[2]CMU Ark Tagger (v0.3.2)

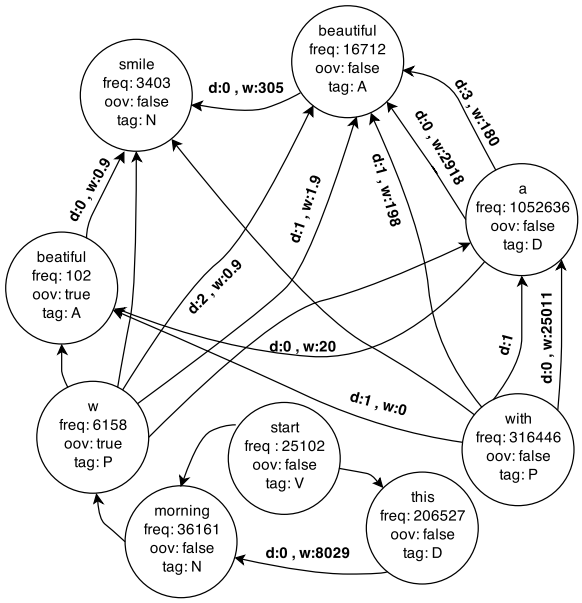


Figure 1: Portion of the word association graph for part of the sample sentence in Table 3. (d: distance, w: edge weight).

node id	**freq**	**oov**	**tag**
smile	3	False	A
smile	3403	False	N
smile	2796	False	V

Table 4: The different nodes in the word association graph representing the token *smile* tagged with different POS tags.

An edge is created between two nodes in the graph, if the corresponding word pair (i.e. token/POS pair) are contextually associated. Two words are considered to be contextually associated if they satisfy the following criteria:

- The two words co-occur within a maximum word distance of $t_{distance}$ in a text message in the corpus.
- Each word has a minimum frequency of $t_{frequency}$ in the corpus.

The directionality of the edges is based on the sequence of words in the text messages in the corpus. In other words, an edge between two nodes is directed from the earlier seen token towards the later seen token in a message. For example, Figure 2 shows the edges that would be derived

Let's$_L$	start$_V$	this$_D$	morning$_N$	w$_P$	a$_D$	beatiful$_A$	smile$_N$	.$_C$

Table 3: Sample tokenized, POS tagged sentence (L: nominal+verbal, V: verb, D: determiner, N: noun, P: Preposition, A: adjective, C: punctuation).

from a text including the phrase "with a beautiful smile". The direction (from,to) and the distance together represent a unique triplet. For each pair of nodes with a specific distance there is an edge with a positive weight, if the two nodes are contextually associated. Each co-occurrence of two contextually associated nodes increases the weight of the edge between them with an average of the nodes' POS tag confidence scores in the text message considered. If we are to expand the graph with the example phrase "with a beautiful smile", the weight of the edge with distance 2 from the node *with*|*P* to the node *smile*|*N* would increase by $(0.9963 + 0.9712)/2$, since the confidence score of the POS tag for the token *with* is 0.9963 and the confidence score of the POS tag of the token *smile* is 0.9712 as shown in Table 2.

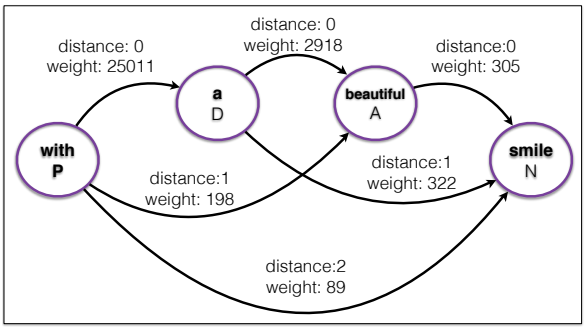


Figure 2: Sample nodes and edges from the word association graph.

3.3 Graph-based Contextual Similarity

Our graph-based contextual similarity method is based on the assumption that an IV word that is the canonical form of an OOV word appears in the same context with the corresponding OOV word. In other words, the two nodes in the graph share several neighbors that co-occur within the same distances to the corresponding two words in social media text. We also assume that an OOV word and its canonical form should have the same POS tag.

Given an input text for normalization, the next step after preprocessing is finding the normalization candidates for each OOV token in the input text. For each ill-formed OOV token o_i in the input text, first the list of tokens that co-occur with o_i in the input text and their positional distances to o_i are extracted. This list is called the neighbor list of token o_i, i.e., NL(o_i).

For each neighbor node n_j in NL(o_i), the word association graph is traversed, and the edges *from* or *to* the node n_j are extracted. The resulting edge list EL(o_i) has edges in the form of (n_j, c_k) or (c_k, n_j), where c_k is a candidate canonical form of the OOV word o_i. Here the neighbor node n_j can be an OOV node, but the candidate node c_k is chosen among the IV nodes. The edges in EL(o_i) are filtered by the relative distance of n_j to o_i as given in the NL(o_i). Any edge between n_j and c_k, whose distance is not the same as the distance between n_j and o_i is removed.

In addition to distance based filtering, POS tag based filtering is also performed on the edges in EL(o_i). Each candidate node should have the same POS tag with the corresponding OOV token. For the OOV token o_i that has the POS tag T_i, all the edges that include candidates with a tag other than T_i are removed from the edge list EL(o_i).

Figure 3 represents a portion from the graph where the neighbors and candidates of the OOV node "beatiful" are shown. In the sample sentence in Table 3 there are two OOV tokens to be normalized, $o_1 = w$ and $o_2 =$ *beatiful*. The neighbor list of o_2, NL(o_2) includes $n_1 = w$, $n_2 = a$ and $n_3 =$ *smile*. For each neighbor in NL(o_2), the candidate nodes ($c_1 =$ *broken*, $c_2 =$ *nice*, $c_3 =$ *new*, $c_4 =$ *beautiful*, $c_5 =$ *big*, $c_6 =$ *best*, $c_7 =$ *great*) are extracted. As shown in Figure 3, there are 11 lines representing the edges between the neighbors of the OOV token and the candidate nodes. These are representative edges in EL(o_2). Each member of the edge list has the same tag (A for Adjective) as the OOV node "beatiful" and the same distance to the corresponding neighbor node of the OOV node.

Each edge in EL(o_i) consists of a neighbor node n_j, a candidate node c_k and an edge weight edgeWeight(n_j, c_k). The edge weight represents the likelihood or the strength of association between the neighbor node n_j and the candidate node c_k. As described in the previous section the edge weights are computed based on the frequency

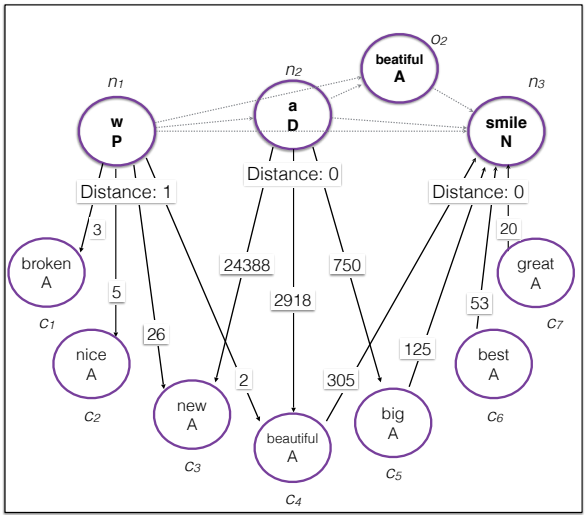


Figure 3: A portion of the graph that includes the OOV token "beatiful", its neighbors and the candidate nodes that each neighbor is connected to. Thick lines show the edge list with relative weights.

of co-occurrence of two tokens, as well as the confidence scores of their POS tags.

The edge weights of the edges in EL(o_2) are shown in Figure 3. The edges that are connected to the OOV neighbor "w" have smaller edge weights such as 3, 5, and 26. On the other hand, the edges that are connected to common words have higher weights. For example, the weight of the edge between the nodes "a" and "new" is 24388. This indicates that they are more common words, and frequently co-occur in the same form ("a new"). Although this edge weight metric is reasonable for identifying the most likely canonical form for the OOV word o_i, it has the drawback of favoring words with high frequencies like common words or stop words. Therefore, to avoid overrated words and get contextually related candidates, we normalize the edge weight edgeWeight(n_j, c_k) with the frequency of the candidate node c_k as shown in Equation 1.

Equation 1 provides a metric that captures contextual similarity based on binary associations. In order to achieve a more comprehensive contextual coverage, a contextual similarity feature is built based on the sum of the binary association scores of several neighbors. As shown in Equation 2, for a candidate node c_k the total edge weight score is the sum of the normalized edge weight scores EWNorm(n_j, c_k), which are the edge weights coming from the different neighbors of the OOV token o_i. We expect this contextual similarity feature to favor and identify the candidates which are (i) related to many neighbors, and (ii) have a high association score with each neighbor.

$$EWNorm(n_j, c_k) = edgeWeight(n_j, c_k)/freq(c_k) \quad (1)$$

$$EW\_Score(o_i, c_k) = \sum_{EL(o_i)} EWNorm(n_j, c_k) \quad (2)$$

Our word association graph includes both OOV and IV tokens, and our OOV detection depends on the spellchecker which fails to identify some OOV tokens that have the same spelling with an IV word. In order to propose better canonical forms, the frequencies of the normalization candidates in the social media corpus have also been incorporated to the contextual similarity feature. Nodes with higher frequencies lead to tokens that are in their most likely grammatical forms.

The final contextual similarity of the token o_i and the candidate c_k is the weighted sum of the total edge weight score and the frequency score of the candidate (see Equation 3). The frequency score of the candidate is a real number between 0 and 1. It is proportional to the frequency of the candidate with respect to the frequencies of the other candidates in the corpus. Since the total edge weight score is our primary contextual resource, we may want to favor edge weight scores. We give the frequency score a weight $0 \leq \beta \leq 1$ to be able to limit its effect on the total contextual similarity score.

$$\begin{aligned} contSimScore(o_i, c_k) = \; & EW\_Score(o_i, c_k) \\ & + \beta * freqScore(c_k) \end{aligned} \quad (3)$$

Hereby, we have the candidate list CL(o_i) for the OOV token o_i that includes all the unique candidates in EL(o_i) and their contextual similarity scores calculated.

3.4 Lexical Similarity

Following the prior work in (Han and Baldwin, 2011; Hassan and Menezes, 2013), our lexical similarity features are based on edit distance (Levenshtein, 1966), double metaphone (phonetic edit distance) (Philips, 2000), and a similarity function

($simCost$) (Contractor et al., 2010) which is defined as the ratio of the Longest Common Subsequence Ratio (LCSR) (Melamed, 1999) of two words and the Edit Distance (ED) between their skeletons (Equations 4 and 5), where the skeleton of a word is obtained by removing its vowels.

$$LCSR(o_j, c_k) = LCS(o_j, c_k)/maxLength(o_j, c_k) \quad (4)$$

$$simCost(o_j, c_k) = LCSR(o_j, c_k)/ED(o_j, c_k) \quad (5)$$

Following the tradition that is inspired from (Kaufmann and Kalita, 2010), before lexical similarity calculations, any repetitions of characters three or more times in OOV tokens are reduced to two (e.g. *goooood* is reduced to *good*). Then, the edit distance, phonetic edit distance, and simCost between each candidate in CL(o_i) and the OOV token o_i are calculated. Edit distance and phonetic edit distance are used to filter the candidates. Any candidate in CL(o_i) with an edit distance greater than t_{edit} and phonetic edit distance greater than $t_{phonetic}$ to o_i is removed from the candidate list CL(o_i).

$$\begin{aligned} lexSimScore(o_i, c_k) = \; & simCost(o_i, c_k) \\ & + \lambda * editScore(o_i, c_k) \end{aligned} \quad (6)$$

For the remaining candidates, the total lexical similarity score (Equation 6) is calculated using simCost and edit distance score[3]. Similar to contextual similarity score, here we have one main lexical similarity feature and one minor lexical similarity feature. The major lexical similarity feature is simCost, whereas the edit distance score is the minor feature. We assigned a weight $0 \leq \lambda \leq 1$ to the edit distance score to be able to lower its contribution while calculating the total lexical similarity score.

3.5 External Score

Since some social media text messages are extremely short and contain several OOV words, they do not provide sufficient context, i.e., IV neighbors, to enable the extraction of good candidates from the word association graph. Therefore, we extended the candidate list obtained through contextual similarity as described in the previous section, by including all the tokens in the word association graph that satisfy the edit distance and phonetic edit distance criteria. We also incorporated candidates from external resources, in other words from a slang dictionary and a transliteration table of numbers and pronouns. If a candidate occurs in the slang dictionary or in the transliteration table as a correspondence to its OOV word, it is assigned an external score of 1, otherwise it is assigned an external score of 0.

The transliterations were first used by (Gouws et al., 2011). Besides the token and its transliteration we also use its POS tag information, which was not available in their system.

The external score favors the well known interpretations of common OOV words. However, unlike the dictionary based methodologies, our system does not return the corresponding unabbreviated word in the slang dictionary or in the transliteration table directly. Only an external score gets assigned and the candidate still needs to compete with other candidates which may have higher contextual similarities and one of those contextually more similar candidates may be returned as the correct normalization instead of the candidate found equivalent to the OOV word in the slang dictionary (or in the transliteration table).

3.6 Overall Scoring

As shown in Equation 7, the final score of a candidate IV token c_k for an OOV token o_i is the sum of its lexical similarity score, contextual similarity score and external score with respect to o_i.

$$\begin{aligned} \text{candScore}(o_i, c_k) = \; & \text{lexSimScore}(o_i, c_k) \\ & + \text{contSimScore}(o_i, c_k) \\ & + \text{externalScore}(o_i, c_k) \end{aligned} \quad (7)$$

4 Experiments

4.1 Datasets

We used the LexNorm1.1 (LN) dataset (Han and Baldwin, 2011) and Pennell and Liu (2014)'s trigram dataset to evaluate our proposed approach. LexNorm1.1 contains 549 tweets with 1184 manually annotated ill-formed OOV tokens. It has been used by recent text normalization studies for evaluation, which enables us to directly compare our performance results with results obtained by the recent previous work (Han and Baldwin, 2011; Pennell and Liu, 2011; Han et al., 2012; Liu et al., 2012; Hassan and Menezes, 2013; Yang and Eisenstein, 2013; Chrupala, 2014). The trigram

[3] an approximate string comparison measure (between 0.0 and 1.0) using the edit distance https://sourceforge.net/projects/febrl/

dataset is an SMS-like corpus collected from twitter status updates sent via SMS. The dataset does not include the complete tweet text but trigrams from tweets and one OOV word in each trigram is annotated. In total 4661 twitter status messages and 7769 tokens are annotated.

4.2 Graph Generation

We used a large corpus of social media text to construct our word association graph. We extracted 1.5 GB of English tweets from Stanford's 476 million Twitter Dataset (Yang and Leskovec, 2011). The language identification of tweets was performed by using the langid.py Python library (Lui and Baldwin, 2012; Baldwin and Lui, 2010).

CMU Ark Tagger (v0.3.2), which is a social media specific POS tagger achieving an accuracy of 95% over social media text (Owoputi et al., 2013; Gimpel et al., 2011), is used for tokenizing and POS tagging the tweets. We used the twitter tagset which includes some extra POS tags specific to social media including URLs and emoticons, Twitter hashtags (#), and twitter at-mentions (@). We made use of these social media specific tags to disambiguate some OOV tokens.

After tokenization, we removed the tokens that were POS tagged as mention (e.g. @brendon), discourse marker (e.g. RT), URL, email address, emoticon, numeral, and punctuation. The remaining tokens are used to build the word association graph. After constructing the graph we only kept the nodes with a frequency greater than 8. For the performance related reasons, the relatedness thresholds $t_{distance}$ and $t_{frequency}$ were chosen as 3 and 8, respectively. The resulting graph contains 105428 nodes and 46609603 edges.

4.3 Candidate Set Generation

While extending the candidate set with lexical features we use $t_{edit} \leq 2 \ \vee \ t_{phonetic} \leq 1$ to keep up with the settings in (Han and Baldwin, 2011). In other words, IV words that are within 2 character edit distance or 1 character edit distance of a given OOV word under phonemic transcription were chosen as lexical similarity candidates. The values for the λ and β parameters in Equations 3 and 6 are set to 0.5. We did not tune these parameters for optimized performance. We selected the value of 0.5 in order to give less weight (half weight) to our minor contextual and lexical similarity features compared to the major ones.

4.4 Normalization Candidates

Most of the prior work assume perfect detection of ill-formed words during test set decoding (Liu et al., 2012; Han and Baldwin, 2011; Pennell and Liu, 2011; Yang and Eisenstein, 2013). To be able to compare our results with studies that do not assume that ill-formed words have been preidentified (Chrupala, 2014; Hassan and Menezes, 2013; Han et al., 2012) we used our graph and built a dictionary to identify the ill-formed words.

Following Han and Baldwin (2011) and Yang and Eisenstein (2013), we created a dictionary by choosing the nodes in our graph that have a frequency property higher than 20. Filtering this dictionary of 49657 words using GNU Aspell dictionary (v0.60.6) we produced a set of 26773 "invocabulary" (IV) words. In our second setup our system does not attemp to normalize the words in this set.

4.5 Results and Analysis

In this paper we introduced a new contextual approach for text normalization. The lexical similarity score described in Section 3.4 and the external score described in Section 3.5 depend on the work of Han and Baldwin (2011). With small changes made to the previously proposed method we took it as a baseline in our study.

As contextual layer we proposed two metrics extracted from the word association graph. The first one depends on the total edge weights between candidates and OOV neighbours, the second one is based on the frequencies of the candidates in the corpus.

As the evaluation metrics we used precision, recall, and F-Measure. Precision calculates the proportion of correctly normalized words among the words for which we produced a normalization. Recall shows the amount of correct normalizations over the words that require normalization (ill-formed OOV words). The main metric that we consider while evaluating the performance of our system is F-Measure which is the harmonic mean of precision and recall.

We investigated the impact of lexSimScore and externalScore seperately on both datasets (Table 5). Using only lexSimScore the system achieved an F-measure of 28.24% on the LexNorm1.1 dataset and 38.70% on the Trigram dataset, which shows that lexical similarity alone is not enough for a good normalization system.

However, the externalScore which is the layer that is more aware of the Internet jargon, along with some social text specific rule based transliterations performs better than expected on both datasets. Mixing these two layers we reach our baseline that is adopted from (Han and Baldwin, 2011). This baseline setup obtained an F-measure of 77.12% on LexNorm1.1, which is slightly better than the result (75.30%) reported by the original system of Han and Baldwin (2011).

The results obtained by our proposed Contextual Word Association Graph (CWA-Graph) system on the LexNorm1.1 and trigram datasets, as well as the results of recent studies that used the same datasets for evaluation are presented in Table 5. The ill-formed words are assumed to have been pre-identified in advance.

Method	Dataset	Precision	Recall	F-measure
lexSimScore	LN	28.28	28.20	28.24
externalScore	LN	64.69	64.52	64.60
lexSimScore+externalScore	LN	77.22	77.02	77.12
Han and Baldwin (2011)	LN	75.30	75.30	75.30
Liu *et al.* (2012)	LN	84.13	78.38	81.15
Yang and Eisenstein (2013)	LN	82.09	**82.09**	82.09
CWA-Graph	LN	**85.50**	79.22	**82.24**
lexSimScore	Trigram	39.10	38.40	38.70
externalScore	Trigram	44.20	43.30	43.80
lexSimScore+externalScore	Trigram	65.50	64.20	64.80
Pennell and Liu (2011)	Trigram	69.7	**69.7**	69.7
CWA-Graph	Trigram	**77.2**	68.8	**72.8**

Table 5: Results obtained when ill-formed words are assumed to have been pre-identified in advance.

Our CWA-Graph approach achieves the best F-measure (82.24%) and precision (85.50%) among the recent previous studies. The high precision value is obtained without compromising much from recall (79.22%). Our recall is the second best among others. The F-score (82.09%) obtained by Yang and Eisenstein (2013)'s system is close to ours and the second best F-score, which on the other hand, has a lower precision.

Without any modification to our system or to the parameters, we were able to improve the results obtained by Pennell and Liu (2011) on the trigram SMS-like dataset. The trigram nature of the dataset resulted in input texts which are (short thus) very limited with regard to contextual information. Nevertheless, our system achieved 72.8% F-Measure using this contextual information even though it is limited.

Along the systems (presented in Table 5) that assume ill-formed tokens have been pre-identified perfectly by an oracle, there are also systems that are not based on this assumption but contain ill-formed word identification components (Han et al., 2012; Hassan and Menezes, 2013; Chrupala, 2014). We used the method described in Section 4.4 to identify the candidate tokens for normalization. Table 6 shows our results compared with the results of other systems that perform ill-formed word detection prior to normalization. We could label 1141 tokens correctly as ill-formed among 1184 ill-formed tokens. We achieved a word error rate (WER) of 2.6%, where Chrupala (2014) reported 4.8% and Han et al. (2012) reported 6.6% WER on the Lexnorm1.1 dataset.

Method	Dataset	Precision	Recall	F-measure
Han et al. (2012)	LN	70.00	17.90	28.50
Hassan and Menezes (2013)	LN	85.37	56.40	69.93
CWA-Graph	LN	**85.87**	**76.52**	**80.92**

Table 6: Results obtained without assuming that ill-formed words have been pre-identified.

As shown in Table 5 some systems have equal precision and recall values (Yang and Eisenstein, 2013; Han and Baldwin, 2011; Pennell and Liu, 2011). Those systems normalize all ill-formed words. On the other hand, our system does not return a normalization, if there are no candidates that are lexically similar, grammatically correct, and contextually close enough. For this reason, we managed to achieve a higher precision compared to the other systems. Our system returns a normalization candidate for an OOV word only if it achieves a similarity score (contextual, lexical, external, or some degree of each feature) above a threshold value. The default threshold used in the system is set equal to the maximum score that can be obtained by lexical features. Thus, we only retrieve candidates that obtain a non-zero contextual similarity score (conSimScore). The results shown at Table 7 and Table 8 demonstrate that CWA-Graph can obtain even higher values of precision by increasing the percentage of contextual context of candidates. It achieved 94.1% precision on the LexNorm1.1 dataset, where the highest precision reported at the same recall level is 85.37% (Hassan and Menezes, 2013). The precision of the normalization system can be set (e.g. as high, medium, low) depending on the application where it will be used.

Our motivation behind introducing the λ and β parameters was to investigate the importance

conSimScore >	Precision	Recall	F-measure
0	**85.5**	**79.2**	**82.2**
0.1	88.8	75.1	81.4
0.2	91.1	72.8	80.9
0.3	92.3	67.6	78.0
0.5	94.1	56.4	70.5

Table 7: Comparison of results for different threshold values on LexNorm1.1, the setup we have used for our other experiments is shown in bold.

conSimScore >	Precision	Recall	F-measure
0	**77.2**	**68.8**	**72.8**
0.1	80.9	65.8	72.6
0.2	84.2	60.8	70.6
0.3	87.6	54.6	67.3
0.4	89.5	47.1	61.7
0.5	90.8	42.1	57.6

Table 8: Comparison of results for different threshold values on trigram dataset, the setup we have used for our other experiments is shown in bold.

of the minor features compared to our major features (described in Sections 3.3 and 3.4). For the experiments reported in Tables 5, 6, 7 and 8 we set the λ and β values to 0.5. We did not tune these parameters for optimized performance. Rather, our aim was to give less weight (half weight) to the minor features compared to the major ones. To analyze the effects of the lambda and beta parameters, we plotted the performance of the system on the LexNorm1.1 data set by varying their values (see Figure 4). It is shown that for λ and β values greater than 0.3 the performance of the system is quite robust. The F-score varies between 80.4% and 82.9%.

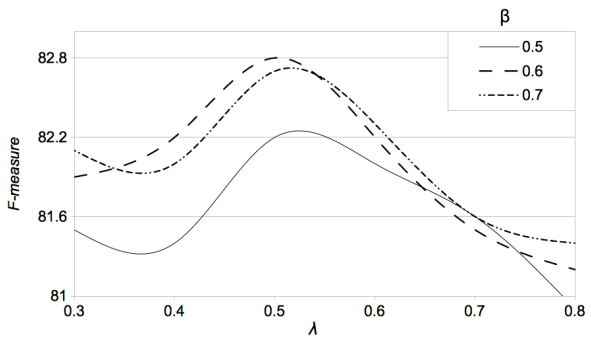


Figure 4: The effect of λ and β on the system performance.

5 Conclusion

In this paper, we present an unsupervised graph-based approach for contextual text normalization. The task of normalization is highly dependent on understanding and capturing the dynamics of the informal nature of social text. Our word association graph is built using a large unlabeled social media corpus. It helps to derive contextual analysis on both clean and noisy data.

It is important to emphasize the difference between corpus based contextual information and contextual information of the input text (input context). Most recent unsupervised systems for text normalization only make use of corpus based context information. However, this approach is led by statistical information. In other words, it finds which IV word the OOV word is commonly normalized to, regardless of the context of the OOV word in the input text message. A major strength of our approach is that it utilizes both corpus based contextual information and input based contextual information. We use corpus based statistical information to connect/associate the words in the contextual word association graph. On the other hand, the neighbors of an OOV word in the input text provide us input based context information. Using input context to find normalizations helps us identify the correct normalization, even if it is not the statistically dominant one.

We compared our approach with the recent social media text normalization systems and achieved state-of-the-art precision and F-measure scores. We reported our results on two datasets. The first one is the standard text normalization dataset (Lexnorm1.1) derived from Twitter. Our results on this dataset showed that our system can serve as a high precision text normalization system which is highly preferable as an NLP preprocessing step. The second dataset we tested our approach is a SMS-like trigram dataset. The tests showed that the proposed system can perform good on SMS data as well.

The system does not require a clean corpus or an annotated corpus. The contextual word association graph can be built by using the publicly available social media text.

References

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A Phrase-based Statistical Model for SMS Text Nor-

malization. *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 33–40.

Timothy Baldwin and Marco Lui. 2010. Language Identification: The Long and the Short of the Matter. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 229–237.

Eric Brill and Robert C. Moore. 2000. An Improved Error Model for Noisy Channel Spelling Correction. *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. 2007. Investigation and Modeling of the Structure of Texting Language. *International Journal on Document Analysis and Recognition*, 10(3):157–174.

Grzegorz Chrupala. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. *Proceedings of the 52st Annual Meeting of the Association for Computational Linguistics*, pages 680–686.

Danish Contractor, Tanveer A. Faruquie, and L. Venkata Subramaniam. 2010. Unsupervised Cleansing of Noisy Text. *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 189–196.

Paul Cook and Suzanne Stevenson. 2009. An Unsupervised Model for Text Message Normalization. *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 71–78.

Jacob Eisenstein. 2013. What to Do About Bad Language on the Internet. *Proceedings of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 359–369.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech Tagging for Twitter: Annotation, Features, and Experiments. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, pages 42–47.

Stephan Gouws, Donald Metzler, Congxing Cai, and Eduard Hovy. 2011. Contextual Bearing on Linguistic Variation in Social Media. *Proceedings of the Workshop on Languages in Social Media*, pages 20–29.

Bo Han and Timothy Baldwin. 2011. Lexical Normalisation of Short Text Messages: Makn Sens a #Twitter. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 368–378.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 421–432.

Hany Hassan and Arul Menezes. 2013. Social Text Normalization Using Contextual Graph Random Walks. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1577–1586.

Max Kaufmann and Jugal Kalita. 2010. Syntactic Normalization of Twitter Messages. *Proceedings of the 8th International Conference on Natural Language Processing*, pages 149–158.

Vladimir Iosifovich Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Fei Liu, Fuliang Weng, and Xiao Jiang. 2012. A Broad-Coverage Normalization System for Social Media Language. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1035–1044.

Marco Lui and Timothy Baldwin. 2012. Langid.Py: An Off-the-shelf Language Identification Tool. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 25–30.

I. Dan Melamed. 1999. Bitext Maps and Alignment via Pattern Recognition. *Computational Linguistics*, 25(1):107–130.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters. *Proceedings of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 380–390.

Deana Pennell and Yang Liu. 2011. A Character-Level Machine Translation Approach for Normalization of SMS Abbreviations. *Fifth International Joint Conference on Natural Language Processing*, pages 974–982.

Deana Pennell and Yang Liu. 2014. Normalization of informal text. *Computer Speech & Language*, 28(1):256 – 277.

Lawrence Philips. 2000. The Double Metaphone Search Algorithm. *C/C++ Users Journal*, 18(6):38–43, June.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180.

Richard Sproat, Alan W. Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. 2001. Normalization of Non-Standard Words. *Computer Speech & Language*, 15(3):287–333.

Kristina Toutanova and Robert C. Moore. 2002. Pronunciation Modeling for Improved Spelling Correction. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 144–151.

Yi Yang and Jacob Eisenstein. 2013. A Log-Linear Model for Unsupervised Text Normalization. *Proceedings of the Empirical Methods on Natural Language Processing*, pages 61–72.

Jaewon Yang and Jure Leskovec. 2011. Patterns of Temporal Variation in Online Media. *Proceedings of the Forth International Conference on Web Search and Web Data Mining*, pages 177–186.

ReNoun: Fact Extraction for Nominal Attributes

Mohamed Yahya*
Max Planck Institute for Informatics
myahya@mpi-inf.mpg.de

Steven Euijong Whang, Rahul Gupta, Alon Halevy
Google Research
{swhang,grahul,halevy}@google.com

Abstract

Search engines are increasingly relying on large knowledge bases of facts to provide direct answers to users' queries. However, the construction of these knowledge bases is largely manual and does not scale to the long and heavy tail of facts. Open information extraction tries to address this challenge, but typically assumes that facts are expressed with verb phrases, and therefore has had difficulty extracting facts for noun-based relations.

We describe ReNoun, an open information extraction system that complements previous efforts by focusing on nominal attributes and on the long tail. ReNoun's approach is based on leveraging a large ontology of noun attributes mined from a text corpus and from user queries. ReNoun creates a seed set of training data by using specialized patterns and requiring that the facts mention an attribute in the ontology. ReNoun then generalizes from this seed set to produce a much larger set of extractions that are then scored. We describe experiments that show that we extract facts with high precision and for attributes that cannot be extracted with verb-based techniques.

1 Introduction

One of the major themes driving the current evolution of search engines is to make the search experience more efficient and mobile friendly for users by providing them concrete answers to queries. These answers, that apply to queries about entities that the search engine knows about (e.g., famous individuals, organizations or locations) complement the links that the search engine typically returns (Sawant and Chakrabati, 2013; Singhal, 2012; Yahya et al., 2012). To support such answers, the search engine maintains a knowledge base that describes various attributes of an entity (e.g., `(Nicolas Sarkozy, wife, Carla Bruni)`). Upon receiving a query, the search engine tries to recognize whether the answer is in its knowledge base.

For the most part, the aforementioned knowledge bases are constructed using manual techniques and carefully supervised information extraction algorithms. As a result, they obtain high coverage on head attributes, but low coverage on tail ones, such as those shown in Table 1. For example, they may have the answer for the query *"Sarkozy's wife"*, but not for *"Hollande's ex-girlfriend"* or *"Google's philanthropic arm"*. In addition to broadening the scope of query answering, extending the coverage of the knowledge base to long tail attributes can also facilitate providing *Web answers* to the user. Specifically, the search engine can use lower-confidence facts to *corroborate* an answer that appears in text in one of the top Web results and highlight them to the user.

This paper describes *ReNoun*, an open-information extraction system that focuses on extracting facts for long tail attributes. The observation underlying our approach is that attributes from the long tail are typically expressed as *nouns*, whereas most previous work on open-information extraction (e.g., Mausam et al. (2012)) extend techniques for extracting attributes expressed in *verb* form. Hence, the main contribution of our work is to develop an extraction system that complements previous efforts, focuses on nominal attributes and is effective for the long tail. To that end, ReNoun begins with a large but imperfect ontology of nominal attributes that is extracted from text and the query stream (Gupta et al., 2014). ReNoun proceeds by using a small set of high-precision extractors that exploit the nominal na-

*Work done during an internship at Google Research.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 325–335,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Attribute	Fact	Phrase	Verb form seen
`legal affairs correspondent`	`(NPR, legal affairs correspondent, Nina Totenberg)`	NPR welcomed Nina Totenberg as its new legal affairs correspondent.	✗
`economist`	`(Princeton, economist, Paul Krugman)`	Princeton economist Paul Krugman was awarded the Nobel prize in 2008.	✗
`ex-boyfriend`	`(Trierweiler, ex-boyfriend, Hollande`	Trierweiler did not have any children with her ex-boyfriend Hollande.	✓
`staff writer`	`(The New Yorker, staff writer, Adam Gopnik)`	Adam Gopnik is one of The New Yorker's best staff writers.	✓

Table 1: Examples of noun phrases as attributes, none which are part of a verb phrase. Additionally, the first two attributes do not occur within a verb phrase in a large corpus (see § 2 for details) in a setting where they can be associated with a triple.

ture of the attributes to obtain a training set, and then generalizes from the training set via distant supervision to find a much larger set of extraction patterns. Finally, ReNoun scores extracted facts by considering how frequently their patterns extract triples and the *coherence* of these patterns, i.e., whether they extract triples for semantically similar attributes. Our experiments demonstrate that ReNoun extracts a large body of high precision facts, and that these facts are not extracted with techniques based on verb phrases.

2 Preliminaries

The goal of ReNoun is to extract triples of the form `(S,A,O)`, where `S` is *subject*, `A` is the *attribute*, and `O` is the *object*. In our setting, the attribute is always a noun phrase. We refer to the subject and object as the *arguments* of the attribute.

ReNoun takes as input a set of attributes, which can be collected using the methods described in Gupta el al. (2014), Lee et al. (2012), and Pasca and van Durme (2007). In this work, we use Biperpedia (Gupta et al., 2014), which is an ontology of nominal attributes automatically extracted from Web text and user queries. For every attribute, Biperpedia supplies the Freebase (Bollacker et al., 2008) domain type (e.g., whether the attribute applies to people, organizations or hotels). Since the attributes themselves are the result of an extraction algorithm, they may include false positives (i.e., attributes that do not make sense).

The focus of ReNoun is on attributes whose values are concrete objects (e.g., `wife`, `protege`, `chief-economist`). Other classes of attributes that we do not consider in this work are (1) numeric (e.g., `population`, `GDP`) that are better extracted from Web tables (Cafarella et al., 2008), and (2) vague (e.g., `culture`, `economy`) whose value is a narrative that would not fit the current mode of query answering on search engines.

We make the distinction between the *fat head* and *long tail* of attributes. To define these two sets, we ordered the attributes in decreasing order of the number of occurrences in the corpus [1]. We defined the fat head to be the attributes until the point N in the ordering such that the sum of the total number of occurrences of attributes before N equaled the number of total occurrences of the attributes after N. In our news corpus, the fat head included 218 attributes (i.e., $N = 218$) and the long tail included 60K attributes. Table 2 shows examples from both.

Fat head	daughter, headquarters president, spokesperson,
Long tail	chief economist, defender, philanthropic arm, protege

Table 2: Examples of fat head and long tail attributes.

The output of ReNoun is a set of *facts*, where each fact could be generated by multiple extractions. We store the provenance of each extraction and the number of times each fact was extracted.

Noun versus verb attributes

ReNoun's goal is to extract facts for attributes expressed as noun phrases. A natural question is whether we can exploit prior work on open information extraction, which focused on extracting relations expressed as verbs. For example, if we can extract facts for the attribute `advised` or `is advisor of`, we can populate the noun attribute `advisor` with the same facts. In Section 7.2 we demonstrate that this approach is limited for several reasons.

First, attributes in knowledge bases are typically expressed as noun phrases. Table 3 shows that

[1] The occurrences were weighted by the number of semantic classes they occur with in the ontology because many classes overlap.

Knowledge Base	%Nouns	%Verbs
Freebase	97	3
DBpedia	96	4

Table 3: Percentage of attributes expressed as nouns phrases among the 100 attributes with the most facts.

the vast majority of the attributes in both Freebase and DBpedia (Auer et al., 2007) are expressed as nouns even for the fat head (and even more so for the long tail). Hence, if we extract the verb form of attributes we would need to translate them into noun form, which would require us to solve the paraphrasing problem and introduce more sources of error (Madnani and Dorr, 2010). Second, as we dig deeper into the long tail, attributes tend to be expressed in text more in noun form rather than verb form. One of the reasons is that the attribute names tend to get longer and therefore unnatural to express as verbs (e.g. *chief privacy officer*, *automotive division*). Finally, there is often a subtle difference in meaning between verb forms and noun forms of attributes. For example, it is common to see the phrase *"Obama advised Merkel on saving the Euro,"* but that would not necessarily mean we want to say that *Obama* is an advisor of *Angela Merkel*, in the common sense of `advisor`.

Processed document corpus

ReNoun extracts facts from a large corpus of 400M news articles. We exploit rich synactic and linguistic cues, by processing these documents with a natural language processing pipeline comprising of – dependency parsing, noun phrase chunking, named entity recognition, coreference resolution, and entity resolution to Freebase. The chunker identifies nominal mentions in the text that include our attributes of interest. As discussed later in the paper, we exploit the dependency parse, coreference and entity resolution heavily during various stages of our pipeline.

3 Overview of ReNoun

Since ReNoun aims at extracting triples for attributes not present in head-heavy knowledge bases, one key challenge is that we do not have any labeled data (i.e. known facts) for such attributes, especially in the long tail. Therefore ReNoun has an initial seed fact extraction step that automatically generates a small corpus of relatively precise seed facts for all attributes, so that distant supervision can be employed. The second big challenge is to filter the noise from the resulting extractions.

ReNoun's extraction pipeline, shown in Figure 1, is composed of four stages.

Seed fact extraction: We begin by extracting a small number of high-precision facts for our attributes. For this step, we rely on manually specified lexical patterns that are specifically tailored for noun phrases, but are general enough to be independent of any specific attributes. When applying such patterns, we exploit coreference to make the generated seed facts more precise by requiring the attribute and object noun phrases of a seed fact to refer to the same real-world entity. This is elaborated further in Section 4.

Extraction pattern generation: Utilizing the seed facts, we use distant supervision (Mintz et al., 2009) to learn a set of dependency parse patterns that are used to extract a lot more facts from the text corpus.

Candidate generation: We apply the learned dependency parse patterns from the previous stage to generate a much larger set of extractions. We aggregate all the extractions that give rise to the same fact and store with them the provenance of the extraction. The extractions generated here are called candidates because they are assigned scores that determine how they are used. The application consuming an extraction can decide whether to discard an extraction or use it, and in this case the manner in which it is used, based on the scores we attach to it and the application's precision requirements.

Scoring: In the final stage, we score the facts, reflecting our confidence in their correctness. Intuitively, we give a pattern a high score if it extracts many facts that have semantically similar attributes, and then propagate this score to the facts extracted by the pattern (Section 6).

4 Seed fact extraction

Since we do not have facts, but only attributes, the first phase of ReNoun's pipeline is to extract a set of high-precision seed facts that are used to train more general extraction patterns. ReNoun extracts seed facts using a manually crafted set of extraction rules (see Table 4). However, the extraction rules and the application of these rules are tailored to our task of extracting noun-based attributes.

Specifically, when we apply an extraction rule to generate a triple `(S,A,O)`, we require that (1) `A` is an attribute in our ontology, and (2) the value of `A` and the object `O` corefer to the same real-world

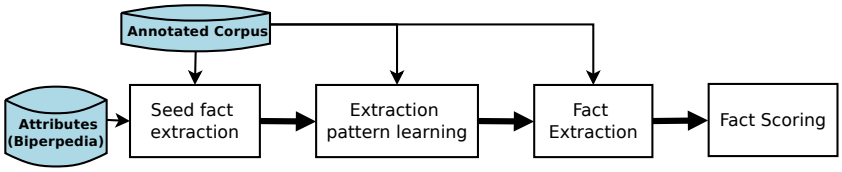


Figure 1: Extraction Pipeline: we begin with a set of high-precision extractors and use distant supervision to train other extractors. We then apply the new extractors and score the resulting triples based on the frequency and coherence of the patterns that produce them.

1. *the* A *of* S, O – the CEO of Google, Larry Page
2. *the* A *of* S *is* O – the CEO of Google is Larry Page
3. O, S A – Larry Page, Google CEO
4. O, S*'s* A – Larry Page, Google's CEO
5. O, [*the*] A *of* S – Larry Page, [the] CEO of Google
6. SAO – Google CEO Larry Page
7. S A, O – Google CEO, Larry Page
8. S*'s* A, O – Google's CEO, Larry Page

Table 4: High precision patterns used for seed fact extraction along with an example of each. Here, the object (O) and the attribute (A) corefer and the subject (S) is in close proximity. In all examples, the resulting fact is `(Google, CEO, Larry Page)`. Patterns are not attribute specific.

entity. For example, in Figure 2, `CEO` is in our ontology and we can use a coreference resolver to infer that `CEO` and `Larry Page` refer to the same entity. The use of coreference follows from the simple observation that objects will often be referred to by nominals, many of which are our attributes of interest. Since the sentence matches our sixth extraction rule, ReNoun extracts the triple `(Google, CEO, Larry Page)`.

Document:
"[Google]$_1$ [CEO]$_2$ [Larry Page]$_2$ started his term in 2011, when [he]$_2$ succeeded [Eric Schmidt]$_3$. [Schmidt]$_3$ has since assumed the role of executive chairman of [the company]$_1$."

(a)

Coreference clusters:

#	Phrases	Freebase ID
1	**Google**, the company	/m/045c7b
2	**Larry Page**, CEO, he	/m/0gjpq
3	**Eric Schmidt**, Schmidt	/m/01gqf4

(b)

Figure 2: Coreference clusters: (a) a document annotated with coreference clusters; (b) a table showing each cluster with the representative phrases in bold and the Freebase ID to which each cluster maps.

We rely on a coreference resolver in the spirit of Haghighi and Klein (2009). The resolver clusters the mentions of entities in a document so the references in each cluster are assumed to refer to the same real-world entity. The resolver also chooses for each cluster a *representative phrase*, which is a proper noun or proper adjective (e.g., *Canadian*). Other phrases in the same cluster can be other proper nouns or adjectives, common nouns like *CEO* or pronouns like *he* in the example. Each cluster is possibly linked by an entity resolver to a Freebase entity using a unique Freebase ID. Figure 2(b) shows the coreference clusters from the sample document, with representative phrases in bold, along with the Freebase ID of each cluster. Note that in our example the phrase *executive chairman*, which is also in our ontology of attributes, is not part of any coreference cluster. Therefore, the fact centered around this attribute in the example will not be part of the seed extractions, but could be extracted in the next phase. The resulting facts use Freebase IDs for the subject and object (for readability, we will use entity names in the rest of this work). In summary, our seed extraction proceeds in two steps. First, we find sentences with candidate attribute-object pairs that corefer and in which the attribute is in our ontology. Second, we match these sentences against our hand-crafted rules to generate the extractions. In Section 7 we show that the precision of our seed facts is 65% for fat head attributes and 80% for long tail ones.

5 Pattern and candidate fact generation

In this section we describe how ReNoun uses the seed facts to learn a much broader set of extraction patterns. ReNoun uses the learned patterns to extract many more candidate facts that are then assigned scores reflecting their quality.

5.1 Dependency patterns

We use the seed facts to learn patterns over dependency parses of text sentences. A dependency parse of a sentence is a directed graph whose vertices correspond to tokens labeled with the word and the POS tag, and the edges are syntactic relations between the corresponding tokens (de Marneffe et al., 2006). A *dependency pattern* is a subgraph of a dependency parse where some words have been replaced by variables, but the POS tags

have been retained (called *delexicalization*). A dependency pattern enables us to extract sentences with the same dependency parse as the sentence that generated the pattern, modulo the delexicalized words. We note that one big benefit of using dependency patterns is that they generalize well, as they ignore extra tokens in the sentence that do not belong to the dependency subgraph of interest.

5.2 Generating dependency patterns

The procedure for dependency pattern generation is shown in Algorithm 1, and Figure 3 shows an example of its application. The input to the algorithm is the ontology of attributes, the seed facts (Section 4), and our processed text corpus (Section 2).

Algorithm 1: Dependency pattern generation

input : Set of attributes $\mathcal{A}$, Seed facts I, Corpus D.
$\mathcal{P}$:= An empty set of dependency pattern-attribute pairs.
foreach *sentence* $s \in D$ **do**
  foreach *triple* $t = (S, A, O)$ *found in* s **do**
    if $t \in I$ **then**
      $G(s)$ = dependency parse of s
      P' = minimal subgraph of $G(s)$ containing the head tokens of S, A and O
      P = Delexicalize(P', S, A, O)
      $\mathcal{P} = \mathcal{P} \cup \{\langle P, A \rangle\}$
return $\mathcal{P}$

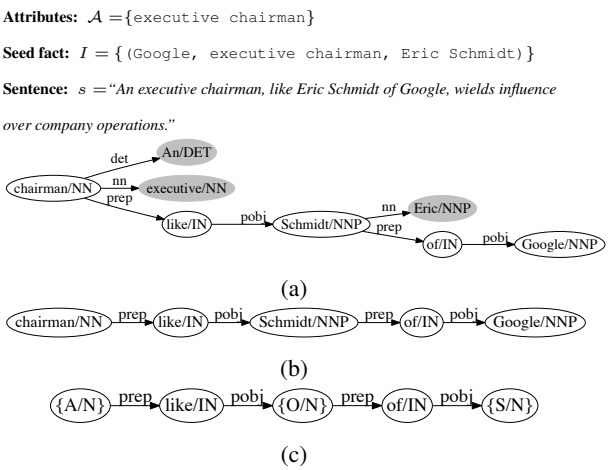


Figure 3: Dependency pattern generation using seed facts, corresponding to Algorithm 1: (a) shows the input to the procedure (dependency parse partially shown); (b) P'; (c) P.

The procedure iterates over the sentences in the corpus, looking for matches between a sentence s and a seed fact f. A sentence s matches f if s contains (i) the attribute in f, and (ii) phrases in coreference clusters that map to the same Freebase IDs as the subject and object of f. When a match is found, we generate a pattern as follows.

We denote by P' the minimal subgraph of the dependency parse of s containing the head tokens of the subject, attribute and object (Figure 3 (b)). We delexicalize the three vertices corresponding to the head tokens of the subject, attribute and object by variables indicating their roles. The POS tag associated with the attribute token is always a noun. The subject and object are additionally allowed to have pronouns and adjectives associated with their tokens. All POS tags corresponding to nouns are lifted to N, in order to match the various types of nouns. We denote the resulting dependency pattern by P and add it to our output, associated with the matched attribute. We note that in principle, the vertex corresponding to the head of the attribute does not need to be delexicalized. However, we do this to improve the efficiency of pattern-matching, since we will often have patterns for different attributes differing only at the attribute vertex.

It is important to note that because of the manner in which the roles of subject and object were assigned during seed fact extraction, the patterns ReNoun generates clearly show which argument will take the role of the subject, and which will take the role of the object. This is in contrast to previous work such as Ollie (Mausam at al., 2012), where the assignment depends on the order in which the arguments are expressed in the sentence from which the fact is being extracted. For example, from the sentence *"Opel was described as GM's most successful subsidiary."* and the seed fact `(GM, subsidiary, Opel)`, the pattern that ReNoun generates will consistently extract facts like `(BMW, subsidiary, Rolls-Royce)`, and not the incorrect inverse, regardless of the relative ordering of the two entities in the sentence.

At this point we have dependency patterns capable of generating more extractions for their seed fact attributes. For efficient matching, we use the output of Algorithm 1 to generate a map from dependency patterns to their attributes with entries like that shown in Figure 4(a). This way, a pattern match can be propagated to all its mapped attributes in one shot, as we explain in Section 5.3. Finally, we discard patterns that do not pass a support threshold, where support is the number of distinct seed facts from which a pattern could be generated.

attributes: {executive chairman, creative director, ...}

(a)

"An executive chairman, like Steve Chase of AOL, is responsible for representing the company."

↓

(AOL, executive chairman, Steve Chase)

(b)

"A creative director, like will.i.am of 3D Systems, may also be referred to as chief creative officer."

↓

(3D Systems, creative director, will.i.am)

(c)

Figure 4: A dependency pattern and its use in extraction: (a) the pattern in our running example and the set of attributes to which it applies; (b) and (c) sentences matching the pattern and the resulting extractions.

attributes: {ex-wife, boyfriend, ex-partner}

$frequency(P) = 574, coherence(P) = 0.429$

Example: *"Putin has two children with his ex-wife, Lyudmila."*

(a)

attributes: {ex-wife, general manager, subsidiary,... }

$frequency(P) = 52349038, coherence(P) = 0.093$

Example: *"Chelsea F.C. general manager José Mourinho..."*

(b)

Figure 5: (a) a coherent pattern extracting facts for semantically similar attributes and (b) an incoherent pattern.

5.3 Applying the dependency patterns

Given the learned patterns, we can now generate new extractions. Each match of a pattern against the corpus will indicate the heads of the potential subject, attribute and object. The noun phrase headed by the token matching the {A/N} vertex is checked against the set of attributes to which the pattern is mapped. If the noun phrase is found among these attributes, then a triple (S, A, O) is constructed from the attribute and the Freebase entities to which the tokens corresponding to the S and O nodes in the pattern are resolved. This triple is then emitted as an extraction along with the pattern that generated it. Figure 4(b) and (c) show two sentences that match the dependency pattern in our running example and the resulting extractions.

Finally, we aggregate our extractions by their generated facts. For each fact f, we save the distinct dependency patterns that yielded f and the total number of times it was found in the corpus.

6 Scoring extracted facts

In this section we describe how we score the candidate facts extracted by applying the dependency patterns in Section 5. Recall that a fact may be obtained from multiple extractions, and assigning scores to each fact (rather than each extraction) enables us to consider all extractions of a fact in aggregate.

We score facts based on the patterns which extract them. Our scheme balances two characteristics of a pattern: its frequency and coherence. *Pattern frequency* is defined as the number of extractions produced by the pattern. Our first observation is that patterns with a large number of extractions are always able to produce correct extractions (in addition to incorrect ones). We also observe that generic patterns produce more erroneous facts compared to more targeted ones. To capture this, we introduce *pattern coherence*, which reflects how targeted a pattern is based on the attributes to which it applies. For example, we observed that if an extraction pattern yields facts for the coherent set of attributes ex-wife, boyfriend, and ex-partner, then its output is consistently good. On the other hand, a pattern that yields facts for a less coherent set of attributes ex-wife, general manager, and subsidiary is more likely to produce noisy extractions. Generic, more incoherent patterns are more sensitive to noise in the linguistic annotation of a document. Figure 5 shows an example pattern for each case, along with its frequency and coherence.

We capture coherence of attributes using word-vector representations of attributes that are created over large text corpora (Mikolov et al., 2013). The word-vector representation $v(w)$ for a word w (multi-word attributes can be preprocessed into single words) is computed in two steps. First, the algorithm counts the number of occurrences of a word w_1 that occurs within the text window centered at w (typically a window of size 10), producing an intermediate vector that potentially has a non-zero value for every word in the corpus. The intermediate vector is then mapped to a much smaller dimension (typically less than 1000) to produce $v(w)$. As shown in (Mikolov et al., 2013), two words w_1 and w_2 for which the cosine dis-

tance between $v(w_1)$ and $v(w_2)$ is small tend to be semantically similar. Therefore, a pattern is coherent if it applies to attributes deemed similar as per their word vectors.

Given an extraction pattern P that extracts facts for a set of attributes $\mathcal{A}$, we define the coherence of P to be the average pairwise coherence of all attributes in $\mathcal{A}$, where the pairwise coherence of two attributes a_1 and a_2 is the cosine distance between $v(a_1)$ and $v(a_2)$.

Finally, we compute the score of a fact f by summing the product of frequency and coherence for each pattern of f as shown in Equation 1.

$$S(f) = \sum_{P \in Pat(f)} frequency(P) \times coherence(P) \quad (1)$$

7 Experimental Evaluation

We describe a set of experiments that validate the contributions of ReNoun. In Sections 7.2 and 7.3 we validate our noun-centric approach: we show that extractions based on verb phrases cannot yield the results of ReNoun and that NomBank, the resource used by state of the art in semantic role-labeling for nouns, will not suffice either. In Sections 7.4-7.6 we evaluate the different components of ReNoun and its overall quality, and in Section 7.7 we discuss the cases in which ReNoun was unable to extract any facts.

7.1 Setting

We used the fat head (FH) and long tail (LT) attributes and annotated news corpus described in Section 2. When evaluating facts, we used majority voting among three human judges, unless otherwise noted. The judges were instructed to consider facts with inverted subjects and objects as incorrect. For example, while `(GM, subsidiary, Opel)` is correct, its inverse is incorrect.

7.2 Verb phrases are not enough

State-of-art open information extraction systems like Ollie (Mausam at al., 2012) assume that a relation worth extracting is expressed somewhere in verb form. We show this is not the case and justify our noun-centric approach. In this experiment we compare ReNoun to a custom implementation of Ollie that uses the same corpus as ReNoun and supports multi-word attributes. While Ollie does try to find relations expressed as nouns, its seed facts are relations expressed as verbs.

We randomly sampled each of FH and LT for 100 attributes for which ReNoun extracts facts and asked a judge to find potentially equivalent Ollie relations. Note that we did not require the judge to find exactly the same triple (thereby biasing the experiment towards finding more attribute matches). Furthermore, the judge was instructed that a verb phrase like *advised by* should be considered a match to the ReNoun attribute `advisor`. However, looking at the data, most facts involving the relation *advised* are not synonymous with the *advisor* relation as we think of it (e.g., *"Obama advised Merkel on saving the Euro"*). This observation suggests that there is an even more subtle difference between the meaning of verb expressions and noun-based expressions in text. This experiment, therefore, gives an upper bound on the number of ReNoun attributes that Ollie can cover.

ReNoun	**Ollie**
flagship company	-
railway minister	-
legal affairs correspondent	-
spokesperson	be spokesperson of
president-elect	be president elect of
co-founder	be co-founder of

Table 5: ReNoun attributes with and without a corresponding Ollie relation.

For FH, not surprisingly, we could find matches for 99 of the 100 attributes. However, for LT, only 31 of the 100 attributes could be found, even under our permissive setting. Most attributes that could not be matched were multi-word noun phrases. While in principle, one could use the Ollie patterns that apply to the head of a multi-word attribute, we found that we generate more interesting patterns for specific multi-word attributes. Table 5 shows examples of attributes with and without verb mappings in Ollie.

We also compare in the other direction and estimate the portion of Ollie relations centered around nouns for which ReNoun fails to extract facts. For this experiment, we randomly sampled 100 Ollie relations that contained common nouns whose objects are concrete values, and looked for equivalent attributes in ReNoun extractions. ReNoun extracts facts for 48 of the Ollie relations. Among the 52 relations with no facts, 25 are not in Biperpedia (which means that ReNoun cannot extract facts for them no matter what). For the other 27 relations, ReNoun did not extract facts for the following reasons. First, some relations expressed actions, which cannot be expressed using nouns only, and are not considered attributes describing the subject entity (e.g., `citation of` in *"Obama's citation*

of the Bible"). Second, some relations have the object (a common noun) embedded within them (e.g., `have microphone in`) and do not have corresponding attributes that can be expressed using nouns only. The remaining relations either have meaningless extractions or use common noun phrases as arguments. ReNoun only uses proper nouns (i.e., entities) for arguments because facts with common noun arguments are rarely interesting without more context. We note that the majority of the 25 Ollie relations without corresponding Biperpedia attributes also fall into one of the three categories above.

7.3 Comparison against NomBank

In principle, the task of extracting noun-mediated relations can be compared to that of semantic role labeling (SRL) for nouns. The task in SRL is to identify a relation, expressed either through a verb or a noun, map it to a semantic frame, and map the arguments of the relation to the various roles within the frame. State of the art SRL systems, such as that of Johansson and Nugues (2008), are trained on NomBank (Meyers et al., 2004) for handling nominal relations, which also means that they are limited by the knowledge it has. We asked a judge to manually search NomBank for 100 attributes randomly drawn from each of FH and LT for which ReNoun extracts facts. For multi-word attributes, we declare a match if its head word was found. We were able to find 80 matches for the FH attributes and 42 for LT ones. For example, we could not find entries for the noun attributes `coach` or `linebacker` (of a football team). This result is easy to explain by the fact that NomBank only has 4700 attributes.

In addition, for some nouns, the associated frames do not allow for the extraction of triples. For example, all frames for the noun `member` specify one argument only, so in the sentence *"John became a member of ACM"*, the output relation is `(ACM, member)` instead of the desired triple `(ACM, member, John)`.

As we did with Ollie, we also looked at nouns from NomBank for which ReNoun does not extract facts. Out of a random sample of 100 NomBank nouns, ReNoun did not extract facts for 29 nouns (four of which are not in Biperpedia). The majority of the missed nouns cannot be used by ReNoun because they either take single arguments (instead of two) or take either prepositional phrases or common nouns (instead of proper nouns correponding to entities) as one their arguments.

7.4 Quality of seed facts

In Section 4, we described our method for extracting seed facts for our attributes. Applying the method to our corpus resulted in 139M extractions, which boiled down to about 680K unique facts covering 11319 attributes. We sampled 100 random facts from each of FH and LT, and obtained 65% precision for FH seed facts and 80% precision for LT ones. This leads us to two observations.

First, the precision of seed facts for LT attributes is high, which makes them suitable for use as a building block in a distant supervision scheme to learn dependency parse patterns. We are primarily interested in LT attributes, which earlier approaches cannot deal with satisfactorily as we demonstrated above.

Second, LT attributes have higher precision than FH attributes. One reason is that multi-word attributes (which tend to be in LT) are sometimes incorrectly chunked, and only their head words are recognized as attributes (which are more likely to be in FH). For example, in the phrase *"America's German coach, Klinsmann"*, the correct attribute is `German coach` (LT), but bad chunking may produce the attribute `coach` (FH) with `Germany` as the subject. Another reason is that FH attributes are likely to occur in speculative contexts where the presence of the attribute is not always an assertion of a fact. (While both FH and LT attributes can be subject to speculative contexts, we observe this more for FH than LT in our data.) For example, before a person is a `railway minister` of a country, there is little mention of her along with the attribute. However, before a person is elected `president`, there is more media about her candidacy. Speculative contexts, combined with incorrect linguistic analysis of sentences, can result in incorrect seed facts (e.g., from *"Republican favorite for US president, Mitt Romney, visited Ohio"*, we extract the incorrect seed fact `(US, president, Mitt Romney)`).

7.5 Candidate generation

Using the seed facts, we ran our candidate generation algorithm (Section 5). In the first step of the algorithm we produced a total of about 2 million unique dependency patterns. A third of these

patterns could extract values for exactly one attribute. Manual inspection of these long tail patterns showed that they were either noise, or do not generalize. We kept patterns supported by at least 10 seed facts, yielding more than 30K patterns.

We then applied the patterns to the corpus. The result was over 460M extractions, aggregated into about 40M unique facts. Of these, about 22M facts were for LT attributes, and 18M for FH. We now evaluate the quality of these facts.

7.6 Scoring extracted facts

In Section 6, we presented a scheme for scoring facts using pattern frequency and coherence. To show its effectiveness we (i) compare it against other scoring schemes, and (ii) show the quality of the top-k facts produced using this scheme, for various k. To compute coherence, we generated attribute word vectors using the word2vec[2] tool trained on a dump of Wikipedia.

First, we compare the quality of our scoring scheme (FREQ_COH) with three other schemes as shown in Table 6. The scheme FREQ is identical to FREQ_COH except that all coherences are set to 1. PATTERN counts the number of distinct patterns that extract the fact while PATTERN_COH sums the pattern coherences. We generated a random sample of 252 FH and LT nouns with no entity disambiguation errors by the underlying natural language processing pipeline. The justification is that none of the schemes we consider here capture such errors. Accounting for such errors requires elaborate signals from the entity linking system, which we leave for future work. For each scoring scheme, we computed the Spearman's rank correlation coefficient ρ between the scores and manual judgments (by three judges). A larger ρ indicates more correlation, and computing ρ was statistically significant (p-value$<$0.01) for all schemes.

Scheme	Spearman's ρ
FREQ	0.486
FREQ_COH	0.495
PATTERN	0.265
PATTERN_COH	0.257

Table 6: Scoring schemes

FREQ and FREQ_COH dominate, which shows that considering the frequency with which patterns perform extraction helps. The two schemes, however, are very close to each other. We observed that adding coherence helps when two facts have similar frequencies, but this effect is tempered when considering a large number of facts.

[2]https://code.google.com/p/word2vec/

	FH		LT	
k	Precision	#Attr	Precision	#Attr
10^2	1.00	8	1.00	50
10^3	0.98	36	1.00	294
10^4	0.96	78	0.98	1548
10^5	0.82	106	0.96	5093
10^6	0.74	124	0.70	7821
All	0.18	141	0.26	11178

Table 7: Precision of random samples of the top-k scoring facts, along with the attribute yield.

Second, we evaluate the scoring of facts generated by ReNoun by the precision of top-k results for several values of k. In this evaluation, facts with disambiguation errors are counted as wrong. The particular context in which ReNoun is applied will determine where in the ordering to set the threshold of facts to consider. We compute precision based on a sample of 50 randomly chosen facts for each k. Table 7 shows the precision results, along with the number of distinct attributes (#Attr) for which values are extracted at each k.

As we can see, ReNoun is capable of generating a large number of high quality facts ($\geq$70% precise at 1M), which our scoring method manages to successfully surface to the top. The major sources of error were (i) incorrect dependency parsing mainly due to errors in boilerplate text removal from news documents, (ii) incorrect coreference resolution of pronouns, (iii) incorrect entity resolution against Freebase, and (iv) cases where a triple is not sufficient (e.g., `ambassador` where both arguments are countries.)

7.7 Missed extractions

We analyze why ReNoun does not extract facts for certain attributes. For FH, we investigate all the 77 attributes for which ReNoun is missing facts. For LT, there are about 50K attributes without corresponding facts, and we use a random sample of 100 of those attributes.

Cause	FH	LT	Example
Vague	23	37	`culture`
Numeric	4	26	`rainfall`
Object not KB entity	11	6	`email`
Plural	30	15	`member firms`
Bad attribute / misspell	3	4	`newsies`
Value expected	6	12	`nationality`
Total	77	100	

Table 8: Analysis of attributes with no extractions.

Table 8 shows the categorization of the missed attributes. The first three categories are cases that are currently outside the scope of ReNoun: vague attributes whose values are long narratives, numeric attributes, and typed attributes (e.g., email) whose values are not modeled as Freebase entities. The next two categories are due to limitations of the ontology, e.g., plural forms of attributes are not always synonymized with singular forms and some attributes are bad. Finally, the "Value expected" category contains the attributes for which ReNoun *should* have extracted values. One reason for missing values is that the corpus itself does not contain values of all attributes. Another reason is that some attributes are not verbalized in text. For example, attributes like `nationality` are usually not explicitly stated when expressed in text.

8 Related Work

Open information extraction (OIE) was introduced by Banko et al. (2007). For a pair of noun phrases, their system, TEXTRUNNER, looks for the attribute (or more generally the relation) in the text between them and uses a classifier to judge the trustworthiness of an extraction. WOE^{parse} (Wu and Weld, 2010) extends this by using dependency parsing to connect the subject and object. Both systems assume that the attribute is between its two arguments, an assumption that ReNoun drops since it is not suitable for nominal attributes.

Closest to our work are ReVerb (Fader et al., 2011) and Ollie (Mausam at al., 2012). ReVerb uses POS tag patterns to locate verb relations and then looks at noun phrases to the left and right for arguments. Ollie uses the ReVerb extractions as its seeds to train patterns that can further extract triples. While Ollie's patterns themselves are not limited to verb relations (they also support noun relations), the ReVerb seeds are limited to verbs, which makes Ollie's coverage on noun relations also limited. In comparison, ReNoun take a noun-centric approach and extracts many facts that do not exist in Ollie.clo

ClausIE (Del Corro and Gemulla, 2013) is an OIE framework that exploits knowledge about the grammar of the English language to find clauses in a sentence using its dependency parse. The clauses are subsequently used to generate extractions at multiple granularities, possibly with more than triples. While ClausIE comes with a predefined set of rules on how to extract facts from a dependency parse, ReNoun learns such rules from its seed facts.

Finally, Nakashole et al. (2014) and Mintz et al. (2009) find additional facts for attributes that already have facts in a knowledge base. In contrast, ReNoun is an OIE framework whose goal is to find facts for attributes without existing facts.

9 Conclusions

We described ReNoun, an open information extraction system for nominal attributes that focuses on the long tail. The key to our approach is to start from a large ontology of nominal attributes and apply noun-specific manual patterns on a large preprocessed corpus (via standard NLP components) to extract precise seed facts. We then learn a set of dependency patterns, which are used to generate a much larger set of candidate facts. We proposed a scoring function for filtering candidate facts based on pattern frequency and coherence. We demonstrated that the majority of long tail attributes in ReNoun do not have corresponding verbs in Ollie. Finally, our experiments show that our scoring function is effective in filtering candidate facts (top-1M facts are $\geq$70% precise).

In the future, we plan to extend ReNoun to extract triples whose components are not limited to Freebase IDs. As an example, extending ReNoun to handle numerical or typed attributes would involve extending our extraction pattern learning to accommodate units (e.g., kilograms) and other special data formats (e.g., addresses).

Acknowledgments

We would like to thank Luna Dong, Anjuli Kannan, Tara McIntosh, and Fei Wu for many discussions about the paper.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives 2007. DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the International Semantic Web Conference*.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni 2007. Open Information Extraction from the Web. In *Proceedings of the International Joint Conference on Artificial Intelligence* .

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the International Conference on Management of Data*.

Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang 2008. WebTables: Exploring the Power of Tables on the Web. In *Proceedings of the VLDB Endowment*.

Luciano Del Corro and Rainer Gemulla. 2013. ClausIE: Clause-based Open Information Extraction. In *Proceedings of the International World Wide Web Conference*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of Language Resources and Evaluation*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying Relations for Open Information Extraction. In *Proceedings of Empirical Methods in Natural Language Processing*.

Rahul Gupta, Alon Halevy, Xuezhi Wang, Steven Whang, and Fei Wu. 2014. Biperpedia: An Ontology for Search Applications. In *Proceedings of the VLDB Endowment*.

Aria Haghighi and Dan Klein 2009. Simple Coreference Resolution with Rich Syntactic and Semantic Features. In *Proceedings of Empirical Methods in Natural Language Processing*.

Richard Johansson and Pierre Nugues 2008. The Effect of Syntactic Representation on Semantic Role Labeling. In *Proceedings of the International Conference on Computational Linguistics*.

Taesung Lee, Zhongyuan Wang, Haixun Wang, and Seung-won Hwang 2013. Attribute Extraction and Scoring: A Probabilistic Approach. In *Proceedings of the International Conference on Data Engineering* .

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open Language Learning for Information Extraction. In *Proceedings of Empirical Methods in Natural Language Processing.*

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant Supervision for Relation Extraction Without Labeled Data. In *Proceedings of the Association for Computational Linguistics*.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. Annotating Noun Argument Structure for NomBank. In *Proceedings of Language Resources and Evaluation*.

Nitin Madnani and Bonnie J. Dorr. 2010. Generating Phrasal and Sentential Paraphrases: A Survey of Data-Driven Methods. In *Computational Linguistics 36(3)*.

Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. 2011. Scalable Knowledge Harvesting with High Precision and High Recall. In *Proceedings of Web Search and Data Mining*.

Marius Pasca. 2014. Acquisition of Open-domain Classes via Intersective Semantics. In *Proceedings of the International World Wide Web Conference*.

Marius Pasca and Benjamin Van Durme. 2007. What You Seek Is What You Get: Extraction of Class Attributes from Query Logs. In *Proceedings of the International Joint Conference on Artificial Intelligence* .

Uma Sawant and Soumen Chakrabarti. 2013. Learning Joint Query Interpretation and Response Ranking. In *Proceedings of the International World Wide Web Conference*.

Amit Singhal. 2012 *Introducing the Knowledge Graph: things, not strings* `http://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html`

Fei Wu and Daniel S. Weld. 2010. Open Information Extraction Using Wikipedia. In *Proceedings of the the Association for Computational Linguistics*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural Language Questions for the Web of Data. In *Proceedings of Empirical Methods in Natural Language Processing*.

Hierarchical Discriminative Classification for Text-Based Geolocation

Benjamin Wing[†] **Jason Baldridge**[†]
[†]Department of Linguistics, University of Texas at Austin
ben@benwing.com, jbaldrid@utexas.edu

Abstract

Text-based document geolocation is commonly rooted in language-based information retrieval techniques over geodesic grids. These methods ignore the natural hierarchy of cells in such grids and fall afoul of independence assumptions. We demonstrate the effectiveness of using logistic regression models on a hierarchy of nodes in the grid, which improves upon the state of the art accuracy by several percent and reduces mean error distances by hundreds of kilometers on data from Twitter, Wikipedia, and Flickr. We also show that logistic regression performs feature selection effectively, assigning high weights to geocentric terms.

1 Introduction

Document geolocation is the identification of the location—a specific latitude and longitude—that forms the primary focus of a given document. This assumes that a document can be adequately associated with a *single* location, which is only valid for certain documents, generally of fairly small size. Nonetheless, there are many natural situations in which such collections arise. For example, a great number of articles in Wikipedia have been manually geotagged; this allows those articles to appear in their geographic locations in geobrowsers like Google Earth. Images in social networks such as Flickr may be geotagged by a camera and their textual tags can be treated as documents. Likewise, tweets in Twitter are often geotagged; in this case, it is possible to view either an individual tweet or the collection of tweets for a given user as a document, respectively identifying the location as the place from which the tweet was sent or the home location of the user.

Early work on document geolocation used heuristic algorithms, predicting locations based on toponyms in the text (named locations, determined with the aid of a gazetteer) (Ding et al., 2000; Smith and Crane, 2001). More recently, various researchers have used topic models for document geolocation (Ahmed et al., 2013; Hong et al., 2012; Eisenstein et al., 2011; Eisenstein et al., 2010) or other types of geographic document summarization (Mehrotra et al., 2013; Adams and Janowicz, 2012; Hao et al., 2010). A number of researchers have used metadata of various sorts for document or user geolocation, including document links and social network connections. This research has sometimes been applied to Wikipedia (Overell, 2009) or Facebook (Backstrom et al., 2010) but more commonly to Twitter, focusing variously on friends and followers (McGee et al., 2013; Sadilek et al., 2012), time zone (Mahmud et al., 2012), declared location (Hecht et al., 2011), or a combination of these (Schulz et al., 2013).

We tackle document geolocation using supervised methods based on the textual content of documents, ignoring their metadata. Metadata-based approaches can achieve great accuracy (e.g. Schulz et al. (2013) obtain 79% accuracy within 100 miles for a US-based Twitter corpus, compared with 49% using our methods on a comparable corpus), but are very specific to the particular corpus and the types of metadata it makes available. For Twitter, the metadata includes the user's declared location and time zone, information which greatly simplifies geolocation and which is unavailable for other types of corpora, such as Wikipedia. In many cases essentially no metadata is available at all, as in historical corpora in the digital humanities (Lunenfeld et al., 2012), such as those in the Perseus project (Crane, 2012). Text-based approaches can be applied to all types of corpora; metadata can be additionally incorporated when available (Han and Cook, 2013).

We introduce a hierarchical discriminative classification method for text-based geotagging. We

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 336–348,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

apply this to corpora in three languages (English, German and Portuguese). This method scales well to large training sets and greatly improves results across a wide variety of corpora, beating current state-of-the-art results by wide margins, including Twitter users (Han et al., 2014, henceforth Han14; Roller et al., 2012, henceforth Roller12); Wikipedia articles (Roller12; Wing and Baldridge, 2011, henceforth WB11); and Flickr images (O'Hare and Murdock, 2013, henceforth OM13). Importantly, this is the first method that improves upon straight uniform-grid Naive Bayes on all of these corpora, in contrast with *k*-d trees (Roller12) and the current state-of-the-art technique for Twitter users of geographically-salient feature selection (Han14).

We also show, contrary to Han14, that logistic regression when properly optimized is more accurate than state-of-the-art techniques, including feature selection, and fast enough to run on large corpora. Logistic regression itself very effectively picks out words with high geographic significance. In addition, because logistic regression does not assume feature independence, complex and overlapping features of various sorts can be employed.

2 Data

We work with six large datasets: two of geotagged tweets, three of Wikipedia articles, and one of Flickr photos. One of the two Twitter datasets is primarily localized to the United States, while the remaining datasets cover the whole world.

TWUS is a dataset of tweets compiled by Roller12. A document in this dataset is the concatenation of all tweets by a single user, as long as at least one of the user's tweets is geotagged with specific, GPS-assigned latitude/longitude coordinates. The earliest such tweet determines the user's location. Tweets outside of a bounding box covering the contiguous United States (including parts of Canada and Mexico) were discarded, as well as users that may be spammers or robots (based on the number of followers, followees and tweets). The resulting dataset contains 38M tweets from 450K users, of which 10,000 each are reserved for the development and test sets.

TWWORLD is a dataset of tweets compiled by Han et al. (2012). It was collected in a similar fashion to TWUS but differs in that it covers the entire Earth instead of primarily the United States, and consists only of geotagged tweets. Non-English tweets and those not near a city were removed, and non-alphabetic, overly short and overly infrequent words were filtered. The resulting dataset consists of 1.4M users, with 10,000 each reserved for the development and test sets.

ENWIKI13 is a dataset consisting of the 864K geotagged articles (out of 14M articles in all) in the November 4, 2013 English Wikipedia dump. It is comparable to the dataset used in WB11 and was processed using an analogous fashion. The articles were randomly split 80/10/10 into training, development and test sets.

DEWIKI14 is a similar dataset consisting of the 324K geotagged articles (out of 1.71M articles in all) in the July 5, 2014 German Wikipedia dump.

PTWIKI14 is a similar dataset consisting of the 131K geotagged articles (out of 817K articles in all) in the June 24, 2014 Portuguese Wikipedia dump.

COPHIR (Bolettieri et al., 2009) is a large dataset of images from the photo-sharing social network Flickr. It consists of 106M images, of which 8.7M are geotagged. Most images contain user-provided tags describing them. We follow algorithms described in OM13 in order to make direct comparison possible. This involves removing photos with empty tag sets and performing *bulk upload filtering*, retaining only one of a set of photos from a given user with identical tag sets. The resulting reduced set of 2.8M images is then divided 80/10/10 into training, development and test sets. The tag set of each photo is concatenated into a single piece of text (in the process losing user-supplied tag boundary information in the case of multi-word tags).

Our code and processed corpora are available for download.[1]

3 Supervised models for document geolocation

The dominant approach for text-based geolocation comes from language modeling approaches in information retrieval (Ponte and Croft, 1998; Manning et al., 2008). For this general strategy, the Earth is sub-divided into a grid, and then each training set document is associated with the cell that contains it. Some model (typically Naive Bayes) is then used to characterize each cell and

[1] https://github.com/utcompling/textgrounder/wiki/WingBaldridge_EMNLP2014

enable new documents to be assigned a latitude and longitude based on those characterizations. There are several options for constructing the grid and for modeling, which we review next.

3.1 Geodesic grids

The simplest grid is a uniform rectangular one with cells of equal-sized degrees, which was used by Serdyukov et al. (2009) for Flickr images and WB11 for Twitter and Wikipedia. This has two problems. Compared to a grid that takes document density into account, it over-represents rural areas at the expense of urban areas. Furthermore, the rectangles are not equal-area, but shrink in width away from the equator (although the shrinkage is mild until near the poles). Roller12 tackle the former issue by using an adaptive grid based on *k*-d trees, while Dias et al. (2012) handle the latter issue with an equal-area quaternary triangular mesh.

An additional issue with geodesic grids is that a single metro area may be divided between two or more cells. This can introduce a statistical bias known as the *modifiable areal unit problem* (Gehlke and Biehl, 1934; Openshaw, 1983). One way to mitigate this, implemented in Roller12's code but not investigated in their paper, is to divide a cell in a *k*-d tree in such a way as to produce the maximum margin between the dividing line and the nearest document on each side.

A more direct method is to use a city-based representation, either with a full set of sufficiently-sized cities covering the Earth and taken from a comprehensive gazetteer (Han14) or a limited, pre-specified set of cities (Kinsella et al., 2011; Sadilek et al., 2012). Han14 amalgamate cities into nearby larger cities within the same state (or equivalent); an even more direct method would use census-tract boundaries when available. Disadvantages of these methods are the dependency on time-specific population data, making them unsuitable for some corpora (e.g. 19th-century documents); the difficulty in adjusting grid resolution in a principled fashion; and the fact that not all documents are near a city (Han14 find that 8% of tweets are "rural" and cannot predicted by their model).

We construct rectangular grids, since they are very easy to implement and Dias et al. (2012)'s triangular mesh did not yield consistently better results over Wikipedia. We use both uniform grids and *k*-d tree grids with midpoint splitting.

3.2 Naive Bayes

A geodesic grid of sufficient granularity creates a large decision space, when each cell is viewed as a label to be predicted by some classifier. This situation naturally lends itself to simple, scalable language-modeling approaches. For this general strategy, each cell is characterized by a *pseudo-document* constructed from the training documents that it contains. A test document's location is then chosen based on the cell with the most similar language model according to standard measures such as Kullback-Leibler (KL) divergence (Zhai and Lafferty, 2001), which seeks the cell whose language model is closest to the test document's, or Naive Bayes (Lewis, 1998), which chooses the cell that assigns the highest probability to the test document.

Han14, Roller12 and WB11 follow this strategy, using KL divergence in preference to Naive Bayes. However, we find that Naive Bayes in conjunction with Dirichlet smoothing (Smucker and Allan, 2006) works at least as well when appropriately tuned. Dirichlet smoothing is a type of discounting model that interpolates between the unsmoothed (maximum-likelihood) document distribution $\tilde{\theta}_{d_i}$ of a document d_i and the unsmoothed distribution $\tilde{\theta}_D$ over all documents. A general interpolation model for the smoothed distribution θ_{d_i} has the following form:

$$P(w|\theta_{d_i}) = (1 - \lambda_{d_i})P(w|\tilde{\theta}_{d_i}) + \lambda_{d_i}P(w|\tilde{\theta}_D) \quad (1)$$

where the discount factor λ_{d_i} indicates how much probability mass to reserve for unseen words. For Dirichlet smoothing, λ_{d_i} is set as:

$$\lambda_{d_i} = 1 - \frac{|d_i|}{|d_i| + m} \quad (2)$$

where $|d_i|$ is the size of the document and m is a tunable parameter. This has the effect of relying more on d_i's distribution and less on the global distribution for larger documents that provide more evidence than shorter ones. Naive Bayes models are estimated easily, which allows them to handle fine-scale grid resolutions with potentially thousands or even hundreds of thousands of non-empty cells to choose among.

Figure 1 shows a choropleth map of the behavior of Naive Bayes, plotting the rank of cells for

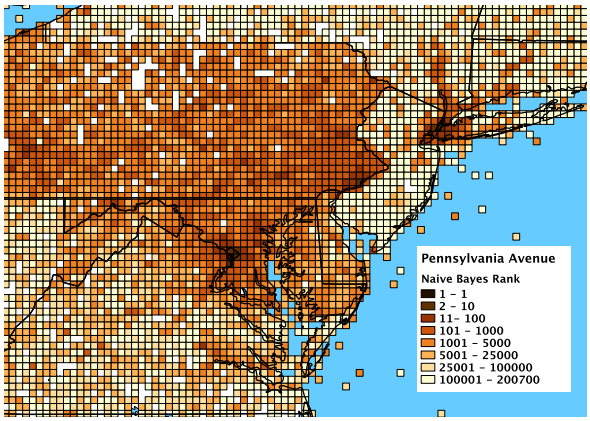


Figure 1: Relative Naive Bayes rank of cells for ENWIKI13 test document *Pennsylvania Avenue (Washington, DC)*, surrounding the true location.

the test document *Pennsylvania Avenue (Washington, DC)* in ENWIKI13, for a uniform 0.1° grid. The top-ranked cell is the correct one.

3.3 Logistic regression

The use of discrete cells over the Earth's surface allows any classification strategy to be employed, including discriminative classifiers such as logistic regression. Logistic regression often produces produces better results than generative classifiers at the cost of more time-consuming training, which limits the size of the problems it may be applied to. Training is generally unable to scale to encompass several thousand or more distinct labels, as is the case with fine-scale grids of the sort we may employ. Nonetheless we find flat logistic regression to be effective on most of our large-scale corpora, and the hierarchical classification strategy discussed in §4 allows us to take advantage of logistic regression without incurring such a high training cost.

3.4 Feature selection

Naive Bayes assumes that features are independent, which penalizes models that must accommodate many features that are poor indicators and which can gang up on the good features. Large improvements have been obtained by reducing the set of words used as features to those that are geographically salient. Cheng et al. (2010; 2013) model word locality using a unimodal distribution taken from Backstrom et al. (2008) and train a classifier to identify geographically local words based on this distribution. This unfortunately requires a large hand-annotated corpus for training. Han14 systematically investigate various feature selection methods for finding geo-indicative words, such as information gain ratio (IGR) (Quinlan, 1993), Ripley's K statistic (O'Sullivan and Unwin, 2010) and geographic density (Chang et al., 2012), showing significant improvements on TWUS and TWWORLD (§2).

For comparison with Han14, we test against an additional baseline: Naive Bayes combined with feature selection done using IGR. Following Han14, we first eliminate words which occur less than 10 times, have non-alphabetic characters in them or are shorter than 3 characters. We then compute the IGR for the remaining words across all cells at a given cell size or bucket size, select the top $N\%$ for some *cutoff percentage* N (which we vary in increments of 2%), and then run Naive Bayes at the same cell size or bucket size.

4 Hierarchical classification

To overcome the limitations of discriminative classifiers in terms of the maximum number of cells they can handle, we introduce hierarchical classification (Silla Jr. and Freitas, 2011) for geolocation. Dias et al. (2012) use a simple two-level generative hierarchical approach using Naive Bayes, but to our knowledge no previous work implements a multi-level discriminative hierarchical model with beam search for geolocation.

To construct the hierarchy, we start with a root cell c_{root} that spans the entire Earth and from there build a tree of cells at different scales, from coarse to fine. A cell at a given level is subdivided to create smaller cells at the next level of resolution that altogether cover the same area as their parent.

We use the *local classifier per parent* approach to hierarchical classification (Silla Jr. and Freitas, 2011) in which an independent classifier is learned for every node of the hierarchy above the leaf nodes. The probability of any node in the hierarchy is the product of the probabilities of that node and all of its ancestors, up to the root. This is defined recursively as:

$$\begin{aligned} P(c_{root}) &= 1.0 \\ P(c_j) &= P(c_j|\uparrow c_j)P(\uparrow c_j) \end{aligned} \quad (3)$$

where $\uparrow c_j$ indicates c_j's parent in the hierarchy.

In addition to allowing one to use many classifiers that each have a manageable number of outcomes, the hierarchical approach naturally lends itself to beam search. Rather than computing the

probability of every leaf cell using equation 3, we use a stratified beam search: starting at the root cell, keep the b highest-probability cells at each level until reaching the leaf node level. With a tight beam—which we show to be very effective—this dramatically reduces the number of model evaluations that must be performed at test time.

Grid size parameters Two factors determine the size of the grids at each level. The first-level grid is constructed the same as for Naive Bayes or flat logistic regression and is controlled by its own parameter. In addition, the *subdivision factor* N determines how we subdivide each cell to get from one level to the next. Both factors must be optimized appropriately.

For the uniform grid, we subdivide each cell into NxN subcells. In practice, there may actually be fewer subcells, because some of the potential subcells may be empty (contain no documents).

For the *k*-d grid, if level 1 is created using a bucket size B (i.e. we recursively divide cells as long as their size exceeds B), then level 2 is created by continuing to recursively divide cells that exceed a smaller bucket size B/N. At this point, the subcells of a given level-1 cell are the leaf cells contained with the cell's geographic area. The construction of level 3 proceeds similarly using bucket size B/N^2, etc.

Note that the subdivision factor has a different meaning for uniform and *k*-d tree grids. Furthermore, because creating the subdividing cells for a given cell involves dividing by N^2 for the uniform grid but N for the *k*-d tree grid, greater subdivision factors are generally required for the *k*-d tree grid to achieve similar-scale resolution.

Figure 2 shows the behavior of hierarchical LR using *k*-d trees for the test document *Pennsylvania Avenue (Washington, DC)* in ENWIKI13. After ranking the first level, the beam zooms in on the top-ranked cells and constructs a finer *k*-d tree under each one (one such subtree is shown in the top-right map callout).

5 Experimental Setup

Configurations. We experiment with several methods for configuring the grid and selecting the best cell. For grids, we use either a **uniform** or ***k*-d** tree grid. For uniform grids, the main tunable parameter is grid size (in **degrees**), while for *k*-d trees it is bucket size (**BK**), i.e. the number of documents above which a node is divided in two.

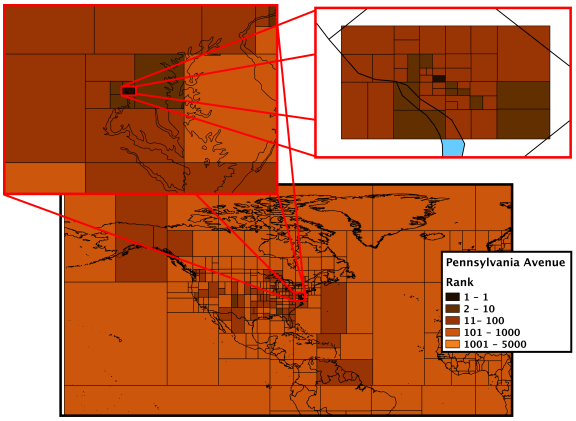


Figure 2: Relative hierarchical LR rank of cells for ENWIKI13 test document *Pennsylvania Avenue (Washington, DC)*, surrounding the true location. The first callout simply expands a portion of level 1, while the second callout shows a level 1 cell subdivided down to level 2.

For cell choice, the options are:

- **NB**: Naive Bayes baseline
- **IGR**: Naive Bayes using features selected by information gain ratio
- **FlatLR**: logistic regression model over all leaf nodes
- **HierLR**: product of logistic regression models at each node in a hierarchical grid (eq. 3)

For Dirichlet smoothing in conjunction with Naive Bayes, we set the Dirichlet parameter $m = 1,000,000$, which we found worked well in preliminary experiments. For hierarchical classification, there are additional parameters: subdivision factor (**SF**) and beam size (**BM**) (§4), and hierarchy depth (**D**) (§6.4). All of our test-set results use a depth of three levels.

Due to its speed and flexibility, we use Vowpal Wabbit (Agarwal et al., 2014) for logistic regression, estimating parameters with limited-memory BFGS (Nocedal, 1980; Byrd et al., 1995). Unless otherwise mentioned, we use 26-bit feature hashing (Weinberger et al., 2009) and 40 passes over the data (optimized based on early experiments on development data) and turn off the hold-out mechanism. For the subcell classifiers in hierarchical classification, which have fewer classes and much less data, we use 24-bit features and 12 passes.

Evaluation. To measure geolocation performance, we use three standard metrics based on *error distance*, i.e. the distance between the correct location and the predicted location. These metrics are **mean** and **median** error distance (Eisenstein et

al., 2010) and *accuracy at 161 km* (**acc@161**), i.e. within a 161-km radius, which was introduced by Cheng et al. (2010) as a proxy for accuracy within a metro area. All of these metrics are independent of cell size, unlike the measure of cell accuracy (fraction of cells correctly predicted) used in Serdyukov et al. (2009). Following Han14, we use acc@161 on development sets when choosing algorithmic parameter values such as cell and bucket sizes.

6 Results

6.1 Twitter

We show the effect of varying cell size in Table 1 and *k*-d tree bucket size in Figure 3. The number of non-empty cells is shown for each cell size and bucket size. For NB, this is the number of cells against which a comparison must be made for each test document; for FlatLR, this is the number of classes that must be distinguished. For HierLR, no figure is given because it varies from level to level and from classifier to classifier. For example, with a uniform grid and subdivision factor of 3, each level-2 subclassifier will have between 1 and 9 labels to choose among, depending on which cells are empty.

Method	Cell Size (Deg)	Cell Size (km)	#Class	Acc. @161	Mean (km)	Med. (km)
NB	0.17°		11,671	36.6	929.5	496.4
	0.50°		2,838	35.4	889.3	466.6
IGR, CU90%	1.5°		501	45.9	787.5	255.6
FlatLR	5°	556	59	35.4	727.8	248.7
	4°	445	99	44.4	718.8	227.9
	3°	334	159	47.3	721.3	186.2
	2.5°	278	208	47.5	743.9	198.9
	2°	223	316	46.9	737.7	209.9
	1.5°	167	501	46.6	762.6	226.9
	1°	111	975	43.0	810.0	303.7
HierLR, D2, SF2, BM5	4°	–	–	48.6	**695.2**	182.2
HierLR, D2, SF2, BM2	3°	–	–	**49.0**	725.1	174.6
HierLR, D3, SF2, BM2	3°	–	–	**49.0**	718.9	**173.8**
HierLR, D2, SF2, BM5	2.5°	–	–	48.2	740.9	187.7

Table 1: Dev set performance for TwUS, with uniform grids. HierLR and IGR parameters optimized using acc@161. Best metric numbers for a given method are underlined, except that overall best numbers are in bold.

FlatLR does much better than NB and IGR, and HierLR is still better. This is despite logistic regression needing to operate at a much lower resolution.[2] Interestingly, uniform-grid 2-level HierLR does better at 4° with a subdivision factor of 2 than the equivalent FlatLR run at 2°.

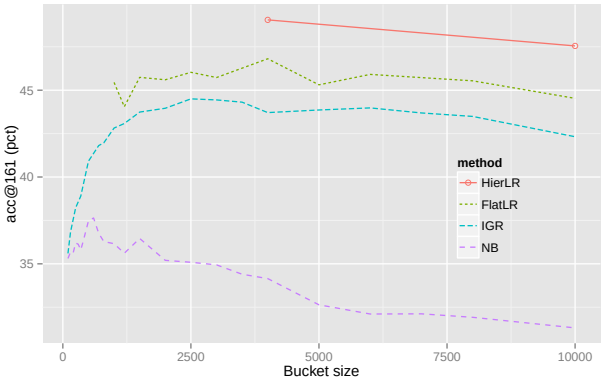


Figure 3: Dev set performance for TwUS, with *k*-d tree grids.

Table 2 shows the test set results for the various methods and metrics described in §5, on both TwUS and TwWorld.[3] HierLR is the best across all metrics; the best acc@161km and median error is obtained with a uniform grid, while HierLR with *k*-d trees obtains the best mean error.

Compared with vanilla NB, our implementation of NB using IGR feature selection obtains large gains for TwUS and moderate gains for TwWorld, showing that IGR can be an effective geolocation method for Twitter. This agrees in general with Han14's findings. We can only compare our figures directly with Han14 for *k*-d trees—in this case they use a version of the same software we use and report figures within 1% of ours for TwUS. Their remaining results are computed using a city-based grid and an NB implementation with add-one smoothing, and are significantly worse than our uniform-grid NB and IGR figures using Dirichlet smoothing, which is known to significantly outperform add-one smoothing (Smucker and Allan, 2006). For example, for NB they report 30.8% acc@161 for TwUS and 20.0% for TwWorld, compared with our 36.2% and 30.2% respectively. We suspect an additional reason for the discrepancy is due to the limitations of their city-based grid, which has no tunable parameter to optimize the grid size and requires that test instances not near a city be reported as incorrect.

Our NB figures also beat the KL divergence figures reported in Roller12 for TwUS (which they term UTGeo2011), perhaps again due to the dif-

[2] The limiting factor for resolution for us was the 24-hour per job limit on our computing cluster.

[3] Note that for TwWorld, it was necessary to modify the parameters normally passed to Vowpal Wabbit, moving up to 27-bit features and 96 passes, and 24-bit features with 24 passes in sublevels of HierLR.

Corpus	TwUS				TwWorld			
Method	Parameters	A@161	Mean	Med.	Parameters	A@161	Mean	Med.
NB Uniform	0.17°	36.2	913.8	476.3	1°	30.2	1690.0	537.2
NB *k*-d	BK1500	36.2	861.4	444.2	BK500	28.7	1735.0	566.2
IGR Uniform	1.5°, CU90%	46.1	770.3	233.9	1°, CU90%	31.0	2204.8	574.7
IGR *k*-d	BK2500, CU90%	44.6	792.0	268.6	BK250, CU92%	29.4	2369.6	655.0
FlatLR Uniform	2.5°	47.2	727.3	195.4	3.7°	32.1	1736.3	500.0
FlatLR *k*-d	BK4000	47.4	692.2	197.0	BK12000	27.8	1939.5	651.6
HierLR Uniform	3°, SF2, BM2	**49.2**	703.6	**170.5**	5°, SF2, BM1	**32.7**	1714.6	**490.0**
HierLR *k*-d	BK4000, SF3, BM1	48.0	**686.6**	191.4	BK60000, SF5, BM1	31.3	**1669.6**	509.1

Table 2: Performance on the test sets of TwUS and TwWorld for different methods and metrics.

ference in smoothing methods.

6.2 Wikipedia

Table 3 shows results on the test set of EnWiki13 for various methods. Table 5 shows the corresponding results for DeWiki14 and PtWiki14. In all cases, the best parameters for each method were determined using acc@161 on the development set, as above.

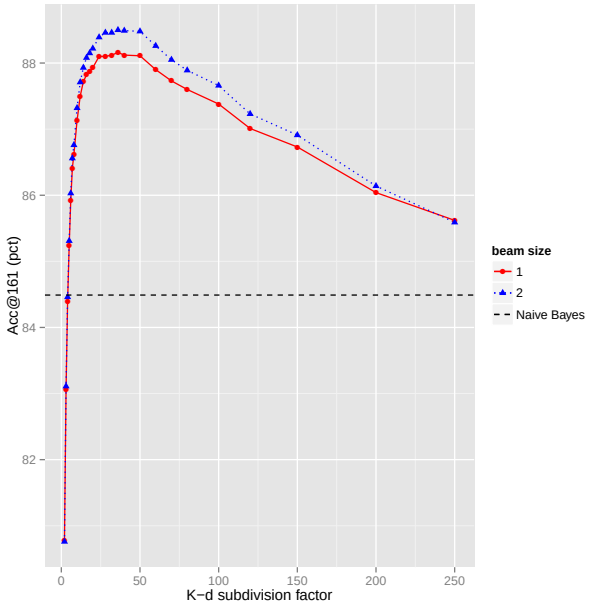


Figure 4: Plot of subdivision factor vs. acc@161 for the EnWiki13 dev set with 2-level *k*-d tree HierLR, bucket size 1500. Beam sizes above 2 yield little improvement.

HierLR is clearly the stand-out winner among all methods and metrics, and particularly so for the *k*-d tree grid. This is achieved through a high subdivision factor, especially in a 2-level hierarchy, where a factor of 36 is best, as shown in Figure 4 for EnWiki13. (For a 3-level hierarchy, the best subdivision factor is 12.)

Unlike for TwUS, FlatLR simply cannot compete with NB in the larger Wikipedias (EnWiki13 and DeWiki14). EnWiki13 especially has dense coverage across the entire world, whereas TwUS only covers the United States and parts of Canada and Mexico. Thus, there are a much larger number of non-empty cells at a given resolution and much coarser resolution required, especially with the uniform grid. For example, at 7.5° there are 933 non-empty cells, comparable to 1° for TwUS. Table 4 shows the number of classes and runtime for FlatLR and HierLR at different parameter values. The hierarchical classification approach is clearly essential for allowing us to scale the discriminative approach for a large, dense dataset across the whole world.

Method	Param	#Class	A@161	Med.	Runtime
FlatLR Uniform	10°	648	19.2	314.1	11h
	8.5°	784	26.5	248.5	16h
	7.5°	933	30.1	232.0	19h
FlatLR *k*-d	BK5000	257	57.1	133.5	5h
	BK2500	501	67.5	94.9	9h
	BK1500	825	74.7	69.9	16h
HierLR Uniform	7.5°,SF2,BM1	—	85.2	67.8	23h
	7.5°,SF3,BM5	—	86.1	34.2	27h
HierLR *k*-d	BK1500,SF5,BM1	—	88.2	19.6	23h
	BK5000,SF10,BM5	—	88.4	18.3	14h
	BK1500,SF12,BM2	—	88.8	15.3	33h

Table 4: Performance/runtime for FlatLR and 3-level HierLR on the EnWiki13 dev set, with varying parameters.

Moving from larger to smaller Wikipedias, FlatLR becomes more competitive. In particular, FlatLR outperforms NB and is close to HierLR for PtWiki14, the smallest of the three (and significantly smaller than TwUS). In this case, the relatively small size of the dataset and its greater geographic specificity (many articles are located in Brazil or Portugal) allows for a fine enough resolution to make FlatLR perform well—comparable to or even finer than NB.

In all of the Wikipedias, NB *k*-d outperforms

Corpus	EnWiki13				CoPhIR			
Method	Parameters	A@161	Mean	Med.	Parameters	A@161	Mean	Med.
NB Uniform	1.5°	84.0	326.8	56.3	1.5°	65.0	1553.5	47.9
NB *k*-d	BK100	84.5	362.3	21.1	BK3500	58.5	1726.9	70.0
IGR Uniform	1.5°, CU96%	81.4	401.9	58.2	1.5°, CU92%	60.8	1683.4	56.7
IGR *k*-d	BK250, CU98%	80.6	423.9	34.3	BK1500, CU62%	54.7	2908.8	83.5
FlatLR Uniform	7.5°	25.5	1347.8	259.4	2.0°	60.6	1942.3	73.7
FlatLR *k*-d	BK1500	74.8	253.2	70.0	BK3000	57.7	1961.4	72.5
HierLR Uniform	7.5°, SF3, BM5	86.2	228.3	34.0	7°, SF4, BM5	65.3	1590.2	**16.7**
HierLR *k*-d	BK1500, SF12, BM2	**88.9**	**168.7**	**15.3**	BK100000, SF15, BM5	**66.0**	**1453.3**	17.9

Table 3: Performance on the test sets of EnWiki13 and CoPhIR for different methods and metrics.

NB uniform, and HierLR outperforms both, but by greatly varying amounts, with only a 1% difference for DeWiki14 but 12% for PtWiki14. It's unclear what causes these variations, although it's worth noting that Roller12's NB *k*-d figures on an older English Wikipedia corpus were are noticeably higher than our figures: They report 90.3% acc@161, compared with our 84.5%. We verified that this is due to corpus differences: we obtain their performance when we run on their Wikipedia corpus. This suggests that the various differences may be due to vagaries of the individual corpora, e.g. the presence of differing numbers of geotagged stub articles, which are very short and thus hard to geolocate.

As for IGR, though it is competitive for Twitter, it performs badly here—in fact, it is even worse than plain Naive Bayes for all three Wikipedias (likewise for CoPhIR, in the next section).

6.3 CoPhIR

Table 3 shows results on the test set of CoPhIR for various methods, similarly to the EnWiki13 results. HierLR is again the clear winner. Unlike for EnWiki13, FlatLR is able to do fairly well. IGR performs poorly, especially when combined with *k*-d.

In general, as can be seen, for CoPhIR the median figures are very low but the mean figures very high, meaning there are many images that can be very accurately placed while the remainder are very difficult to place. (The former images likely have the location mentioned in the tags, while the latter do not.)

For CoPhIR, and also TwWorld, HierLR performs best when the root level is significantly coarser than the cell or bucket size that is best for FlatLR. The best setting for the root level appears to be correlated with cell accuracy, which in general increases with larger cell sizes. The intuition here is that HierLR works by drilling down from a single top-level child of the root cell. Thus, the higher the cell accuracy, the greater the fraction of test instances that can be improved in this fashion, and in general the better the ultimate values of the main metrics. (The above discussion isn't strictly true for beam sizes above 1, but these tend to produce marginal improvements, with little if any gain from going above a beam size of 5.) The large size of a coarse root-child cell, and correspondingly poor results for acc@161, can be offset by a high subdivision factor, which does not materially slow down the training process.

Our NB results are not directly comparable with OM13's results on CoPhIR because they use various cell-based accuracy metrics while we use cell-size-independent metrics. The closest to our acc@161 metric is their Ac1 metric, which at a cell size of 100 km corresponds to a 300km-per-side square at the equator, roughly comparable to our 161-km-radius circle. They report Ac1 figures of 57.7% for term frequency and 65.3% for user frequency, which counts the number of distinct users in a cell using a given term and is intended to offset bias resulting from users who upload a large batch of similar photos at a given location. Our term frequency figure of 65.0% significantly beats theirs, but we found that user frequency actually degraded our dev set results by 5%. The reason for this discrepancy is unclear.

6.4 Parameterization variations

Optimizing for median. Note that better values for the other metrics, especially median, can be achieved by specifically optimizing for these metrics. In general, the best parameters for median are finer-scale than those for acc@161: smaller grid sizes and bucket sizes, and greater subdivision factors. This is especially revealing in EnWiki13 and CoPhIR. For example, on the EnWiki13

Corpus	DEWIKI14				PTWIKI14			
Method	Parameters	A@161	Mean	Med.	Parameters	A@161	Mean	Med.
NB Uniform	1°	88.4	257.9	35.0	1°	76.6	470.0	48.3
NB *k*-d	BK25	89.3	192.0	**7.6**	BK100	77.1	325.0	45.9
IGR Uniform	2°, CU82%	87.1	312.9	68.2	2°, CU54%	71.3	594.6	89.4
IGR *k*-d	BK50, CU100%	86.0	226.8	10.9	BK100, CU100%	71.3	491.9	57.7
FlatLR Uniform	5°	55.1	340.4	150.1	2°	88.9	320.0	70.8
FlatLR *k*-d	BK350	82.0	193.2	24.5	BK25	86.8	320.8	30.0
HierLR Uniform	7°, SF3, BM5	88.5	184.8	30.0	7°, SF2, BM5	88.6	223.5	64.7
HierLR *k*-d	BK3500, SF25, BM5	**90.2**	**122.5**	8.6	BK250, SF12, BM2	**89.5**	**186.6**	**27.2**

Table 5: Performance on the test sets of DEWIKI14 and PTWIKI14 for different methods and metrics.

dev set, the "best" uniform NB parameter of 1.5°, as optimized on acc@161, yields a median error of 56.1 km, but an error of just 16.7 km can be achieved with the parameter setting 0.25° (which, however, drops acc@161 from 83.8% to 78.3% in the process). Similarly, for the COPHIR dev set, the optimized uniform 2-level HierLR median error of 46.6 km can be reduced to just 8.1 km by dropping from 7° to 3.5° and bumping up the subdivision factor from 4 to 35—again, causing a drop in acc@161 from 68.6% to 65.5%.

Hierarchy depth. We use a 3-level hierarchy throughout for the test set results. Evaluation on development data showed that 2-level hierarchies perform comparably for several data sets, but are less effective overall. We did not find improvements from using more than three levels. When using a simple local classifier per parent approach as we do, which chains together spines of related but independently trained classifiers when assigning a probability to a leaf cell, most of the benefit presumably comes from simply enabling logistic regression to be used with fine-grained leaf cells, overcoming the limitations of FlatLR. Further benefits of the hierarchical approach might be achieved with the data-biasing and bottom-up error propagation techniques of Bennett and Nguyen (2009) or the hierarchical Bayesian approach of Gopal et al. (2012), which is able to handle large-scale corpora and thousands of classes.

6.5 Feature Selection

The main focus of Han14 is identifying geographically salient words through feature selection. Logistic regression performs feature selection naturally by assigning higher weights to features that better discriminate among the target classes.

Table 6 shows the top 20 features ranked by feature weight for a number of different cells, labeled by the largest city in the cell. The features were produced using a uniform 5° grid, trained using 27-bit features and 40 passes over TWUS. The high number of bits per feature were chosen to ensure as few collisions as possible of different features (as it would be impossible to distinguish two words that were hashed together).

Most words are clearly region specific, consisting of cities, states and abbreviations, sports teams (*broncos*, *texans*, *niners*, *saints*), well-known streets (*bourbon*, *folsom*), characteristic features (*desert*, *bayou*, *earthquake*, *temple*), local brands (*whataburger*, *soopers*, *heb*), local foods (*gumbo*, *poutine*), and dialect terms (*hella*, *buku*).

Top-IGR words		Bottom-IGR words	
lockerby	presswiches	plan	times
killdeer	haubrich	party	end
fordville	yabbo	men	twitter
azilda	presswich	happy	full
ahauah	pozuelo	show	part
hutmacher	akeley	top	forget
cere	chewelah	extra	close
miramichi	computacionales	late	dead
alamosa	bevilacqua	facebook	cool
multiservicios	presswiche	friday	enjoy
ghibran	curtisinn	black	true
briaroaks	guymon	dream	found
joekins	dakotamart	hey	drink
numerica	missoula	face	pay
bemidji	mimbres	finally	meet
amn	shingobee	easy	lost
roug	gottsch	time	find
pbtisd	uprr	live	touch
marcenado	hesperus	wow	birthday
banerjee	racingmason	yesterday	ago

Table 7: Top and bottom 40 features selected using IGR for TWUS with a uniform 1.5° grid.

As a comparison, Table 7 shows the top and bottom 40 features selected using IGR on the same corpus. Unlike for logistic regression, the top IGR features are mostly obscure words, only some of

Salt Lake	San Francisco	New Orleans	Phoenix	Denver	Houston	Montreal	Seattle	Tulsa	Los Angeles
utah	sacramento	orleans	tucson	denver	houston	montreal	seattle	tulsa	knotts
slc	hella	jtfo	az	colorado	antonio	mtl	portland	okc	sd
salt	sac	prelaw	phoenix	broncos	texans	quebec	tacoma	oklahoma	pasadena
byu	niners	saints	arizona	aurora	sa	magrib	wa	wichita	diego
provo	berkeley	louisiana	asu	amarillo	corpus	rue	vancouver	ou	ucla
ut	safeway	bourbon	tempe	soopers	whataburger	habs	bellevue	kansas	disneyland
utes	oakland	kmsl	scottsdale	colfax	heb	canadian	oregon	ku	irvine
idaho	earthquake	uptown	phx	springs	otc	ouest	seahawks	lawrence	socal
orem	sf	joked	chandler	centennial	utsa	mcgill	pdx	shaki	tijuana
sandy	modesto	wya	fry	pueblo	mcallen	coin	uw	ks	riverside
rio	exploit	canal	glendale	larimer	westheimer	gmusic	puyallup	edmond	pomona
ogden	stockton	metairie	desert	meadows	pearland	laval	safeway	osu	turnt
lds	hayward	westbank	harkins	parker	jammin	poutine	huskies	stillwater	angeles
temple	cal	bayou	camelback	blake	mayne	boul	everett	topeka	usc
murray	jose	houma	mesa	cherry	katy	est	seatac	sooners	chargers
menudito	swaaaaggg	lawd	gilbert	siiiiim	jamming	je	ducks	straighht	oc
mormon	folsom	gtf	pima	coors	tsu	sherbrooke	victoria	kc	compton
gateway	roseville	magazine	dbacks	englewood	marcos	pas	beaverton	manhattan	meadowview
megaplex	juiced	gumbo	mcdowell	pikes	laredo	fkn	hella	boomer	rancho
lake	vallejo	buku	devils	rockies	texas	centre	sounders	sooner	ventura

Table 6: Top 20 features selected for various regions using logistic regression on TwUS with a uniform 5° grid.

which have geographic significance, while the bottom words are quite common. To some extent this is a feature of IGR, since it divides by the binary entropy of each word, which is directly related to its frequency. However, it shows why cutoffs around 90% of the original feature set are necessary to achieve good performance on the Twitter corpora. (IGR does not perform well on Wikipedia or CoPhIR, as shown above.)

7 Conclusion

This paper demonstrates that major performance improvements to geolocation based only on text can be obtained by using a hierarchy of logistic regression classifiers. Logistic regression also allows for the use of complex, interdependent features, beyond the simple unigram models commonly employed. Our preliminary experiments did not show noticeable improvements from bigram or character-based features, but it is possible that higher-level features such as morphological, part-of-speech or syntactic features could yield further performance gains. And, of course, these improved text-based models may help decrease error even further when metadata (e.g. time zone and declared location) are available.

An interesting extension of this work is to rely upon the natural clustering of related documents. Joint modeling of geographic topics and locations has been attempted (see §1), but has generally been applied to much smaller corpora than those considered here. Skiles (2012) found significant improvements by clustering the training documents of large-scale corpora using K-means, training separate models from each cluster, and estimating a test document's location with the cluster model returning the best overall similarity (e.g. through KL divergence). Bergsma et al. (2013) likewise cluster tweets using K-means but predict location only at the country level. Such methods could be combined with hierarchical classification to yield further gains.

Acknowledgments

We would like to thank Grant Delozier for assistance in generating choropleth graphs, and the three anonymous reviewers for their feedback. This research was supported by a grant from the Morris Memorial Trust Fund of the New York Community Trust.

References

Benjamin Adams and Krzysztof Janowicz. 2012. On the geo-indicativeness of non-georeferenced text. In John G. Breslin, Nicole B. Ellison, James G. Shanahan, and Zeynep Tufekci, editors, *ICWSM'12: Proceedings of the 6th International AAAI Conference on Weblogs and Social Media*. The AAAI Press.

Alekh Agarwal, Oliveier Chapelle, Miroslav Dudík, and John Langford. 2014. A reliable effective terascale linear learning system. *Journal of Machine Learning Research*, 15:1111–1133.

Amr Ahmed, Liangjie Hong, and Alexander J. Smola. 2013. Hierarchical geographical modeling of user

locations from social media posts. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, pages 25–36, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Lars Backstrom, Jon Kleinberg, Ravi Kumar, and Jasmine Novak. 2008. Spatial variation in search engine queries. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 357–366, New York, NY, USA. ACM.

Lars Backstrom, Eric Sun, and Cameron Marlow. 2010. Find me if you can: improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 61–70, New York, NY, USA. ACM.

Paul N. Bennett and Nam Nguyen. 2009. Refined experts: improving classification in large taxonomies. In James Allan, Javed A. Aslam, Mark Sanderson, ChengXiang Zhai, and Justin Zobel, editors, *SIGIR*, pages 11–18. ACM.

Shane Bergsma, Mark Dredze, Benjamin Van Durme, Theresa Wilson, and David Yarowsky. 2013. Broadly improving user classification via communication-based name and location clustering on twitter. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1010–1019, Atlanta, Georgia, June. Association for Computational Linguistics.

Paolo Bolettieri, Andrea Esuli, Fabrizio Falchi, Claudio Lucchese, Raffaele Perego, Tommaso Piccioli, and Fausto Rabitti. 2009. Cophir: a test collection for content-based image retrieval. *CoRR*, abs/0905.4627.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

Hau-Wen Chang, Dongwon Lee, Mohammed Eltaher, and Jeongkyu Lee. 2012. @phillies tweeting from philly? predicting twitter user locations with spatial word usage. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 111–118. IEEE Computer Society.

Zhiyuan Cheng, James Caverlee, and Kyumin Lee. 2010. You are where you tweet: A content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 759–768.

Zhiyuan Cheng, James Caverlee, and Kyumin Lee. 2013. A content-driven framework for geolocating microblog users. *ACM Trans. Intell. Syst. Technol.*, 4(1):2:1–2:27, February.

Gregory Crane, 2012. *The Perseus Project*, pages 644–653. SAGE Publications, Inc.

Duarte Dias, Ivo Anastácio, and Bruno Martins. 2012. A Language Modeling Approach for Georeferencing Textual Documents. In *Proceedings of the Spanish Conference in Information Retrieval*.

Junyan Ding, Luis Gravano, and Narayanan Shivakumar. 2000. Computing geographical scopes of web resources. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 545–556, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1277–1287, Cambridge, MA, October. Association for Computational Linguistics.

Jacon Eisenstein, Ahmed Ahmed, and Eric P. Xing. 2011. Sparse additive generative models of text. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1041–1048.

Charles E. Gehlke and Katherine Biehl. 1934. Certain effects of grouping upon the size of the correlation coefficient in census tract material. *Journal of the American Statistical Association*, 29(185):169–170.

Siddharth Gopal, Yiming Yang, Bing Bai, and Alexandru Niculescu-Mizil. 2012. Bayesian models for large-scale hierarchical classification. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Lon Bottou, and Kilian Q. Weinberger, editors, *NIPS*, pages 2420–2428.

Bo Han and Paul Cook. 2013. A stacking-based approach to twitter user geolocation prediction. In *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013): System Demonstrations*, pages 7–12.

Bo Han, Paul Cook, and Tim Baldwin. 2012. Geolocation prediction in social media data by finding location indicative words. In *International Conference on Computational Linguistics (COLING)*, page 17, Mumbai, India, December.

Bo Han, Paul Cook, and Tim Baldwin. 2014. Text-based twitter user geolocation prediction. *Journal of Artificial Intelligence Research*, 49(1):451–500.

Qiang Hao, Rui Cai, Changhu Wang, Rong Xiao, Jiang-Ming Yang, Yanwei Pang, and Lei Zhang. 2010. Equip tourists with knowledge mined from travelogues. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 401–410, New York, NY, USA. ACM.

Brent Hecht, Lichan Hong, Bongwon Suh, and Ed H. Chi. 2011. Tweets from justin bieber's heart: The dynamics of the location field in user profiles. In

Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '11, pages 237–246, New York, NY, USA. ACM.

Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J. Smola, and Kostas Tsioutsiouliklis. 2012. Discovering geographical topics in the twitter stream. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 769–778, New York, NY, USA. ACM.

Sheila Kinsella, Vanessa Murdock, and Neil O'Hare. 2011. "I'm eating a sandwich in Glasgow": Modeling locations with tweets. In *Proceedings of the 3rd International Workshop on Search and Mining User-generated Contents*, pages 61–68.

David D. Lewis. 1998. Naive (bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 4–15, London, UK, UK. Springer-Verlag.

Peter Lunenfeld, Anne Burdick, Johanna Drucker, Todd Presner, and Jeffrey Schnapp. 2012. *Digital humanities*. MIT Press, Cambridge, MA.

Jalal Mahmud, Jeffrey Nichols, and Clemens Drews. 2012. Where is this tweet from? inferring home locations of twitter users. In John G. Breslin, Nicole B. Ellison, James G. Shanahan, and Zeynep Tufekci, editors, *ICWSM'12: Proceedings of the 6th International AAAI Conference on Weblogs and Social Media*. The AAAI Press.

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

Jeffrey McGee, James Caverlee, and Zhiyuan Cheng. 2013. Location prediction in social media based on tie strength. In *Proceedings of the 22nd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '13, pages 459–468, New York, NY, USA. ACM.

Rishabh Mehrotra, Scott Sanner, Wray Buntine, and Lexing Xie. 2013. Improving lda topic models for microblogs via tweet pooling and automatic labeling. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 889–892, New York, NY, USA. ACM.

Jorge Nocedal. 1980. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation*, 35(151):773–782.

Neil O'Hare and Vanessa Murdock. 2013. Modeling locations with social media. *Information Retrieval*, 16(1):30–62.

Stan Openshaw. 1983. *The modifiable areal unit problem*. Geo Books.

David O'Sullivan and David J. Unwin, 2010. *Point Pattern Analysis*, pages 121–155. John Wiley & Sons, Inc.

Simon Overell. 2009. *Geographic Information Retrieval: Classification, Disambiguation and Modelling*. Ph.D. thesis, Imperial College London.

Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 275–281, New York, NY, USA. ACM.

J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Stephen Roller, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge. 2012. Supervised text-based geolocation using language models on an adaptive grid. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1500–1510, Stroudsburg, PA, USA. Association for Computational Linguistics.

Adam Sadilek, Henry Kautz, and Jeffrey P. Bigham. 2012. Finding your friends and following them to where you are. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*, pages 723–732.

Axel Schulz, Aristotelis Hadjakos, Heiko Paulheim, Johannes Nachtwey, and Max Mühlhäuser. 2013. A multi-indicator approach for geolocalization of tweets. In Emre Kiciman, Nicole B. Ellison, Bernie Hogan, Paul Resnick, and Ian Soboroff, editors, *ICWSM'13: Proceedings of the 7th International AAAI Conference on Weblogs and Social Media*. The AAAI Press.

Pavel Serdyukov, Vanessa Murdock, and Roelof van Zwol. 2009. Placing flickr photos on a map. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 484–491, New York, NY, USA. ACM.

Carlos N. Silla Jr. and Alex A. Freitas. 2011. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):182–196, January.

Erik David Skiles. 2012. Document geolocation using language models built from lexical and geographic similarity. Master's thesis, University of Texas at Austin.

David A. Smith and Gregory Crane. 2001. Disambiguating geographic names in a historical digital library. In *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries*, ECDL '01, pages 127–136, London, UK. Springer-Verlag.

Mark D. Smucker and James Allan. 2006. An investigation of Dirichlet prior smoothing's performance advantage. Technical report, University of Massachusetts, Amherst.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1113–1120, New York, NY, USA. ACM.

Benjamin Wing and Jason Baldridge. 2011. Simple supervised document geolocation with geodesic grids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 955–964, Portland, Oregon, USA, June. Association for Computational Linguistics.

Chengxiang Zhai and John Lafferty. 2001. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 403–410, New York, NY, USA. ACM.

Probabilistic Models of Cross-Lingual Semantic Similarity in Context Based on Latent Cross-Lingual Concepts Induced from Comparable Data

Ivan Vulić and Marie-Francine Moens
Department of Computer Science
KU Leuven, Belgium
{ivan.vulic|marie-francine.moens}@cs.kuleuven.be

Abstract

We propose the first probabilistic approach to modeling cross-lingual semantic similarity (CLSS) in context which requires only comparable data. The approach relies on an idea of projecting words and sets of words into a shared latent semantic space spanned by language-pair independent latent semantic concepts (e.g., cross-lingual topics obtained by a multilingual topic model). These latent cross-lingual concepts are induced from a comparable corpus without any additional lexical resources. Word meaning is represented as a probability distribution over the latent concepts, and a change in meaning is represented as a change in the distribution over these latent concepts. We present new models that modulate the isolated out-of-context word representations with contextual knowledge. Results on the task of suggesting word translations in context for 3 language pairs reveal the utility of the proposed contextualized models of cross-lingual semantic similarity.

1 Introduction

Cross-lingual semantic similarity (CLSS) is a metric that measures to which extent words (or more generally, text units) describe similar semantic concepts and convey similar meanings across languages. Models of cross-lingual similarity are typically used to automatically induce bilingual lexicons and have found numerous applications in information retrieval (IR), statistical machine translation (SMT) and other natural language processing (NLP) tasks. Within the IR framework, the output of the CLSS models is a key resource in the models of dictionary-based cross-lingual information retrieval (Ballesteros and Croft, 1997; Lavrenko et al., 2002; Levow et al., 2005; Wang and Oard, 2006) or may be utilized in query expansion in cross-lingual IR models (Adriani and van Rijsbergen, 1999; Vulić et al., 2013). These CLSS models may also be utilized as an additional source of knowledge in SMT systems (Och and Ney, 2003; Wu et al., 2008). Additionally, the models are a crucial component in the cross-lingual tasks involving a sort of cross-lingual knowledge transfer, where the knowledge about utterances in one language may be transferred to another. The utility of the transfer or annotation projection by means of bilingual lexicons obtained from the CLSS models has already been proven in various tasks such as semantic role labeling (Padó and Lapata, 2009; van der Plas et al., 2011), parsing (Zhao et al., 2009; Durrett et al., 2012; Täckström et al., 2013b), POS tagging (Yarowsky and Ngai, 2001; Das and Petrov, 2011; Täckström et al., 2013a; Ganchev and Das, 2013), verb classification (Merlo et al., 2002), inducing selectional preferences (Peirsman and Padó, 2010), named entity recognition (Kim et al., 2012), named entity segmentation (Ganchev and Das, 2013), etc.

The models of cross-lingual semantic similarity from parallel corpora rely on word alignment models (Brown et al., 1993; Och and Ney, 2003), but due to a relative scarceness of parallel texts for many language pairs and domains, the models of cross-lingual similarity from comparable corpora have gained much attention recently.

All these models from parallel and comparable corpora provide *ranked lists of semantically similar words* in the target language *in isolation* or *invariably*, that is, they do not explicitly iden-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 349–362,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

tify and encode different senses of words. In practice, it means that, given the sentence *"The coach of his team was not satisfied with the game yesterday."*, these context-insensitive models of similarity are not able to detect that the Spanish word *entrenador* is more similar to the polysemous word *coach* in the context of this sentence than the Spanish word *autocar*, although *autocar* is listed as the most semantically similar word to *coach* globally/invariably without any observed context. In another example, while Spanish words *partido*, *encuentro*, *cerilla* or *correspondencia* are all highly similar to the ambiguous English word *match* when observed in isolation, given the Spanish sentence *"She was unable to find a match in her pocket to light up a cigarette."*, it is clear that the strength of semantic similarity should change in context as only *cerilla* exhibits a strong semantic similarity to *match* within this particular sentential context.

Following this intuition, in this paper we investigate models of cross-lingual semantic similarity in context. The context-sensitive models of similarity target to *re-rank* the lists of semantically similar words based on the co-occurring contexts of words. Unlike prior work (e.g., (Ng et al., 2003; Prior et al., 2011; Apidianaki, 2011)), *we explore these models in a particularly difficult and minimalist setting that builds only on co-occurrence counts and latent cross-lingual semantic concepts induced directly from comparable corpora, and which does not rely on any other resource (e.g., machine-readable dictionaries, parallel corpora, explicit ontology and category knowledge)*. In that respect, the work reported in this paper extends the current research on purely statistical data-driven distributional models of cross-lingual semantic similarity that are built upon the idea of latent cross-lingual concepts (Haghighi et al., 2008; Daumé III and Jagarlamudi, 2011; Vulić et al., 2011; Vulić and Moens, 2013) induced from non-parallel data. While all the previous models in this framework are context-insensitive models of semantic similarity, we demonstrate how to build context-aware models of semantic similarity within the same probabilistic framework which relies on the same shared set of latent concepts.

The main contributions of this paper are:

- We present a new probabilistic approach to modeling cross-lingual semantic similarity in context based on latent cross-lingual semantic concepts induced from non-parallel data.
- We show how to use the models of cross-lingual semantic similarity in the task of suggesting word translations in context.
- We provide results for three language pairs which demonstrate that contextualized models of similarity significantly outscore context-insensitive models.

2 Towards Cross-Lingual Semantic Similarity in Context

Latent Cross-Lingual Concepts. Latent cross-lingual concepts/senses may be interpreted as language-independent semantic concepts present in a multilingual corpus (e.g., document-aligned Wikipedia articles in English, Spanish and Dutch) that have their language-specific representations in different languages. For instance, having a multilingual collection in English, Spanish and Dutch, and then discovering a latent semantic concept on *Soccer*, that concept would be represented by words (actually probabilities over words $P(w|z_k)$, where w denotes a word, and z_k denotes k-th latent concept): {*player, goal, coach, ...*} in English, *balón (ball), futbolista (soccer player), equipo (team), ...*} in Spanish, and {*wedstrijd (match), elftal (soccer team), doelpunt (goal), ...*} in Dutch. Given a multilingual corpus $\mathcal{C}$, the goal is to learn and extract a set $\mathcal{Z}$ of K latent cross-lingual concepts $\{z_1, \ldots, z_K\}$ that optimally describe the observed data, that is, the multilingual corpus $\mathcal{C}$. Extracting cross-lingual concepts actually implies learning *per-document concept distributions* for each document in the corpus, and discovering language-specific representations of these concepts given by *per-concept word distributions* in each language.

$\mathcal{Z} = \{z_1, \ldots, z_K\}$ represents the set of K latent cross-lingual concepts present in the multilingual corpus. These K semantic concepts actually span a latent cross-lingual semantic space. Each word w, irrespective of its actual language, may be represented in that latent semantic space as a K-dimensional vector, where each vector component is a conditional concept score $P(z_k|w)$.

A number of models may be employed to induce the latent concepts. For instance, one could use cross-lingual Latent Semantic Indexing (Dumais et al., 1996), probabilistic Principal Component Analysis (Tipping and Bishop, 1999), or a probabilistic interpretation of non-negative matrix

factorization (Lee and Seung, 1999; Gaussier and Goutte, 2005; Ding et al., 2008) on concatenated documents in aligned document pairs. Other more recent models include matching canonical correlation analysis (Haghighi et al., 2008; Daumé III and Jagarlamudi, 2011) and multilingual probabilistic topic models (Ni et al., 2009; De Smet and Moens, 2009; Mimno et al., 2009; Boyd-Graber and Blei, 2009; Zhang et al., 2010; Fukumasu et al., 2012).

Due to its inherent language pair independent nature and state-of-the-art performance in the tasks such as bilingual lexicon extraction (Vulić et al., 2011) and cross-lingual information retrieval (Vulić et al., 2013), the description in this paper relies on the multilingual probabilistic topic modeling (MuPTM) framework. We draw a direct parallel between latent cross-lingual concepts and latent cross-lingual topics, and we present the framework from the MuPTM perspective, but the proposed framework is generic and allows the usage of all other models that are able to compute probability scores $P(z_k|w)$. These scores in MuPTM are induced from their output language-specific *per-topic word distributions*. The multilingual probabilistic topic models output probability scores $P(w_i^S|z_k)$ and $P(w_j^T|z_k)$ for each $w_i^S \in V^S$ and $w_j^T \in V^T$ and each $z_k \in \mathcal{Z}$, and it holds $\sum_{w_i^S \in V^S} P(w_i^S|z_k) = 1$ and $\sum_{w_j^T \in V^T} P(w_j^T|z_k) = 1$. The scores are then used to compute scores $P(z_k|w_i^S)$ and $P(z_k|w_j^T)$ in order to represent words from the two different languages in the same latent semantic space in a uniform way.

Context-Insensitive Models of Similarity. Without observing any context, the standard models of semantic word similarity that rely on the semantic space spanned by latent cross-lingual concepts in both monolingual (Dinu and Lapata, 2010a; Dinu and Lapata, 2010b) and multilingual settings (Vulić et al., 2011) typically proceed in the following manner. Latent language-independent concepts (e.g., cross-lingual topics or latent word senses) are estimated on a large corpus. The K-dimensional vector representation of the word $w_1^S \in V^S$ is:

$$vec(w_1^S) = [P(z_1|w_1^S), \ldots, P(z_K|w_1^S)] \quad (1)$$

Similarly, we are able to represent any target language word w_2^T in the same latent semantic space by a K-dimensional vector with scores $P(z_k|w_2^T)$. Each word regardless of its language is represented as a distribution over K latent concepts. The similarity between w_1^S and some word $w_2^T \in V^T$ is then computed as the similarity between their K-dimensional vector representations using some of the standard similarity measures (e.g., the Kullback-Leibler or the Jensen-Shannon divergence, the cosine measure). These methods use only global co-occurrence statistics from the training set and do not take into account any contextual information. They provide only *out-of-context word representations* and are therefore able to deliver only *context-insensitive models of similarity*.

Defining Context. Given an occurrence of a word w_1^S, we build its context set $Con(w_1^S) = \{cw_1^S, \ldots, cw_r^S\}$ that comprises r words from V^S that co-occur with w_1^S in a defined *contextual scope* or granularity. In this work we do not investigate the influence of the context scope (e.g., document-based, paragraph-based, window-based contexts). Following the recent work from Huang et al. (2012) in the monolingual setting, we limit the contextual scope to the *sentential context*. However, we emphasize that the proposed models are designed to be fully functional regardless of the actual chosen context granularity. e.g., when operating in the sentential context, $Con(w_1^S)$ consists of words occurring in the same sentence with the particular instance of w_1^S. Following Mitchell and Lapata (2008), for the sake of simplicity, we impose the *bag-of-words* assumption, and do not take into account the order of words in the context set as well as context words' dependency relations to w_1^S. Investigating different context types (e.g., dependency-based) is a subject of future work.

By using all words occurring with w_1^S in a context set (e.g., a sentence) to build the set $Con(w_1^S)$, we do not make any distinction between "informative and "uninformative" context words. However, some context words bear more contextual information about the observed word w_1^S and are stronger indicators of the correct word meaning in that particular context. For instance, in the sentence *"The coach of his team was not satisfied with the game yesterday"*, words *game* and *team* are strong clues that *coach* should be translated as *entrenador* while the context word *yesterday* does not bring any extra contextual information that could resolve the ambiguity.

Therefore, in the final context set $Con(w_1^S)$ it is useful to retain only the context words that re-

ally bring extra semantic information. We achieve that by exploiting the same latent semantic space to provide the similarity score between the observed word w_1^S and each word $cw_i^S, i = 1, \ldots, r$ from its context set $Con(w_1^S)$. Each word cw_i^S may be represented by its vector $vec(cw_i^S)$ (see eq. (1)) in the same latent semantic space, and there we can compute the similarity between its vector and $vec(w_1^S)$. We can then sort the similarity scores for each cw_i^S and retain only the top scoring M context words in the final set $Con(w_1^S)$. The procedure of *context sorting and pruning* should improve the semantic cohesion between w_1^S and its context since only informative context features are now present in $Con(w_1^S)$, and we reduce the noise coming from uninformative contextual features that are not semantically related to w_1^S. Other options for the context sorting and pruning are possible, but the main goal in this paper is to illustrate the core utility of the procedure.

3 Cross-Lingual Semantic Similarity in Context via Latent Concepts

Representing Context. The probabilistic framework that is supported by latent cross-lingual concepts allows for having the K-dimensional vector representations in the same latent semantic space spanned by cross-lingual topics for: (1) Single words regardless of their actual language, and (2) Sets that comprise multiple words. Therefore, *we are able to project the observed source word, all target words, and the context set of the observed source word to the same latent semantic space spanned by latent cross-lingual concepts*.

Eq. (1) shows how to represent single words in the latent semantic space. Now, we present a way to address compositionality, that is, we show how to build the same representations in the same latent semantic space beyond the word level. We need to compute a conditional concept distribution for the context set $Con(w_1^S)$, that is, we have to compute the probability scores $P(z_k|Con(w_1^S))$ for each $z_k \in \mathcal{Z}$. Remember that the context $Con(w_1^S)$ is actually a set of r (or M after pruning) words $Con(w_1^S) = \{cw_1^S, \ldots, cw_r^S\}$. Under the *single-topic assumption* (Griffiths et al., 2007) and following Bayes' rule, it holds:

$$P(z_k|Con(w_1^S)) = \frac{P(Con(w_1^S)|z_k)P(z_k)}{P(Con(w_1^S))} = \frac{P(cw_1^S, \ldots, cw_r^S|z_k)P(z_k)}{\sum_{l=1}^{K} P(cw_1^S, \ldots, cw_r^S|z_l)P(z_l)} \quad (2)$$

$$= \frac{\prod_{j=1}^{r} P(cw_j^S|z_k)P(z_k)}{\sum_{l=1}^{K} \prod_{j=1}^{r} P(cw_j^S|z_l)P(z_l)} \quad (3)$$

Note that here we use a simplification where we assume that all $cw_j^S \in Con(w_1^S)$ are conditionally independent given z_k. The assumption of the conditional independence of unigrams is a standard heuristic applied in *bag-of-words* model in NLP and IR (e.g., one may observe a direct analogy to probabilistic language models for IR where the assumption of independence of query words is imposed (Ponte and Croft, 1998; Hiemstra, 1998; Lavrenko and Croft, 2001)), but we have to forewarn the reader that in general the equation $P(cw_1^S, \ldots, cw_r^S|z_k) = \prod_{j=1}^{r} P(cw_j^S|z_k)$ is not exact. However, by adopting the conditional independence assumption, in case of the uniform topic prior $P(z_k)$ (i.e., we assume that we do not posses any prior knowledge about the importance of latent cross-lingual concepts in a multilingual corpus), eq. (3) may be further simplified:

$$P(z_k|Con(w_1^S)) \approx \frac{\prod_{j=1}^{r} P(cw_j^S|z_k)}{\sum_{l=1}^{K} \prod_{j=1}^{r} P(cw_j^S|z_l)} \quad (4)$$

The representation of the context set in the latent semantic space is then:

$$vec(Con(w_1^S)) = [P(z_1|Con(w_1^S)), \ldots, P(z_K|Con(w_1^S))]$$

We can then compute the similarity between words and sets of words given in the same latent semantic space in a uniform way, irrespective of their actual language. We use all these properties when building our context-sensitive CLSS models.

One remark: As a by-product of our modeling approach, by this procedure for computing representations for sets of words, we have in fact paved the way towards compositional cross-lingual models of similarity which rely on latent cross-lingual concepts. Similar to compositional models in monolingual settings (Mitchell and Lapata, 2010; Rudolph and Giesbrecht, 2010; Baroni and Zamparelli, 2010; Socher et al., 2011; Grefenstette and Sadrzadeh, 2011; Blacoe and Lapata, 2012; Clarke, 2012; Socher et al., 2012) and multilingual settings (Hermann and Blunsom, 2014; Kočiský et al., 2014), the representation of a set of words (e.g., a phrase or a sentence) is exactly the same as the representation of a single word; it is simply a K-dimensional real-valued vector. Our work on inducing structured representations of words and

text units beyond words is similar to (Klementiev et al., 2012; Hermann and Blunsom, 2014; Kočiský et al., 2014), but unlike them, we do not need high-quality sentence-aligned parallel data to induce bilingual text representations. Moreover, this work on compositionality in multilingual settings is only preliminary (e.g., we treat phrases and sentences as bags-of-words), and in future work we will aim to include syntactic information in the composition models as already done in monolingual settings (Socher et al., 2012; Hermann and Blunsom, 2013).

Intuition behind the Approach. Going back to our novel CLSS models in context, these models rely on the representations of words and their contexts in the same latent semantic space spanned by latent cross-lingual concepts/topics. The models differ in the way the contextual knowledge is fused with the out-of-context word representations.

The key idea behind these models is to represent a word w_1^S in the latent semantic space as a distribution over the latent cross-lingual concepts, but now with an additional modulation of the representation after taking its local context into account. The modulated word representation in the semantic space spanned by K latent cross-lingual concepts is then:

$$vec(w_1^S, Con(w_1^S)) = [P'(z_1|w_1^S), \ldots, P'(z_K|w_1^S)] \quad (5)$$

where $P'(z_K|w_1^S)$ denotes the recalculated (or modulated) probability score for the conditional concept/topic distribution of w_1^S after observing its context $Con(w_1^S)$. For an illustration of the key idea, see fig. 1. The intuition is that the context helps to disambiguate the true meaning of the occurrence of the word w_1^S. In other words, after observing the context of the word w_1^S, fewer latent cross-lingual concepts will share most of the probability mass in the modulated context-aware word representation.

Model I: Direct-Fusion. The first approach makes the conditional distribution over latent semantic concepts directly dependent on both word w_1^S and its context $Con(w_1^S)$. The probability score $P'(z_k|w_1^S)$ from eq. (5) for each $z_k \in \mathcal{Z}$ is then given as $P'(z_k|w_1^S) = P(z_k|w_1^S, Con(w_1^S))$.

We have to estimate the probability $P(z_k|w_1^S, Con(w_1^S))$, that is, the probability that word w_1^S is assigned to the latent concept/topic z_k given its context $Con(w_1^S)$:

$$P(z_k|w_1^S, Con(w_1^S)) = \frac{P(z_k, w_1^S)P(Con(w_1^S)|z_k)}{\sum_{l=1}^{K} P(z_l, w_1^S)P(Con(w_1^S)|z_l)} \quad (6)$$

Since $P(z_k, w_1^S) = P(w_1^S|z_k)P(z_k)$, if we closely follow the derivation from eq. (3) which shows how to project context into the latent semantic space (and again assume the uniform topic prior $P(z_k)$), we finally obtain the following formula:

$$P'(z_k|w_1^S) \approx \frac{P(w_1^S|z_k)\prod_{j=1}^{r} P(cw_j^S|z_k)}{\sum_{l=1}^{K} P(w_1^S|z_l)\prod_{j=1}^{r} P(cw_j^S|z_l)} \quad (7)$$

The ranking of all words $w_2^T \in V^T$ according to their similarity to w_1^S may be computed by detecting the similarity score between their representation in the K-dimensional latent semantic space and the modulated source word representation as given by eq. (5) and eq. (7) using any of the existing similarity functions (Lee, 1999; Cha, 2007). The similarity score $Sim(w_1^S, w_2^T, Con(w_1^S))$ between some $w_2^T \in V^T$ represented by its vector $vec(w_2^T)$ and the observed word w_1^S given its context $Con(w_1^S)$ is computed as:

$$\begin{aligned} &sim(w_1^S, w_2^T, Con(w_1^S)) \\ &= SF\Big(vec\big(w_1^S, Con(w_1^S)\big), vec\big(w_2^T\big)\Big) \end{aligned} \quad (8)$$

where SF denotes a similarity function. Words are then ranked according to their respective similarity scores and the best scoring candidate may be selected as the best translation of an occurrence of the word w_1^S given its local context. Since the contextual knowledge is integrated directly into the estimation of probability $P(z_k|w_1^S, Con(w_1^S))$, we name this context-aware CLSS model the *Direct-Fusion* model.

Model II: Smoothed-Fusion. The next model follows the modeling paradigm established within the framework of language modeling (LM), where the idea is to "back off" to a lower order N-gram in case we do not possess any evidence about a higher-order N-gram (Jurafsky and Martin, 2000). The idea now is to smooth the representation of a word in the latent semantic space induced only by the words in its local context with the out-of-context type-based representation of that word induced directly from a large training corpus. In other words, the modulated probability score $P'(z_k|w_1^S)$ from eq. (5) is calculated as:

$$P'(z_k|w_1^S) = \lambda_1 P(z_k|Con(w_1^S)) + (1-\lambda_1)P(z_k|w_1^S) \quad (9)$$

where λ_1 is the interpolation parameter, $P(z_k|w_1^S)$ is the out-of-context conditional concept probability score as in eq. (1), and $P(z_k|Con(w_1^S))$ is given by eq. (3). This model compromises between the pure contextual word representation and

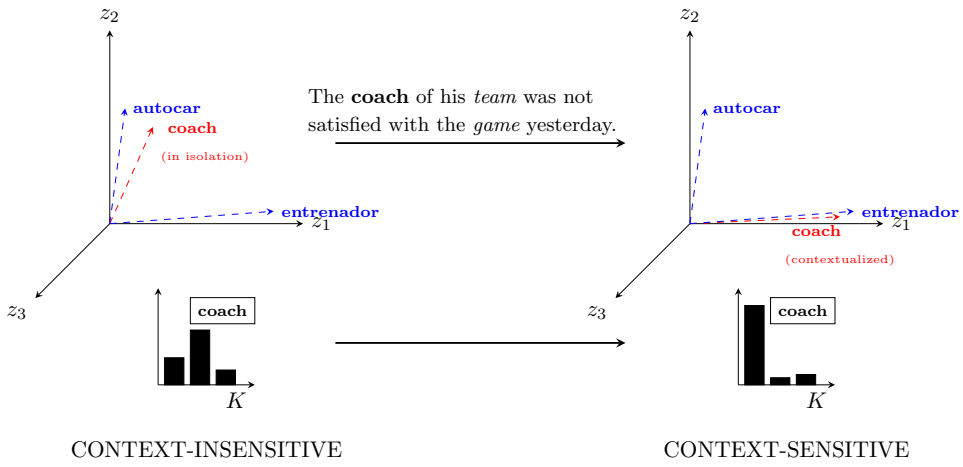


Figure 1: An illustrative toy example of the main intuitions in our probabilistic framework for building context sensitive models with only three latent cross-lingual concepts (axes z_1, z_2 and z_3): A change in meaning is reflected as a change in a probability distribution over latent cross-lingual concepts that span a shared latent semantic space. A change in the probability distribution may then actually steer an English word *coach* towards its correct (Spanish) meaning in context.

the out-of-context word representation. In cases when the local context of word w_1^S is informative enough, the factor $P(z_k|Con(w_1^S))$ is sufficient to provide the ranking of terms in V^T, that is, to detect words that are semantically similar to w_1^S based on its context. However, if the context is not reliable, we have to smooth the pure context-based representation with the out-of-context word representation (the factor $P(z_k|w_1^S)$). We call this model the *Smoothed-Fusion* model.

The ranking of words $w_2^T \in V^T$ then finally proceeds in the same manner as in Direct-Fusion following eq. (8), but now using eq. (9) for the modulated probability scores $P'(z_k|w_1^S)$.

Model III: Late-Fusion. The last model is conceptually similar to Smoothed-Fusion, but it performs smoothing at a later stage. It proceeds in two steps: (1) Given a target word $w_2^T \in V^T$, the model computes similarity scores separately between (i) the context set $Con(w_1^S)$ and w_2^T, and (ii) the word w_1^S in isolation and w_2^T (again, on the type level); (2) It linearly combines the obtained similarity scores. More formally, we may write:

$$\begin{aligned} &Sim(w_1^S, w_2^T, Con(w_1^S)) \\ &= \lambda_2 SF\Big(vec(Con(w_1^S)), vec(w_2^T)\Big) \\ &+ (1-\lambda_2) SF\Big(vec(w_1^S), vec(w_2^T)\Big) \end{aligned} \quad (10)$$

where λ_2 is the interpolation parameter. Since this model computes the similarity with each target word separately for the source word in isolation and its local context, and combines the obtained similarity scores after the computations, this model is called *Late-Fusion*.

4 Experimental Setup

Evaluation Task: Suggesting Word Translations in Context. Given an occurrence of a polysemous word $w_1^S \in V^S$ in the source language L_S with vocabulary V^S, the task is to choose the correct translation in the target language L_T of that particular occurrence of w_1^S from the given set $\mathcal{T} = \{t_1^T, \ldots, t_q^T\}$, $\mathcal{T} \subseteq V^T$, of its q possible translations/meanings (i.e., its translation or sense inventory). The task of *suggesting a word translation in context* may be interpreted as ranking the q translations with respect to the observed local context $Con(w_1^S)$ of the occurrence of the word w_1^S. The best scoring translation candidate in the ranked list is then the suggested correct translation for that particular occurrence of w_1^S after observing its local context $Con(w_1^S)$.

Training Data. We use the following corpora for inducing latent cross-lingual concepts/topics, i.e., for training our multilingual topic model: (i) a collection of 13,696 Spanish-English Wikipedia article pairs (Wiki-ES-EN), (ii) a collection of 18,898 Italian-English Wikipedia article pairs, (iii) a collection of 7,612 Dutch-English Wikipedia article pairs (Wiki-NL-EN), and (iv) the Wiki-NL-EN corpus augmented with 6,206 Dutch-English document pairs from Europarl (Koehn, 2005) (Wiki+EP-NL-EN). The corpora were previously used in (Vulić and Moens, 2013). No explicit use is made of sentence-level alignments in Europarl.

Sentence in Italian	Correct Translation (EN)
1. I primi **calci** furono prodotti in legno ma recentemente...	stock
2. In caso di osteoporosi si verifica un eccesso di rilascio di **calcio** dallo scheletro...	calcium
3. La crescita del **calcio** femminile professionistico ha visto il lancio di competizioni...	football
4. Il **calcio** di questa pistola (Beretta Modello 21a, calibro .25) ha le guancette in materiale...	stock

Table 1: Example sentences from our IT evaluation dataset with corresponding correct translations.

Spanish	**Italian**	**Dutch**
Ambiguous word (Possible senses/translations)	*Ambiguous word* (Possible senses/translations)	*Ambiguous word* (Possible senses/translations)
1. estación (station; season)	*1. raggio* (ray; radius; spoke)	*1. toren* (rook; tower)
2. ensayo (essay; rehearsal; trial)	*2. accordo* (chord; agreement)	*2. beeld* (image; statue)
3. núcleo (core; kernel; nucleus)	*3. moto* (motion; motorcycle)	3. *blade* (blade; leaf; magazine)
4. vela (sail; candle)	*4. calcio* (calcium; football; stock)	*4.fusie* (fusion; merger)
5. escudo (escudo; escutcheon; shield)	*5. terra* (earth; land)	*5. stam* (stem; trunk; tribe)
6. papa (Pope; potato)	*6. tavola* (board; panel; table)	*6. koper* (copper; buyer)
7. cola (glue; coke; tail; queue)	*7. campione* (champion; sample)	*7. bloem* (flower; flour)
8. cometa (comet; kite)	*8. carta* (card; paper; map)	*8. spanning* (voltage; tension; stress)
9. disco (disco; discus; disk)	*9. piano* (floor; plane; plan; piano)	*9. noot* (note; nut)
10. banda (band; gang; strip)	*10. disco* (disco; discus; disk)	*10. akkoord* (chord; agreement)
11. cinta (ribbon; tape)	*11. istruzione* (education; instruction)	*11. munt* (coin; currency; mint)
12. banco (bank; bench; shoal)	*12. gabinetto* (cabinet; office; toilet)	*12. pool* (pole; pool)
13. frente (forehead; front)	*13. torre* (rook; tower)	*13. band* (band; tyre; tape)
14. fuga (escape; fugue; leak)	*14. campo* (camp; field)	*14. kern* (core; kernel; nucleus)
15. gota (gout; drop)	*15. gomma* (rubber; gum; tyre)	*15. kop* (cup; head)

Table 2: Sets of 15 ambiguous words in Spanish, Italian and Dutch from our test set accompanied by the sets of their respective possible senses/translations in English.

All corpora are theme-aligned comparable corpora, i.e, the aligned document pairs discuss similar themes, but are in general not direct translations (except for Europarl). By training on Wiki+EP-NL-EN we want to test how the training corpus of higher quality affects the estimation of latent cross-lingual concepts that span the shared latent semantic space and, consequently, the overall results in the task of suggesting word translations in context. Following prior work (Koehn and Knight, 2002; Haghighi et al., 2008; Prochasson and Fung, 2011; Vulić and Moens, 2013), we retain only nouns that occur at least 5 times in the corpus. We record lemmatized word forms when available, and original forms otherwise. We use TreeTagger (Schmid, 1994) for POS tagging and lemmatization.

Test Data. We have constructed test datasets in Spanish (ES), Italian (IT) and Dutch (NL), where the aim is to find their correct translation in English (EN) given the sentential context. We have selected 15 polysemous nouns (see tab. 2 for the list of nouns along with their possible translations) in each of the 3 languages, and have manually extracted 24 sentences (not present in the training data) for each noun that capture different meanings of the noun from Wikipedia. In order to construct datasets that are balanced across different possible translations of a noun, in case of q different translation candidates in $\mathcal{T}$ for some word w_1^S, the dataset contains exactly $24/q$ sentences for each translation from $\mathcal{T}$. In total, we have designed 360 sentences for each language

pair (ES/IT/NL-EN), 1080 sentences in total.[1]. We have used 5 extra nouns with 20 sentences each as a development set to tune the parameters of our models. As a by-product, we have built an initial repository of ES/IT/NL ambiguous words. Tab. 1 presents a small sample from the IT evaluation dataset, and illustrates the task of suggesting word translations in context.

Evaluation Procedure. Our task is to present the system a list of possible translations and let the system decide a *single most likely translation* given the word and its sentential context. Ground truth thus contains one word, that is, one correct translation for each sentence from the evaluation dataset. We have manually annotated the correct translation for the ground truth[1] by inspecting the discourse in Wikipedia articles and the interlingual Wikipedia links. We measure the performance of all models as *Top 1* accuracy (Acc_1) (Gaussier et al., 2004; Tamura et al., 2012). It denotes the number of word instances from the evaluation dataset whose top proposed candidate in the ranked list of translation candidates from $\mathcal{T}$ is exactly the correct translation for that word instance as given by ground truth over the total number of test word instances (360 in each test dataset).

Parameters. We have tuned λ_1 and λ_2 on the development sets. We set $\lambda_1 = \lambda_2 = 0.9$ for all language pairs. We use sorted context sets (see sect. 2) and perform a cut-off at $M = 3$ most descriptive context words in the sorted context sets for all models. In the following section we discuss the utility of this context sorting and pruning, as well as its influence on the overall results.

Inducing Latent Cross-Lingual Concepts. Our context-aware models are generic and allow experimentations with different models that induce latent cross-lingual semantic concepts. However, in this particular work we present results obtained by a multilingual probabilistic topic model called bilingual LDA (Mimno et al., 2009; Ni et al., 2009; De Smet and Moens, 2009). The BiLDA model is a straightforward multilingual extension of the standard LDA model (Blei et al., 2003). For the details regarding the modeling, generative story and training of the bilingual LDA model, we refer the interested reader to the aforementioned relevant literature.

We have used the Gibbs sampling procedure (Geman and Geman, 1984) tailored for BiLDA in particular for training and have experimented with different number of topics K in the interval $300 - 2500$. Here, we present only the results obtained with $K = 2000$ for all language pairs which also yielded the best or near-optimal performance in (Dinu and Lapata, 2010b; Vulić et al., 2011). Other parameters of the model are set to the typical values according to Steyvers and Griffiths (2007): $\alpha = 50/K$ and $\beta = 0.01$. [2]

Models in Comparison. We test the performance of our Direct-Fusion, Smoothed-Fusion and Late-Fusion models, and compare their results with the context-insensitive CLSS models described in sect. 2 (*No-Context*). We provide results with two different similarity functions: (1) We have tested different SF-s (e.g., the Kullback-Leibler and the Jensen-Shannon divergence, the cosine measure) on the K-dimensional vector representations, and have detected that in general the best scores are obtained with the Bhattacharyya coefficient (*BC*) (Cha, 2007; Kazama et al., 2010), (2) Another similarity method we use is the so-called *Cue* method (Griffiths et al., 2007; Vulić et al., 2011), which models the probability that a target word t_i^T will be generated as an association response given some cue source word w_1^S. In short, the method computes the score $P(t_i^T|w_1^S) = P(t_i^T|z_k)P(z_k|w_1^S)$. We can use the scores $P(t_i^T|w_1^S)$ obtained by inputting out-of-context probability scores $P(z_k|w_1^S)$ or modulated probability scores $P'(z_k|w_1^S)$ to produce the ranking of translation candidates.

5 Results and Discussion

The performance of all the models in comparison is displayed in tab. 3. These results lead us to several conclusions:

(i) All proposed context-sensitive CLSS models suggesting word translations in context significantly outperform context-insensitive CLSS models, which are able to produce only word translations in isolation. The improvements in results when taking context into account are ob-

[1] Available at `http://people.cs.kuleuven.be/~ivan.vulic/software/`

[2] We are well aware that different hyper-parameter settings (Asuncion et al., 2009; Lu et al., 2011), might have influence on the quality of learned latent cross-lingual concepts/topics and, consequently, the quality of latent semantic space, but that analysis is not the focus of this work. Additionally, we perform *semantic space pruning* (Reisinger and Mooney, 2010; Vulić and Moens, 2013). All computations are performed over the best scoring 100 cross-lingual topics according to their respective scores $P(z_k|w_i^S)$ similarly to (Vulić and Moens, 2013).

Direction:	ES→EN		IT→EN		NL→EN (Wiki)		NL→EN (Wiki+EP)	
Model	Acc_1 (SF=BC)	Acc_1 (SF=Cue)	Acc_1 (SF=BC)	Acc_1 (SF=Cue)	Acc_1 (SF=BC)	Acc_1 (SF=Cue)	Acc_1 (SF=BC)	Acc_1 (SF=Cue)
No-Context	.406	.406	.408	.408	.433	.433	.433	.433
Direct-Fusion	.617	.575	.714	.697	.603	.592	.606	.636
Smoothed-Fusion	.664	**.703***	.731	**.789***	**.669**	**.712***	**.692**	**.761***
Late-Fusion	**.675**	.667	**.742**	.728	.667	.644	.683	.722

Table 3: Results on the 3 evaluation datasets. Translation direction is ES/IT/NL→EN. The improvements of all contextualized models over non-contextualized models are statistically significant according to a chi-square statistical significance test ($p<0.05$). The asterisk (*) denotes significant improvements of Smoothed-Fusion over Late-Fusion using the same significance test.

served for all 3 language pairs. The large improvements in the results (i.e., we observe an average relative increase of 51.6% for the BC+Direct-Fusion combination, 64.3% for BC+Smoothed-Fusion, 64.9% for BC+Late-Fusion, 49.1% for Cue+Direct-Fusion, 76.7% for Cue+Smoothed-Fusion, and 64.5% for Cue+Late-Fusion) confirm that the local context of a word is essential in acquiring correct word translations for polysemous words, as isolated non-contextualized word representations are not sufficient.

(ii) The choice of a similarity function influences the results. On average, the Cue method as SF outperforms other standard similarity functions (e.g., Kullback-Leibler, Jensen-Shannon, cosine, BC) in this evaluation task. However, it is again important to state that *regardless of the actual choice of SF, context-aware models that modulate out-of-context word representations using the knowledge of local context outscore context-insensitive models that utilize non-modulated out-of-context representations* (with all other parameters equal).

(iii) The Direct-Fusion model, conceptually similar to a model of word similarity in context in monolingual settings (Dinu and Lapata, 2010a), is outperformed by the other two context-sensitive models. In Direct-Fusion, the observed word and its context are modeled in the same fashion, that is, the model does not distinguish between the word and its surrounding context when it computes the modulated probability scores $P'(z_k|w_1^S)$ (see eq. (7)). Unlike Direct-Fusion, the modeling assumptions of Smoothed-Fusion and Late-Fusion provide a clear distinction between the observed word w_1^S and its context $Con(w_1^S)$ and combine the out-of-context representation of w_1^S and its contextual knowledge into a smoothed LM-inspired probabilistic model. As the results reveal, that strategy leads to better overall scores. The best scores in general are obtained by Smoothed-Fusion, but it is also outperformed by Late-Fusion in several experimental runs where BC was used as SF. However, the difference in results between Smoothed-Fusion and Late-Fusion in these experimental runs is not statistically significant according to a chi-squared significance test ($p < 0.05$).

(iv) The results for Dutch-English are influenced by the quality of training data. The performance of our models of similarity is higher for models that rely on latent-cross lingual topics estimated from the data of higher quality (i.e., compare the results when trained on Wiki and Wiki+EP in tab. 3). The overall quality of our models of similarity is of course dependent on the quality of the latent cross-lingual topics estimated from training data, and the quality of these latent cross-lingual concepts is further dependent on the quality of multilingual training data. This finding is in line with a similar finding reported for the task of bilingual lexicon extraction (Vulić and Moens, 2013).

(v) Although Dutch is regarded as more similar to English than Italian or Spanish, we do not observe any major increase in the results on both test datasets for the English-Dutch language pair compared to English-Spanish/Italian. That phenomenon may be attributed to the difference in size and quality of our training Wikipedia datasets. Moreover, while the probabilistic framework proposed in this chapter is completely language pair agnostic as it does not make any language pair dependent modeling assumptions, we acknowledge the fact that all three language pairs comprise languages coming from the same phylum, that is, the Indo-European language family. Future extensions of our probabilistic modeling framework also include porting the framework to other more distant language pairs that do not share the same roots nor the same alphabet (e.g., English-Chinese/Hindi).

Analysis of Context Sorting and Pruning. We

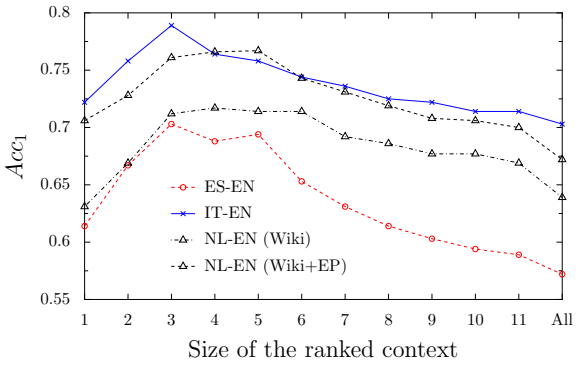


Figure 2: The influence of the size of sorted context on the accuracy of word translation in context. The model is Cue+Smoothed-Fusion.

also investigate the utility of context sorting and pruning, and its influence on the overall results in our evaluation task. Therefore, we have conducted experiments with sorted context sets that were pruned at different positions, ranging from 1 (only the most similar word to w_1^S in a sentence is included in the context set $Con(w_1^S)$) to *All* (all words occurring in a same sentence with w_1^S are included in $Con(w_1^S)$). The monolingual similarity between w_1^S and each potential context word in a sentence has been computed using BC on their out-of-context representations in the latent semantic space spanned by cross-lingual topics. Fig. 2 shows how the size of the sorted context influences the overall results. The presented results have been obtained by the Cue+Smoothed-Fusion combination, but a similar behavior is observed when employing other combinations.

Fig. 2 clearly indicates the importance of context sorting and pruning. The procedure ensures that only the most semantically similar words in a given scope (e.g., a sentence) influence the choice of a correct meaning. In other words, closely semantically similar words in the same sentence are more reliable indicators for the most probable word meaning. They are more informative in modulating the out-of-context word representations in context-sensitive similarity models. We observe large improvements in scores when we retain only the top M semantically similar words in the context set (e.g., when M=5, the scores are 0.694, 0.758, 0.717, and 0.767 for ES-EN, IT-EN, NL-EN (Wiki) and NL-EN (Wiki+EP), respectively; while the same scores are 0.572, 0.703, 0.639 and 0.672 when M=*All*).

6 Conclusions and Future Work

We have proposed a new probabilistic approach to modeling cross-lingual semantic similarity in context, which relies only on co-occurrence counts and latent cross-lingual concepts which can be estimated using only comparable data. The approach is purely statistical and it does not make any additional language-pair dependent assumptions; it does not rely on a bilingual lexicon, orthographic clues or predefined ontology/category knowledge, and it does not require parallel data.

The key idea in the approach is to represent words, regardless of their actual language, as distributions over the latent concepts, and both out-of-context and contextualized word representations are then presented in the same latent space spanned by the latent semantic concepts. A change in word meaning after observing its context is reflected in a change of its distribution over the latent concepts. Results for three language pairs have clearly shown the importance of the newly developed modulated or "contextualized" word representations in the task of suggesting word translations in context.

We believe that the proposed framework is only a start, as it ignites a series of new research questions and perspectives. One may further examine the influence of context scope (e.g., document-based vs. sentence-based vs. window-based contexts), as well as context selection and aggregation (see sect. 2) on the contextualized models. For instance, similar to the model from Ó Séaghdha and Korhonen (2011) in the monolingual setting, one may try to introduce dependency-based contexts (Padó and Lapata, 2007) and incorporate the syntax-based knowledge in the context-aware CLSS modeling. It is also worth studying other models that induce latent semantic concepts from multilingual data (see sect. 2) within this framework of context-sensitive CLSS modeling. One may also investigate a similar approach to context-sensitive CLSS modeling that could operate with explicitly defined concept categories (Gabrilovich and Markovitch, 2007; Cimiano et al., 2009; Hassan and Mihalcea, 2009; Hassan and Mihalcea, 2011; McCrae et al., 2013).

Acknowledgments

We would like to thank the anonymous reviewers for their comments and suggestions. This research has been carried out in the framework of the Smart Computer-Aided Translation Environment (SCATE) project (IWT-SBO 130041).

References

Mirna Adriani and C. J. van Rijsbergen. 1999. Term similarity-based query expansion for cross-language information retrieval. In *Proceedings of the 3rd European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pages 311–322.

Marianna Apidianaki. 2011. Unsupervised cross-lingual lexical substitution. In *Proceedings of the 1st Workshop on Unsupervised Learning in NLP*, pages 13–23.

Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. 2009. On smoothing and inference for topic models. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 27–34.

Lisa Ballesteros and W. Bruce Croft. 1997. Phrasal translation and query expansion techniques for cross-language information retrieval. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 84–91.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1183–1193.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 546–556.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jordan Boyd-Graber and David M. Blei. 2009. Multilingual topic models for unaligned text. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 75–82.

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

Sung-Hyuk Cha. 2007. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307.

Philipp Cimiano, Antje Schultz, Sergej Sizov, Philipp Sorg, and Steffen Staab. 2009. Explicit versus latent concept models for cross-language information retrieval. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence (IJCAI)*, pages 1513–1518.

Daoud Clarke. 2012. A context-theoretic framework for compositionality in distributional semantics. *Computational Linguistics*, 38(1):41–71.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 600–609.

Hal Daumé III and Jagadeesh Jagarlamudi. 2011. Domain adaptation for machine translation by mining unseen words. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 407–412.

Wim De Smet and Marie-Francine Moens. 2009. Cross-language linking of news stories on the Web using interlingual topic modeling. In *Proceedings of the CIKM 2009 Workshop on Social Web Search and Mining (SWSM@CIKM)*, pages 57–64.

Chris H. Q. Ding, Tao Li, and Wei Peng. 2008. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, 52(8):3913–3927.

Georgiana Dinu and Mirella Lapata. 2010a. Measuring distributional similarity in context. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1162–1172.

Georgiana Dinu and Mirella Lapata. 2010b. Topic models for meaning similarity in context. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 250–258.

Susan T. Dumais, Thomas K. Landauer, and Michael Littman. 1996. Automatic cross-linguistic information retrieval using Latent Semantic Indexing. In *Proceedings of the SIGIR Workshop on Cross-Linguistic Information Retrieval*, pages 16–23.

Greg Durrett, Adam Pauls, and Dan Klein. 2012. Syntactic transfer using a bilingual lexicon. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1–11.

Kosuke Fukumasu, Koji Eguchi, and Eric P. Xing. 2012. Symmetric correspondence topic models for multilingual text analysis. In *Procedings of the 25th Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 1295–1303.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1606–1611.

Kuzman Ganchev and Dipanjan Das. 2013. Cross-lingual discriminative learning of sequence models with posterior regularization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1996–2006.

Éric Gaussier and Cyril Goutte. 2005. Relation between PLSA and NMF and implications. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 601–602.

Éric Gaussier, Jean-Michel Renders, Irina Matveeva, Cyril Goutte, and Hervé Déjean. 2004. A geometric view on bilingual lexicon extraction from comparable corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 526–533.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1394–1404.

Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114(2):211–244.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 771–779.

Samer Hassan and Rada Mihalcea. 2009. Cross-lingual semantic relatedness using encyclopedic knowledge. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1192–1201.

Samer Hassan and Rada Mihalcea. 2011. Semantic relatedness using salient semantic analysis. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI)*, pages 884–889.

Karl Moritz Hermann and Phil Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 894–904.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 58–68.

Djoerd Hiemstra. 1998. A linguistically motivated probabilistic model of information retrieval. In *Proceedings of the 2nd European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pages 569–584.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 873–882.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR.

Jun'ichi Kazama, Stijn De Saeger, Kow Kuroda, Masaki Murata, and Kentaro Torisawa. 2010. A Bayesian method for robust estimation of distributional similarities. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 247–256.

Sungchul Kim, Kristina Toutanova, and Hwanjo Yu. 2012. Multilingual named entity recognition using parallel data and metadata from Wikipedia. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 694–702.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 1459–1474.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of the ACL Workshop on Unsupervised Lexical Acquisition (ULA)*, pages 9–16.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the 10th Machine Translation Summit (MT SUMMIT)*, pages 79–86.

Tomáš Kočiský, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 224–229.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance-based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 120–127.

Victor Lavrenko, Martin Choquette, and W. Bruce Croft. 2002. Cross-lingual relevance models. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 175–182.

Daniel D. Lee and H. Sebastian Seung. 1999. Algorithms for non-negative matrix factorization. In *Proceedings of the 12th Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 556–562.

Lillian Lee. 1999. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 25–32.

Gina-Anne Levow, Douglas W. Oard, and Philip Resnik. 2005. Dictionary-based techniques for cross-language information retrieval. *Information Processing and Management*, 41(3):523–547.

Yue Lu, Qiaozhu Mei, and Chengxiang Zhai. 2011. Investigating task performance of probabilistic topic models: An empirical study of PLSA and LDA. *Information Retrieval*, 14(2):178–203.

John Philip McCrae, Philipp Cimiano, and Roman Klinger. 2013. Orthonormal explicit topic analysis for cross-lingual document matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1732–1740.

Paola Merlo, Suzanne Stevenson, Vivian Tsang, and Gianluca Allaria. 2002. A multilingual paradigm for automatic verb classification. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 207–214.

David Mimno, Hanna Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 880–889.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 236–244.

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

Hwee Tou Ng, Bin Wang, and Yee Seng Chan. 2003. Exploiting parallel texts for word sense disambiguation: An empirical study. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 455–462.

Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. 2009. Mining multilingual topics from Wikipedia. In *Proceedings of the 18th International World Wide Web Conference (WWW)*, pages 1155–1156.

Diarmuid Ó Séaghdha and Anna Korhonen. 2011. Probabilistic models of similarity in syntactic context. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1047–1057.

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

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

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36:307–340.

Yves Peirsman and Sebastian Padó. 2010. Cross-lingual induction of selectional preferences with bilingual vector spaces. In *Proceedings of the 11th Meeting of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 921–929.

Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 275–281.

Anat Prior, Shuly Wintner, Brian MacWhinney, and Alon Lavie. 2011. Translation ambiguity in and out of context. *Applied Psycholinguistics*, 32(1):93–111.

Emmanuel Prochasson and Pascale Fung. 2011. Rare word translation extraction from aligned comparable documents. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 1327–1335.

Joseph Reisinger and Raymond J. Mooney. 2010. A mixture model with sharing for lexical semantics. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1182.

Sebastian Rudolph and Eugenie Giesbrecht. 2010. Compositional matrix-space models of language. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 907–916.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of the 24th Annual Conference on Advances in Neural*

Information Processing Systems (NIPS), pages 801–809.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1201–1211.

Mark Steyvers and Tom Griffiths. 2007. Probabilistic topic models. *Handbook of Latent Semantic Analysis*, 427(7):424–440.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013a. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of ACL*, 1:1–12.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013b. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 14th Meeting of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1061–1071.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2012. Bilingual lexicon extraction from comparable corpora using label propagation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 24–36.

Michael E. Tipping and Christopher M. Bishop. 1999. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482.

Lonneke van der Plas, Paola Merlo, and James Henderson. 2011. Scaling up automatic cross-lingual semantic role annotation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 299–304.

Ivan Vulić and Marie-Francine Moens. 2013. Cross-lingual semantic similarity of words as the similarity of their semantic word responses. In *Proceedings of the 14th Meeting of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 106–116.

Ivan Vulić, Wim De Smet, and Marie-Francine Moens. 2011. Identifying word translations from comparable corpora using latent topic models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 479–484.

Ivan Vulić, Wim De Smet, and Marie-Francine Moens. 2013. Cross-language information retrieval models based on latent topic models trained with document-aligned comparable corpora. *Information Retrieval*, 16(3):331–368.

Jianqiang Wang and Douglas W. Oard. 2006. Combining bidirectional translation and synonymy for cross-language information retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 202–209.

Hua Wu, Haifeng Wang, and Chengqing Zong. 2008. Domain adaptation for statistical machine translation with domain dictionary and monolingual corpora. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 993–1000.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 200–207.

Duo Zhang, Qiaozhu Mei, and ChengXiang Zhai. 2010. Cross-lingual latent topic extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1128–1137.

Hai Zhao, Yan Song, Chunyu Kit, and Guodong Zhou. 2009. Cross language dependency parsing using a bilingual lexicon. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 55–63.

Multi-Predicate Semantic Role Labeling

Haitong Yang and Chengqing Zong
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
{htyang, cqzong}@nlpr.ia.ac.cn

Abstract

The current approaches to Semantic Role Labeling (SRL) usually perform role classification for each predicate separately and the interaction among individual predicate's role labeling is ignored if there is more than one predicate in a sentence. In this paper, we prove that different predicates in a sentence could help each other during SRL. In multi-predicate role labeling, there are mainly two key points: argument identification and role labeling of the arguments shared by multiple predicates. To address these issues, in the stage of argument identification, we propose novel predicate-related features which help remove many argument identification errors; in the stage of argument classification, we adopt a discriminative reranking approach to perform role classification of the shared arguments, in which a large set of global features are proposed. We conducted experiments on two standard benchmarks: Chinese PropBank and English PropBank. The experimental results show that our approach can significantly improve SRL performance, especially in Chinese PropBank.

1 Introduction

Semantic Role Labeling (SRL) is a kind of shallow semantic parsing task and its goal is to recognize some related phrases and assign a joint structure (WHO did WHAT to WHOM, WHEN, WHERE, WHY, HOW) to each predicate of a sentence (Gildea and Jurafsky, 2002). Because of the ability of encoding semantic information, SRL has been applied in many tasks of NLP, such as question and answering (Narayanan and Harabagir, 2004), information extraction (Surdeanu et al., 2003; Christensen et al., 2005), and machine translation (Wu and Fung, 2009; Liu and Gildea, 2010; Xiong et al., 2012; Zhai et al., 2012).

The justices will be **forced** to **reconsider** the questions.

[A1] [Pred]

[A0] [Pred] [A1]

Figure 1: A sentence from English PropBank, with an argument shared by multiple predicates

Currently, an SRL system works as follows: first identify argument candidates and then perform classification for each argument candidate. However, this process only focuses on one independent predicate without considering the internal relations of multiple predicates in a sentence. According to our statistics, more than 80% sentences in Propbank carry multiple predicates. One example is shown in Figure 1, in which there are two predicates 'Force' and 'Reconsider'. Moreover, the constituent 'the justices' is shared by the two predicates and is labeled as A1 for 'Force' but as A0 for 'Reconsider'. We call this phenomenon of the shared arguments **Role Transition** . Intuitively, all predicates in a sentence are closely related to each other and the internal relations between them would be helpful for SRL.

This paper has made deep investigation on multi-predicate semantic role labeling. We think there are mainly two key points: argument identification and role labeling of the arguments shared by multiple predicates. We adopt different strategies to address these two issues.

During argument identification, there are a large number of identification errors caused by the poor performance of auto syntax trees. However, many of these errors can be removed, if we take other predicates into consideration. To achieve this purpose, we propose novel predicates-related features which have been proved to be effective to recog-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 363–373,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

nize many identification errors. After these features added, the precision of argument identification improves significantly by 1.6 points and 0.9 points in experiments on Chinese PropBank and English PropBank respectively, with a slight loss in recall.

Role labeling of the shared arguments is another key point. The predicates and their shared argument could be considered as a joint structure, with strong dependencies between the shared argument's roles. If we consider linguistic basis for joint modeling of the shared argument's roles, there are at least two types of information to be captured. The first type of information is the compatibility of Role Transition among the shared argument's roles. A noun phrase may be labeled as A0 for a predicate and at the same time, it can be labeled as A1 for another predicate. However, there are few cases that a noun phrase is labeled as A0 for a predicate and as AM-ADV for another predicate at the same time. Secondly, joint modeling the shared arguments could explore global information. For example, in '*The columbia mall is expected to open*', there are two predicates 'expect' and 'open' and a shared argument 'the columbia mall'. Because this shared argument is before 'open' and the predicate 'open' is in active voice, a base classifier often incorrectly label this argument A0 for 'open'. But if we observe that the argument is also an argument of 'expect', it should be labeled as A1 for 'expect' and 'open'.

Motivated by the above observations, we attempt to jointly model the shared arguments' roles. Specifically, we utilize the discriminative reranking approach that has been successfully employed in many NLP tasks. Typically, this method first creates a list of n-best candidates from a base system, and then reranks them with arbitrary features (both local and global), which are either not computable or are computationally intractable within the base model.

We conducted experiments on Chinese PropBank and English PropBank. Results show that compared with a state-of-the-art base model, the accuracy of our joint model improves significantly by 2.4 points and 1.5 points on Chinese PropBank and English PropBank respectively, which suggests that there are substantial gains to be made by jointly modeling the shared arguments of multiple predicates.

Our contributions can be summarized as follows:

- To the best of our knowledge, this is the first work to investigate the mutual effect of multiple predicates' semantic role labeling.
- We present a rich set of features for argument identification and shared arguments' classification that yield promising performance.
- We evaluate our method on two standard benchmarks: Chinese PropBank and English PropBank. Our approach performs well in both, which suggests its good universality.

The remainder of this paper is organized as follows. Section 2 gives an overview of our approach. We discuss the mutual effect of multi-predicate' argument identification and argument classification in Section 3 and Section 4 respectively. The experiments and results are presented in Section 5. Some discussion and analysis can be found in Section 6. Section 7 discusses the related works. Finally, the conclusion and future work are in Section 8.

2 Approach Overview

As illustrated in Figure 2, our approach follows the standard separation of the task of semantic role labeling into two phases: **Argument Identification** and **Argument Classification** . We investigate the effect of multiple predicates in Argument Identification and Argument Classification respectively. Specifically, in the stage of Argument Identification, we introduce new features related to predicates which are effective to recognize many argument identification errors. In the stage of Argument Classification, we concentrate on the classification of the arguments shared by multiple predicates. We first use a base model to generate n-best candidates for the shared arguments and then construct a joint model to rerank the n-best list, in which a rich set of global features are proposed.

3 Argument Identification

In this section, we investigate multi-predicate' mutual effects in Argument Identification. Argument Identification is to recognize the arguments from all candidates of each predicate. Here, we use the Maximum Entropy (ME) classifier to perform binary classification. As a discriminative model, ME can easily incorporate arbitrary features and

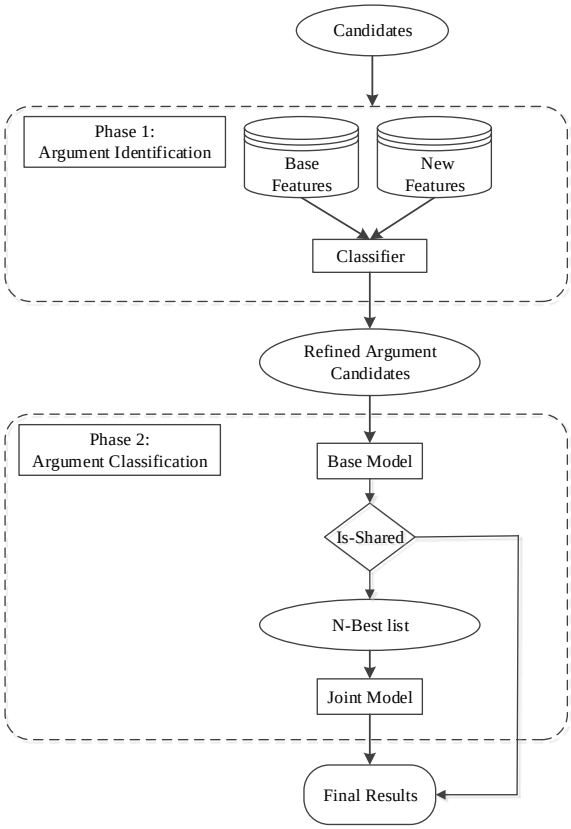


Figure 2: The overview of our approach

achieve good performance. The model is formulated as follows:

$$p(y|x) = \frac{1}{Z(x)} \exp(\sum_i \theta_i f_i(x, y)) \quad (1)$$

in which x is the input sample, y(0 or 1) is the output label, $f(x, y)$ are feature functions and $Z(x)$ is a normalization term as follows:

$$Z(x) = \sum_y \exp(\sum_i \theta_i f_i(x, y))$$

3.1 Base Features

Xue (2008) took a critical look at the features used in SRL and achieved good performance. So, we use the same features in Xue (2008) as the base features:

- Predicate lemma
- Path from node to predicate
- Head word
- Head word's part-of-speech
- Verb class (Xue, 2008)
- Predicate and Head word combination
- Predicate and Phrase type combination
- Verb class and Head word combination
- Verb class and Phrase type combination

3.2 Additional Features

In the SRL community, it is widely recognized that the overall performance of a system is largely determined by the quality of syntactic parsers (Gildea and Palmer, 2002), which is particularly notable in the identification stage. Unfortunately, the state-of-the-art auto parsers fall short of the demands of applications. Moreover, when there are multiple predicates, or even multiple clauses in a sentence, the problem of syntactic ambiguity increases drastically (Kim et al., 2000). For example, in Figure 3, there is a sentence with two consecutive predicates '是' (is) and '开发' (develop). Compared with the gold tree, the auto tree is less preferable, which makes the classifier easily mistake '建筑' (building) as an argument of '开发' (develop) with base features. But this identification error can be removed if we note that there is another predicate '是' (is) before '开发' (devel-

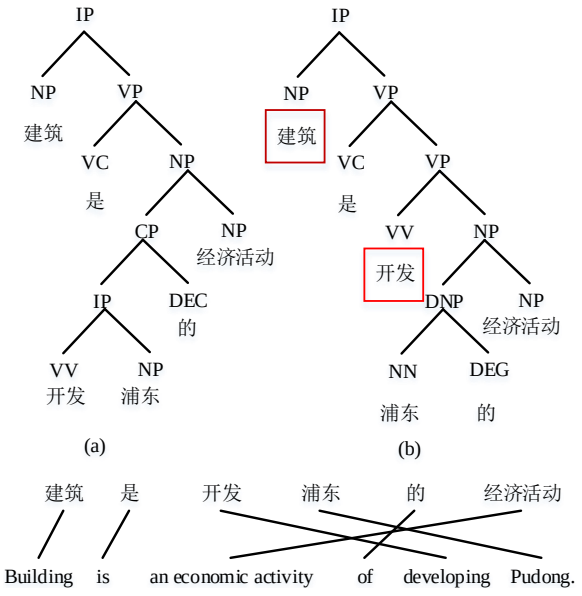


Figure 3: An example from Chinese PropBank. Tree (a) is the gold syntax tree and (b) is parsed by a state of-the-art parser Berkeley parser. On tree (b), '建筑' (building) is mistaken as an argument of '开发' (develop) with base features.

op). Similar examples with the pattern 'NP + 是 + VV' can be found in PropBank, in which the subject NP of the sentence is usually not an argument of the latter predicate. Thus, '是' (is) is an effective clue to detect this kind of identification error.

It is challenging to obtain a fully correct syntax tree for a complex sentence with multiple predicates. Therefore, base features that heavily rely on syntax trees often fail in discriminating arguments from candidates as demonstrated in Figure 3. However, by considering the elements of neighboring predicates, we could capture useful clues like in the above example and remove many identification errors. Below, we define novel predicaterelated features to encode these 'clues' to refine candidates.

There are mainly five kinds of features as follows.

- **Is the given predicate the nearest one?**

  This is a binary feature that indicates whether the predicate is the nearest one to the candidate.

- **Local adjunct**

  This is a binary feature designed for adjective and adverbial phrases. Some adjunct phrases, such as '仅' (only), have a limited sphere of influence. If the candidate is 'local' but the given predicate is not the nearest one, the candidate is often not an argument for the given predicate. To collect local adjuncts, we traverse the whole training set to get the initial lexicon of adjuncts and refine it manually.

- **Cut-Clause**

  This type of feature is a binary feature designed to distinguish identification errors of noun phrase candidates. If a noun phrase candidate is separated from the given predicate by a clause consisting of a NP and VP, the candidate is usually not the argument of the given predicate.

- **Different Relative Positions with Conjunctions**

  This is a binary feature that describes whether the candidate and the predicate are located in different positions as separated by conjunctions such as '但是' (but). Conjunctions are often used to concatenate two clauses, but the first clause commonly describes one proposition and the second clause describes another one. Thus, if the candidate and the given predicate have different positions relative to the conjunctions, the candidate is usually not the argument of the given predicate.

- **Consecutive Predicates Sequence**

  When multiple predicates appear in a sentence consecutively, parse errors frequently occurs due to the problems of syntactic ambiguity as demonstrated in Figure 2. To indicate such errors, sequence features of the candidates and consecutive predicates are defined specifically. For instance, for the candidate '建筑' (building) of '开发' (develop), the features are 'cand-是-开发' and 'cand-是-VV', in which we use 'cand' to represent the position of the candidate.

4 Argument Classification

In this section, we investigate multi-predicate' mutual effects in Argument Classification. Argument Classification is to assign a label to each argument candidate recognized by the phase of Argument Identification.

4.1 Base Model

A conventional method in Argument Classification is to assign a label to each argument candidate by a classifier independently. We call this kind of method Base Model. In the base model, we still adopt ME (1) as our classifier; all base features of Argument Identification are contained (shown in subsection 3.1). In addition, there are some other features:

- Position: the relative position of the candidate argument compared to the predicate
- Subcat frame: the syntactic rule that expands the parent of the verb
- The first and the last word of the candidate
- Phrase type: the syntactic tag of the candidate argument
- Subcat frame+: the frame that consists of the NPs (Xue, 2008).

4.2 Joint Model

As discussed briefly in Section 1, there are many dependencies between the shared arguments' labeling for different predicates, but the base model completely ignores such useful information. To incorporate these dependencies, we employ the discriminative reranking method. Here, we first establish a unified framework for reranking. For an input x, the generic reranker selects the best output y^* among the set of candidates $GEN(x)$ according to the scoring function:

$$y^* = \underset{y \in GEN(x)}{\operatorname{argmax}} \; score(y) \tag{2}$$

In our task, $GEN(x)$ is a set of the n-best candidates generated from the base model. As usual, we calculate the score of a candidate by the dot product between a high dimensional feature and a weight W:

$$score(y) = W \cdot f(y) \tag{3}$$

We estimate the weight W using the averaged perceptron algorithm (Collins, 2002a) which is well known for its fast speed and good performance in similar large-parameter NLP tasks (Huang, 2008). The training algorithm of the generic averaged perceptron is shown in Table 1. In line 5, the algorithm updates W with the difference (if any) between the feature representations of the best scoring candidate and the gold candidate. We also use a refinement called "averaged parameters" that the final weight vector W is the average of weight vectors over T iterations and N samples. This averaging effect has been shown to reduce overfitting and produces more stable results (Collins, 2002a).

Pseudocode: Averaged Structured Perceptron
1: **Input**: training data(x_t, y_t^*) for $t = 1, ..., T$;
2: $\bar{w}^{(0)} \leftarrow 0$; $v \leftarrow 0$; $i \leftarrow 0$
3: **for** n in 1,...,N **do**
4: **for** t in $1, ..., T$ **do**
5: $\bar{w}^{(i+1)} \leftarrow$ update $\bar{w}^{(i)}$ according to (x_t, y_t^*)
6: $v \leftarrow v + \bar{w}^{i+1}$
7: $i \leftarrow i + 1$
8: $\bar{w} \leftarrow v//(N * T)$
9: **return** $\bar{w}$

Table 1: The perceptron training algorithm

4.3 Features for Joint Model

Here, we introduce features used in the joint model. For clear illustration, we describe these features in the context of the example in Figure 1.

Role Transition (RT): a binary feature to indicate whether the transitions among roles of the candidate are reasonable. Because all roles are assigned to the same candidate, all role transitions should be compatible. For instance, if an argument is labeled as AM-TMP for one predicate, it cannot be labeled as AM-LOC for another predicate. This feature is constructed by traversing the training data to ascertain whether transitions between all roles are reasonable. In Table 2, we list some role transitions which are obtained from the training data of experiments on Chinese PropBank.

Roles and Predicates' Sequence (RPS): a joint feature template that concatenates roles and the given predicates. For the gold candidate 'Arg1, Arg0', the feature is 'Arg1-force, Arg0-reconsider'.

Roles and Predicates' Sequence with Relative Orders (RPSWR): the template is similar to the above one except that relative orders between roles and predicates are added. If the shared argument is before the given predicate, the feature is described as 'Role-Predicate'; otherwise, the feature is 'Predicate-Role'. And, if the predicate's voice is passive, the order is reversed. Thus, for the gold candidate 'Arg1, Arg0', this feature is 'force-Arg1, Arg0-reconsider'.

Roles and Phrase Type Sequence (RPTS)

Roles and Head Word Sequence (RHWS)

Roles and Head Word's POS Sequence (RHWPS)

These three features are utilized to explore the shared argument's relations with roles.

Time and Location Class (TLC): We find there are much confusions between AM-TMP and AM-LOC in the base model. To fix these errors, we add two features: Time and Location Class. For these features, we just collect phrases labeled as AM-TMP and AM-LOC from the training data. When the argument belongs to Time or Location Class, we add a sequence template consisting of 'Role-Time' for Time Class or 'Role-Location' for Location Class. For the gold candidate 'Arg1, Arg0', the feature is 'Arg1-none, Arg0-none' because 'the justices' belongs neither to Time Class nor to Location Class.

Role	Arg0	Arg1	Arg2	AM-LOC	AM-TMP	AM-ADV	AM-MNR	AM-TPC
Arg0	+	+	+	+	+	+	+	+
Arg1	+	+	+	+	-	+	+	+
Arg2	+	+	+	+	-	-	-	+
AM-LOC	+	+	+	+	-	+	-	+
AM-TMP	+	-	-	-	+	+	-	-
AM-ADV	+	-	+	-	+	+	-	-
AM-MNR	+	+	-	-	-	-	+	-
AM-TPC	+	+	+	+	+	+	-	+

Table 2: Some role transitons from Chinese PropBank. "+" means reasonable role transition and "-" means illegal.

5 Experiments

5.1 Experimental Setting

To evaluate the performance of our approach, we have conducted on two standard benchmarks: Chinese PropBank and English PropBank. The experimental setting is as follows:

Chinese:

We use Chinese Proposition Bank 1.0. All data are divided into three parts. 648 files (from chtb_081.fid to chtb_899.fid) are used as the training set. 40 files (from chtb_041.fid to chtb_080.fid) constitutes the development set. The test set consists of 72 files (chtb_001.fid to chtb_040.fid and chtb_900.fid to chtb_931.fid). This data setting is the same as in (Xue, 2008; Sun et al., 2009). We adopt Berkeley Parser[1] to carry out auto parsing for SRL and the parser is retrained on the training set. We used n =10 joint assignments for training the joint model and testing.

English:

We choose English Propbank as the evaluation corpus. According to the traditional partition, the training set consists of the annotations in Sections 2 to 21, the development set is Section 24, and the test set is Section 23. This data setting is the same as in (Xue and Palmer, 2004; Toutanova et al., 2005). We adopt Charniak Parser[2] to carry out auto parsing for SRL and the parser is retrained on the training set. We used n =10 joint assignments for training the joint model and testing.

5.2 Experiment on Argument Identification

We first investigate the performance of our approach in Argument Identification.

For the task of Argument Identification (AI), we adopt auto parser to produce auto parsing trees for SRL. The results are shown in Table 3. We can see that in the experiment of Chinese, the F1 score reaches to 78.79% with base features. While after additional predicates-related features are added, the precision has improved by 1.6 points with slight loss in recall, which leads to the improvement of 0.6 points in F1. The similar effect occurred in the experiment of English. After additional features added in the identification module, the precision is improved by about 0.9 points with a slight loss in recall, leading to an improvement of 0.3 points in F1. However, the improvement in English is slight smaller than in Chinese. We think the main reason is that there are less parse errors in English than in Chinese. All results demonstrate that the novel predicted-related features are effective in recognizing many identification errors which are difficult to discriminate with base features.

		P(%)	R(%)	F_1(%)
Ch	Base	84.36	73.90	78.79
	+Additional	85.97	73.72	79.38*
En	Base	82.86	76.83	79.73
	+Additional	83.75	76.69	80.06

Table 3: Comparison with Base Features in Argument Identification. Scores marked by "*" are significantly better ($p < 0.05$) than base features.

5.3 Experiment on Argument Classification

5.3.1 Results

Errors produced in AI will influenced the evaluation of Argument Classification (AC). So, to evaluate fairly we assume that the argument constituents of a predicate are already known, and the

[1] http://code.google.com/p/berkeleyparser/

[2] https://github.com/BLLIP/bllip-parser

		Num	Acc(%)
Ch	Shared	2060	91.36
	All	8462	92.77
En	Shared	2015	93.85
	All	14061	92.30

Table 4: Performance of the Base Model in Argument Classification

	Methods	Acc(%)
Ch	Base	91.36
	Joint	**93.74***
En	Base	93.85
	Joint	**95.33***

Table 5: Comparison with Base Model on shared arguments. Scores marked by "*" are significantly better ($p < 0.05$) than base model.

task is only to assign the correct labels to the constituents. The evaluation criterion is Accuracy.

The results of the base model are shown in Table 4. We first note that in testing set, there are a large number of shared arguments, which weighs about one quarter of all arguments in Chinese and 14% in English. Therefore, the fine processing of these arguments is essential for argument classification. However, the base model cannot handle these shared arguments so well in Chinese that the accuracy of the shared arguments is lower by about 1.4 points than the average value of all arguments. Nevertheless, from Table 5 we can see that our joint model's accuracy on the shared arguments reaches 93.74%, 2.4 points higher than the base model in Chinese. Although the base model obtain good performance on shared arguments of English, our joint model's performance reaches 95.33%, 1.5 points higher than the base model. This indicates that even though the base model is optimized to utilize a large set of features and achieves the state-of-the-art performance, it is still advantageous to model the joint information of the shared arguments.

Another point to note is that our joint model in resolving English SRL task is not so good as in Chinese SRL. There are mainly two reasons. The first reason is that the shared arguments occur less in English than in Chinese so that training samples are insufficient for our discriminative model. The second reason is the annotation of some intransitive verbs. In English PropBank, there is a class of intransitive verbs such as "land" (known as verbs of variable behavior), for which the argument can be tagged as either ARG0 or ARG1. Here, we take examples from the guideline[3] of English PropBank to explain.

"A bullet (ARG1) landed at his feet"

"He (ARG0) landed"

In the above examples, the two arguments and the predicate 'land' have the same relative order and voice but the arguments have different labels for their respective predicates. In fact, according to the intention of the annotator, ARG0 and ARG1 are both correct. Unfortunately, in English PropBank, there is only one gold label for each argument, which leads to much noise for our joint model. Moreover, such situations are not rare in the corpus.

5.3.2 Feature Performance

We investigate effects of the features of joint model to the performance and results are shown in Table 6. Each row shows the improvement over the baseline when that feature is used in the joint model. We can see that features proposed are beneficial to the performance of the joint model. But some features like 'RPS' and 'RPSRO' play a more important role.

Features	Chinese	English
base	91.36	93.85
RT	91.70	94.10
RPS	92.30	94.70
RPSRO	92.24	94.50
RPTS	91.80	94.18
RHWS	91.63	93.95
RHWPS	91.43	94.23
TCL	91.93	94.23
All	**93.74**	**95.33**

Table 6: Features performance in the Joint Model. We use first letter of words to represent features.

5.4 SRL Results

We also conducted the complete experiment on the auto parse trees. The results are shown in Table 7. In experiments on Chinese PropBank, we can see that after novel predicate-related features are added in the stage of Argument Identification, our model outperforms the base model by 0.5 points

[3] http://verbs.colorado.edu/propbank/EPB-AnnotationGuidelines.pdf

		$F_1(\%)$
Chinese	Base	74.04
	Base + AI	74.50
	Base + AI + AC	75.31
English	Base	76.44
	Base + AI	76.70
	Base + AI + AC	77.00

Table 7: Results on auto parse trees. Base means the baseline system, +AI meaning predcates-related features added in AI, + AC meaning joint module added.

	Methods	$F_1(\%)$
Chinese	Xue(2008)	71.90
	Sun et al.(2009)	74.12
	Ours	75.31
English	Surdeanu and Turmo(2005)	76.46
	Ours	77.00

Table 8: Comparison with Other Methods

in F1. Furthermore, after incorporating the joint module, the performance goes up to 75.31%, 1.3 points higher than the base model. We obtain similar observations in experiments on English PropBank, but due to reasons illustrated in Subsection 5.3, the performance of our method is slight better than the base model.

We compare our method with others and the results are shown in Table 8. In Chinese, Xue (2008) and Sun et al. (2009) are pioneer works in Chinese SRL. Our approach outperforms these approaches by about 3.4 and 1.9 F1 points respectively. In English SRL, we compare out method with Surdeanu and Turmo (2005) which is best result obtained with single parse tree as the input in CONLL 2005 SRL evaluation. Our approach is better than their approach which ignores the relation of multiple predicates' SRL.

6 Discussion and Analysis

In this section, we discuss some case studies that illustrate the advantages of our model. Some examples from our experiments are shown in Table 9. In example (1), the argument is a prepositional phrase '在 普及 九 年 义务 教育 的 同时' (at the same time of compulsory education) and shared by two predicates '得到' (witness) and '扩大' (expand). In the corpus, a prepositional phrase is commonly labeled as ARGM-LOC and ARGM-TMP. Thus, the base model labeled the argument into these classes but one as ARGM-LOC, another as ARGM-TMP. Unfortunately, ARGM-LOC for '得到' (witness) is wrong while our joint model outputs both correct answers, which benefits from the role transition feature. From Table 1, we can see that the role transition between ARGM-TMP and ARGM-LOC is impossible, which lowers the score of candidates containing both ARGM-LOC and ARGM-TMP in the joint model. Thus, the joint model is more likely to output the gold candidate.

In example (2), the argument is '海拉尔 飞机场' (Hailar Airport) and shared by two predicates '扩建' (expand) and '成为' (become). Because of the high similarity of the features in the base model, the argument for both predicates is classified into the same class ARG0, but the label for '扩建' (expand) is wrong. Nevertheless, our joint model obtains both correct labels, which benefits from the global features. After searching the training data, we find some similar examples to this one, such as '地铁 运行 里程 已 扩建 至 1 2 0 公里' (The railway operation mileage is expanded to 120 kilometers), in which '地铁 运行 里程' (the railway operation mileage) is labeled as ARG1 for '扩建' (expand) but ARG0 for '至' (to). We think these samples provide evidence for our joint model while these information has not been captured by the base model.

In example (3), the argument is '国内外 知名度很 高 的 大 集团' (a large group with high reputation) and shared by predicates '发展' (develop) and '为' (become). Different from the above cases in which only one label is wrong in the base model, both labels for '发展' (develop) and '为' (become) are misclassified by the base model. However, our method still gets correct answers for both predicates, which also benefits from the global features.

7 Related work

Our work is related to semantic role labeling and discriminative reranking. In this section, we briefly review these two types of work.

On Semantic Role Labeling

Gildea and Jurafsky (2002) first presented a system based on a statistical classifier which is trained on a hand-annotated corpora FrameNet. In their pioneering work, they used a gold or autoparsed syntax tree as the input and then extracted various lexical and syntactic features to identify the

Examples	Base	Ours
1. **在 普及 九 年 义务 教育 的 同时**，中等 职业 教育 也 **得到**[1]发展规模，不断**扩大**[2] (**At the same time of compulsory education**, secondary vocational education have **achieved**[1] significant development and constant **expanding**[2])	ARGM-LOC | 得到 ARGM-TMP | 扩大	ARGM-TMP | 得到 ARGM-TMP | 扩大
2. **海拉尔飞机场 扩建**[1] **成为**[2] 国际航空港 (**Hailar Airport** had been **expanded**[1] and **became**[2] an international airport)	ARG0 | 扩建 ARG0 | 成为	ARG1 | 扩建 ARG0 | 成为
3. 海尔 集团 的 销售 收入 已 突破 六十亿元, **发展**[1] **为**[2] **国内外知名度 很 高 的 大 集团** (Haier Group's sales revenue has exceeded six billion yuan and it has **developed**[1] to **be**[2] **a large group with high reputation**)	ARG1 | 发展 ARG0 | 为	ARG3 | 发展 ARG1 | 为

Table 9: Some examples in our experiments

semantic roles for a given predicate. After Gildea and Jurafsky (2002), there have been a large number of works on automatic semantic role labeling. Based on a basic discriminative model, Punyakanok et al. (2004) constructed an integer linear programming architecture, in which the dependency relations among arguments are implied in the constraint conditions. Toutanova et al. (2008) proposed a joint model to explore relations of all arguments of the same predicate. Unlike them, this paper focus on mining relations of different predicates' semantic roles in one sentence. And, there have been many extensions in machine learning models (Moschitti et al., 2008), feature engineering (Xue and Palmer, 2004), and inference procedures (Toutanova et al., 2005; Punyakanok et al., 2008; Zhuang and Zong, 2010a; Zhuang and Zong, 2010b).

Sun and Jurafsky (2004) did the preliminary work on Chinese SRL without employing any large semantically annotated corpus of Chinese. They just labeled the predicate-argument structures of ten specified verbs to a small collection of Chinese sentences, and utilized Support Vector Machine to identify and classify the arguments. They made the first attempt on Chinese SRL and produced promising results. After the PropBank (Xue and Palmer, 2003) was built, Xue and Palmer (2004) and Xue (2008) took a critical look at features of argument detection and argument classification. Unlike others' using syntax trees as the input of SRL, Sun et al. (2009) performed Chinese semantic role labeling with shallow parsing. Li et al. (2010) explored joint syntactic and semantic parsing of Chinese to further improve the performance of both syntactic parsing and SRL.

However, to the best of our knowledge, in the literatures, there is no work related to multi-predicate semantic role labeling.

On Discriminative Reranking

Discriminative reranking is a common approach in the NLP community. Its general procedure is that a base system first generates n-best candidates and with the help of global features, we obtain better performance through reranking the n-best candidates. It has been shown to be effective for various natural language processing tasks,such as syntactic parsing (Collins, 2000; Collins, 2002b; Collins and Koo, 2005; Charniak and Johnson, 2005; Huang, 2008), semantic parsing (Lu et al., 2008; Ge and Mooney, 2006), part-of-speech tagging (Collins, 2002a), named entity recognition (Collins, 2002c), machine translation (Shen et al., 2004) and surface realization in generation (Konstas and Lapata, 2012).

8 Conclusion and Feature Work

This paper investigates the interaction effect among multi-predicate's SRL. Our investigation has shown that there is much interaction effect of multi-predicate's SRL both in Argument Identification and in Argument Classification. In the stage of argument identification, we proposed novel features related to predicates and successfully removed many argument identification errors. In the stage of argument classification, we concentrate on the classification of the arguments shared by multiple predicates. Experiments have shown

that the base model often fails in classifying the shared arguments. To perform the classification of the shared arguments, we propose a joint model and with the help of the global features, our joint model yields better performance than the base model. To the best of our knowledge, this is the first work of investigating the interaction effect of multi-predicate's SRL.

In the future, we will explore more effective features for multi-predicate's identification and classification. Since we adopt reranking approach in the shared arguments' classification, the performance is limited by n-best list. Also, we would like to explore whether there is another method to resolve the problem.

Acknowledgments

We thank the three anonymous reviewers for their helpful comments and suggestions. We would also like to thank Nianwen Xue for help in baseline system, and Tao Zhuang, Feifei Zhai, Lu Xiang and Junjie Li for insightful discussions. The research work has been partially funded by the Natural Science Foundation of China under Grant No.61333018 and the Hi-Tech Research and Development Program ("863" Program) of China under Grant No.2012AA011101, and also the High New Technology Research and Development Program of Xinjiang Uyghur Autonomous Region under Grant No.201312103 as well.

References

Collin F. Baker, Charles j. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of COLING-ACL 1998*.

Janara Christensen, Mausam, Stephen Soderland and Oren Etzioni. 2010. Semantic Role Labeling for Open Information Extraction. In *Proceedings of ACL 2010*.

Eugene Charniak and Mark Johnson. 2005. Coarse tofine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL 1998*.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25 ‑ 69.

Michael Collins. 2000. Discriminative reranking fornatural language parsing. In *Proceedings of ICML 2000*.

Michael Collins. 2002a. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP 2002*.

Michael Collins. 2002b. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL 2002*.

Michael Collins. 2002c. Ranking algorithms fornamed-entity extraction: Boosting and the voted perceptron. In *Proceedings of ACL 2002*.

Ruifang Ge and Raymond J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proceedings of COLING/ACL 2002*.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling for semantic roles. In *Computational Linguistics*, 28(3): 245-288.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *ACL 2002*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL 2008*.

Sung Dong Kim, Byoung-Tak Zhang and Yung Taek Kim. 2000. Reducing Parsing Complexity by IntraSentence Segmentation based on Maximum Entropy Model. In *Proceedings of SIGDAT 2000*.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to PropBank. In *Proceedings of LREC 2002*.

Junhui Li, Guodong Zhou and Hwee Tou Ng. 2010. Joint Syntactic and Semantic Parsing of Chinese. In *Proceedings of ACL 2010*.

Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of COLING 2010*.

Junhui Li, Guodong Zhou and Hwee Tou Ng. 2010. Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of EMNLP 2008*.

Alessandro Moschitti, Daniel Pighin and Roberto Basili. 2008. Tree Kernels for Semantic Role Labeling. In *Computational Linguistics*, 34(2): 193-224.

Srini Narayanan and Sanda Harabagiu. 2004. Question Answering based on Semantic Structures. In *Proceedings of COLING 2004*.

Vasin Punyakanok, Dan Roth, Wen-tau Yih and Dav Zimak. 2004. Semantic Role Labeling via Integer Linear Programming Inference. In *Proceedings of COLING 2004*.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *Proceedings of HLT/NAACL 2004*.

Mihai Surdeanu, Sanda Harabagiu, John Williams and Paul Aarseth. 2003. Using Predicate-Argument Structures for Information Extraction. In *Proceedings of ACL 2003*.

Mihai Surdeanu and Jordi Turmo. 2005. Semantic Role Labeling Using Complete Syntactic Analysis. In *Proceedings of CONLL 2005*.

Weiwei Sun, Zhifang Sui, Meng Wang and Xin Wang. 2009. Chinese Semantic Role Labeling with Shallow Parsing. In *Proceedings of ACL 2009*.

Ioannis Konstas and Mirella Lapata. 2012. Concept-totext generation via discriminative reranking. In *Proceedings of ACL 2012*.

Kristina Toutanova, Aria Haghighi and Christopher D.Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL 2005*.

Kristina Toutanova, Aria Haghighi and Christopher D.Manning. 2008. A Global Joint Model for Semantic Role Labeling. In *Computational Linguistics*, 34(2): 161-191.

Dekai Wu and Pascale Fung. 2009. Can semantic role labeling improve smt. In *Proceedings of EAMT 2009*.

Deyi Xiong, Min Zhang and Haizhou Li. 2012. Modeling the Translation of Predicate-Argument Structure for SMT. In *Proceedings of ACL 2012*.

Nianwen Xue and Martha Palmer. 2003. Annotating the Propositions in the Penn Chinese Treebank. In *Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing.*.

Nianwen Xue and Martha Palmer. 2004. Calibrating Features for Semantic Role Labeling. In *Proceedings of EMNLP 2004*.

Nianwen Xue. 2008. Labeling Chinese Predicates with Semantic Roles. In *Computational Linguistics*, 34(2): 225-255.

Feifei Zhai, Jiajun Zhang, Yu Zhou and Chengqing Zong. 2012. Machine Translation by Modeling Predicate-Argument Structure Transformation. In *Proceedings of COLING 2012*.

Tao Zhuang and Chengqing Zong. 2010a. A Minimum Error Weighting Combination Strategy for Chinese Semantic Role Labeling. In *Proceedings of COLING 2010*.

Tao Zhuang and Chengqing Zong. 2010b. Joint Inference for Bilingual Semantic Role Labeling. In *Proceedings of EMNLP 2010*.

Werdy: Recognition and Disambiguation of Verbs and Verb Phrases with Syntactic and Semantic Pruning

Luciano Del Corro **Rainer Gemulla** **Gerhard Weikum**
Max-Planck-Institut für Informatik
Saarbrücken, Germany
{ delcorro, rgemulla, weikum }@mpi-inf.mpg.de

Abstract

Word-sense recognition and disambiguation (WERD) is the task of identifying word phrases and their senses in natural language text. Though it is well understood how to disambiguate noun phrases, this task is much less studied for verbs and verbal phrases. We present Werdy, a framework for WERD with particular focus on verbs and verbal phrases. Our framework first identifies multi-word expressions based on the syntactic structure of the sentence; this allows us to recognize both contiguous and non-contiguous phrases. We then generate a list of candidate senses for each word or phrase, using novel syntactic and semantic pruning techniques. We also construct and leverage a new resource of pairs of senses for verbs and their object arguments. Finally, we feed the so-obtained candidate senses into standard word-sense disambiguation (WSD) methods, and boost their precision and recall. Our experiments indicate that Werdy significantly increases the performance of existing WSD methods.

1 Introduction

Understanding the semantics of words and multi-word expressions in natural language text is an important task for automatic knowledge acquisition. It serves as a fundamental building block in a wide area of applications, including semantic parsing, question answering, paraphrasing, knowledge base construction, etc. In this paper, we study the task of word-sense recognition and disambiguation (WERD) with a focus on verbs and verbal phrases. Verbs are the central element in a sentence, and the key to understand the relations between sets of entities expressed in a sentence.

We propose Werdy, a method to (i) automatically recognize in natural language text both single words and multi-word phrases that match entries in a lexical knowledge base (KB) like WordNet (Fellbaum, 1998), and (ii) disambiguate these words or phrases by identifying their senses in the KB. WordNet is a comprehensive lexical resource for word-sense disambiguation (WSD), covering nouns, verbs, adjectives, adverbs, and many multi-word expressions. In the following, the notion of an *entry* refers to a word or phrase in the KB, whereas a *sense* denotes the lexical synset of the entry's meaning in the given sentence.

A key challenge for recognizing KB entries in natural language text is that entries often consist of multiple words. In WordNet-3.0 more than 40% of the entries are multi-word. Such entries are challenging to recognize accurately for two main reasons: First, the components of multi-word entries in the KB (such as *fiscal year*) often consist of components that are themselves KB entries (*fiscal* and *year*). Second, multi-word entries (such as *take a breath*) may not appear consecutively in a sentence ("He takes a deep breath."). Werdy addresses the latter problem by (conceptually) matching the syntactic structure of the KB entries to the syntactic structure of the input sentence. To address the former problem, Werdy identifies all possible entries in a sentence and passes them to the disambiguation phase (*take*, *breath*, *take a breath*, ...); the disambiguation phase provides more information about which multi-word entries to keep. Thus, our method solves the recognition and the disambiguation tasks jointly.

Once KB entries have been identified, Werdy

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 374–385,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

disambiguates each entry against its possible senses. State-of-the-art methods for WSD (Navigli, 2009) work fairly well for nouns and noun phrases. However, the disambiguation of verbs and verbal phrases has received much less attention in the literature.

WSD methods can be roughly categorized into (i) methods that are based on supervised training over sense-annotated corpora (e.g., Zhong and Ng (2010)), and (ii) methods that harness KB's to assess the semantic relatedness among word senses for mapping entries to senses (e.g., Ponzetto and Navigli (2010)). For these methods, mapping verbs to senses is a difficult task since verbs tend to have more senses than nouns. In WordNet, including monosemous words, there are on average 1.24 senses per noun and 2.17 per verb.

To disambiguate verbs and verbal phrases, Werdy proceeds in multiple steps. First, Werdy obtains the set of candidate senses for each recognized entry from the KB. Second, it reduces the set of candidate entries using novel syntactic and semantic pruning techniques. The key insight behind our syntactic pruning is that each verb sense tends to occur in a only limited number of *syntactic patterns*. For example, the sentence "Albert Einstein remained in Princeton" has a subject ("Albert Einstein"), a verb ("remained") and an adverbial ("in Princeton"), it follows an SVA clause pattern. We can thus safely prune verb senses that do not match the syntactic structure of the sentence. Moreover, each verb sense is compatible with only a limited number of *semantic argument types* (such as *location*, *river*, *person*, *musician*, etc); this phenomena is called *selectional preference* or *selectional restriction*. Senses that are compatible only with argument types not present in the sentence can be pruned. Our pruning steps are based on the idea that a verb selects the categories of its arguments both syntactically (c-selection) and semantically (s-selection). In the final step, Werdy employs a state-of-the-art general WSD method to select the most suitable sense from the remaining candidates. Since incorrect senses have already been greatly pruned, this step significantly gains accuracy and efficiency over standard WSD.

Our semantic pruning technique builds on a newly created resource of pairs of senses for verbs and their object arguments. For example, the WordNet verb sense ⟨*play*-1⟩ (i.e., the 1st sense of the verb entry "play") selects as direct object the noun sense ⟨*sport*-1⟩. We refer to this novel resource as the *VO Sense Repository*, or VOS repository for short.[1] It is constructed from the WordNet gloss-tags corpus, the SemCor dataset, and a small set of manually created VO sense pairs.

We evaluated Werdy on the SemEval-2007 coarse-grained WSD task (Navigli et al., 2007), both with and without automatic recognition of entries. We found that our techniques boost state-of-the-art WSD methods and obtain high-quality results. Werdy significantly increases the precision and recall of the best performing baselines.

The rest of the paper is organized as follows. Section 2 gives an overview of Werdy components. Section 3 presents the entry recognition, and Sections 4 and 5 discuss our novel syntactic and semantic pruning techniques. Section 6 presents the Semantic VO Repository and how we constructed it. Section 7 gives the results of our evaluation. Section 8 discusses related work.

2 Overview of Werdy

Werdy consists of four steps: (i) entry recognition, (ii) syntactic pruning, (iii) semantic pruning, and (iv) word-sense disambiguation. The novel contribution of this paper is in the first three steps, and in the construction of the VO sense repository. Each of these steps operates on the clause level, i.e., we first determine the set of clauses present in the input sentence and then process clauses separately. A *clause* is a part of a sentence that expresses some statement or coherent piece of information. Clauses are thus suitable minimal units for automatic text understanding tasks (Del Corro and Gemulla, 2013); see Sec.3 for details.

In the *entry-recognition step* (Sec. 3), Werdy obtains for the input sentence a set of potential KB entries along with their part-of-speech tags. The candidate senses of each entry are obtained from WordNet. For instance, in the sentence "He takes a deep and long breath", the set of potential entries includes *take* (verb, 44 candidate senses), *take a breath* (verb, 1 candidate sense), and *breath* (noun, 5 candidate senses). Note that in contrast to Werdy, most existing word-sense disambiguation methods assume that entries have already been (correctly) identified.

[1] The VOS repository, Werdy's source code, and results of our experimental study are available at `http://people.mpi-inf.mpg.de/~corrogg/`.

In the *syntactic-pruning step* (Sec. 4), we eliminate candidate senses that do not agree with the syntactic structure of the clause. It is well-established that the syntactic realization of a clause is intrinsically related with the sense of its verb (Quirk et al., 1985; Levin, 1993; Hanks, 1996; Baker et al., 1998; Palmer et al., 2005). Quirk et al. (1985) identified seven possible clause types in the English language (such as "subject verb adverbial", SVA). We make use of techniques inspired by Del Corro and Gemulla (2013) to identify the clause type of each clause in the sentence. We then match the clause type with the set of WordNet frames (e.g., "somebody verb something") that WordNet provides for each verb sense, and prune verb senses for which there is no match.

In the *semantic-pruning step* (Sec. 5), we further prune the set of candidate senses by taking the semantic types of direct objects into account. Similarly to the syntactic relation mentioned above, a verb sense also imposes a (selectional) restriction on the semantic type of its arguments (Quirk et al., 1985; Levin, 1993; Hanks, 1996; Baker et al., 1998; Palmer et al., 2005). For instance, the verb *play* with sense *participate in games or sports* requires an object argument of type ⟨*game*-1⟩[2], ⟨*game*-3⟩, or ⟨*sport*-1⟩. Senses that do not match the arguments found in the clause are pruned. This step is based on the newly constructed VOS Repository (Sec. 6). Note that when there is no direct object, only the syntactic pruning step applies.

3 Entry Recognition

The key challenge in recognizing lexical KB entries in text is that entries are not restricted to single words. In addition to named entities (such as people, places, etc.), KB's contain multi-word expressions. For example, WordNet-3.0 contains entries such as *take place* (verb), *let down* (verb), *take into account* (verb), *be born* (verb), *high school* (noun), *fiscal year* (noun), and *Prime Minister* (noun). Note that each individual word in a multi-word entry is usually also an entry by itself, and can even be part of several multi-word entries. To ensure correct disambiguation, all potential multi-word entries need to be recognized (Finlayson and Kulkarni, 2011), even when they do not appear as consecutive words in a sentence.

Werdy addresses these challenges by exploring the syntactic structure of both the input sentence and the lexical KB entries. The structure of the sentence is captured in a dependency parse (DP). Given a word in a sentence, Werdy conceptually generates all subtrees of the DP starting at that word, and matches them against the KB. This process can be performed efficiently as WordNet entries are short and can be indexed appropriately. To match the individual words of a sentence against the words of a KB entry, we follow the standard approach and perform lemmatization and stemming (Finlayson, 2014). To further handle personal pronouns and possessives, we follow Arranz et al. (2005) and normalize personal pronouns (I, you, my, your, ...) to *one's*, and reflexive pronouns (myself, yourself, ...) to *oneself*.

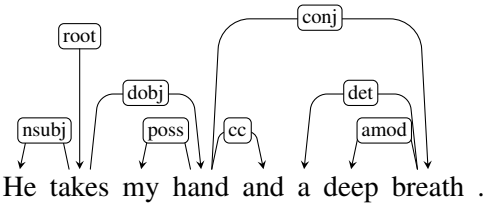


Figure 1: An example dependency parse

Consider the example sentence "He takes my hand and a deep breath". We first identify the clauses and their DP's (Fig. 1) using the method of Del Corro and Gemulla (2013), which also processes coordinating conjunctions. We obtain clauses "He takes my hand" and "He takes a deep breath", which we process separately. To obtain possible entries for the first clause, we start with its head word (*take*) and incrementally consider its descendants (*take hand*, *take one's hand*, ...). The exploration is terminated as early as possible; for example, we do not consider *take one's hand* because there is no WordNet entry that contains both *take* and *hand*. For the second clause, we start with *take* (found in WordNet), then expand to *take breath* (not found but can occur together), then *take a breath* (found), then *take a deep breath* (not found, cannot occur together) and so on.

Note that the word "take" in the sentence refer to two different entries and senses: "take" for the first clause and "take a breath" for the second clause. In this stage no decisions are made about selecting entries and disambiguating them; these decisions are made in the final WSD stage of Werdy.

We tested Werdy's entry-recognizer on the SemEval-2007 corpus. We detected the correct en-

[2]We use the notation ⟨*WordNet entry*-sense number⟩.

Pattern	Clause type	Example	WN frame example [frame number]
SV_i	SV	AE died.	Somebody verb [2]
SV_eA	SVA	AE remained in Princeton.	Somebody verb PP [22]
SV_cC	SVC	AE is smart.	Somebody verb adjective [6]
$SV_{mt}O$	SVO	AE has won the Nobel Prize.	Somebody verb something [8]
$SV_{dt}O_iO$	SVOO	RSAS gave AE the Nobel Prize.	Somebody verb somebody something [14]
$SV_{ct}OA$	SVOA	The doorman showed AE to his office.	Somebody verb somebody PP [20]
$SV_{ct}OC$	SVOC	AE declared the meeting open.	Something verb something adjective/noun [5]

S: Subject, V: Verb, C: Complement, O: Direct object, O_i: Indirect object, A: Adverbial, V_i: Intransitive verb, V_c: Copular verb, V_c: Extended-copular verb, V_{mt}: Monotransitive verb, V_{dt}: Ditransitive verb, V_{ct}: Complex-transitive verb

Table 1: Clause types and examples of matching WordNet frames

tries for all but two verbs (out of more than 400). The two missed entries (*take up* and *get rolling*) resulted from incorrect dependency parses.

4 Syntactic Pruning

Once the KB entries have been recognized, Werdy prunes the set of possible senses of each verb entry by considering the syntactic structure of the clause in which the entry occurs. This pruning is based on the observation that each verb sense may occur only in a limited number of "clause types", each having specific semantic functions (Quirk et al., 1985). When the clause type of the sentence is incompatible with a candidate sense of an entry, this sense is eliminated.

Werdy first detects in the input sentence the set of clauses and their constituents. A clause consists of one *subject* (S), one *verb* (V), and optionally an *indirect object* (O), a *direct object* (O), a *complement* (C) and one or more *adverbials* (A). Not all combinations of clause constituents appear in the English language. When we classify clauses according to the grammatical function of their constituents, we obtain only seven different clause types (Quirk et al., 1985); see Table 1. For example, the sentence "He takes my hand" is of type SVO; here "He" is the subject, "takes" the verb, and "my hand" the object. The clause type can (in principle) be determined by observing the verb type and its complementation (Del Corro and Gemulla, 2013).

For instance, consider the SVA clause "The student remained in Princeton". The verb *remain* has four senses in WN: (1) stay the same; remain in a certain state (e.g., "The dress remained wet"), (2) continue in a place, position, or situation ("He remained dean for another year"), (3) be left; of persons, questions, problems ("There remains the question of who pulled the trigger") or (4) stay behind ("The hostility remained long after they made up"). The first sense of *remain* requires an SVC pattern; the other cases require either SV or SVA. Our example clause is of type SVA so that we can safely prune the first sense.

WordNet provides an important resource for obtaining the set of clause types that are compatible with each sense of a verb. In particular, each verb sense in WordNet is annotated with a set of *frames* (e.g., "somebody verb something") in which they may occur, capturing both syntactic and semantic constraints. There are 35 different frames in total.[3] We manually assigned a set of clause types to each frame (e.g., SVO to frame "somebody verb something"). Table 1 shows an example frame for each of the seven clause types. On average, each WordNet-3.0 verb sense is associated with 1.57 frames; the maximum number of frames per sense is 9. The distribution of frames is highly skewed: More than 61% of the 21,649 frame annotations belong to one of four simple SVO frames (numbers 8, 9, 10 and 11), and 22 out of the 35 frames have less than 100 instances. This skew makes the syntactic pruning step effective for non-SVO clauses, but less effective for SVO clauses.

Werdy directly determines a set of possible frame types for each clause of the input sentence. Our approach is based on the clause-type detection method of Del Corro and Gemulla (2013), but we also consider additional information that is captured in frames but not in clause types. For example, we distinguish different realizations of objects (such as clausal objects from non-clausal objects), which are not captured in the clause type. Given the DP of a clause, Werdy identifies the

[3] See `http://wordnet.princeton.edu/wordnet/man/wninput.5WN.html`.

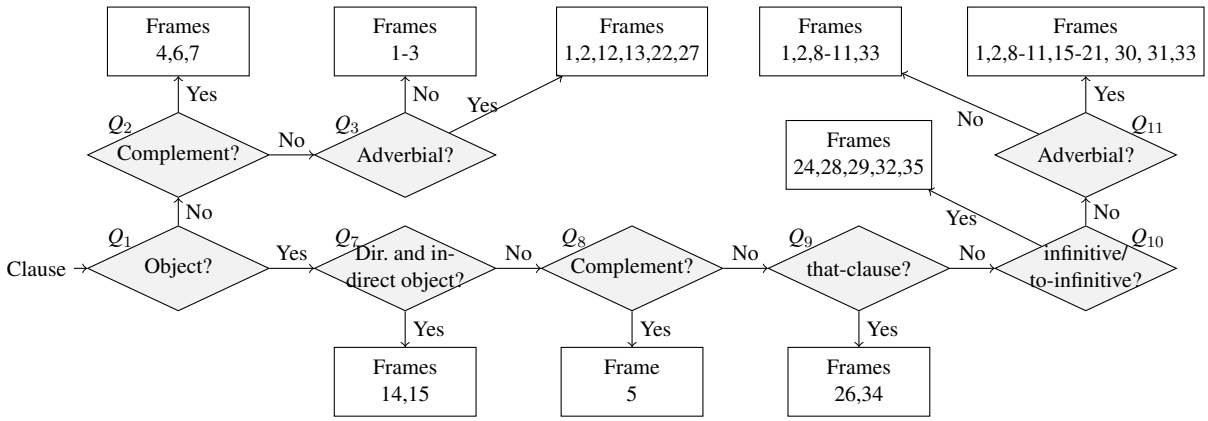


Figure 2: Flow chart for frame detection

set of WN frames that can potentially match the clause as outlined in the flowchart of Fig. 2. Werdy walks through the flowchart; for each question, we check for the presence or absence of a specific constituent of a clause (e.g., a direct object for Q_1) and proceed appropriately until we obtain a set of possible frames. This set is further reduced by considering additional information in the frames (not shown; e.g., that the verb must end on "-ing"). For our example clause "The student remained in Princeton", we first identify possible frames $\{ 1, 2, 12, 13, 22, 27 \}$ using the flowchart (Q_1 no, Q_2 no, Q_3 yes); using the additional information in the frames, Werdy then further prunes this set to $\{ 1, 2, 22 \}$. The corresponding set of remaining candidate sense for *remain* is as given above, i.e., $\{ \langle$*remain*-2$\rangle, \langle$*remain*-3$\rangle, \langle$*remain*-4$\rangle \}$.

Our mapping of clause types to WordNet frames is judiciously designed for the way WordNet is organized. For instance, frames containing adverbials generally do not specify whether or not the adverbial is obligatory; here we are conservative in that we do not prune such frames if the input clause does not contain an adverbial. As another example, some frames overlap or subsume each other; e.g, frame "somebody verb something" (8) subsumes "somebody verb that clause" (26). In some word senses annotated with the more general frame, the more specific one can also apply (e.g., $\langle$*point out*-1$\rangle$ is annotated with 8 but not 26; 26 can apply), in others it does not (e.g., $\langle$*play*-1$\rangle$ is also annotated with 8 but not 26; but here 26 cannot apply). To ensure the effectiveness of syntactic pruning, we only consider the frames that are directly specified in WordNet. This procedure often produces the desired results; in a few cases, however, we do prune the correct sense (e.g., frame 26 for clause "He points out that . . . ").

5 Semantic Pruning

A verb sense imposes a restriction on the semantic type of the arguments it may take and vice versa (Quirk et al., 1985; Levin, 1993; Hanks, 1996; Baker et al., 1998; Palmer et al., 2005; Kipper et al., 2008). This allows us to further prune the verb candidate set by discarding verb senses whose semantic argument is not present in the clause.

WordNet frames potentially allow a shallow type pruning based on the semantics provided for the clause constituents. However we could solely distinguish people ("somebody") from things ("something"), which is too crude to obtain substantial pruning effects. Moreover, this distinction is sometimes ambiguous.

Instead, we have developed a more powerful approach to semantic pruning based on our VOS repository. We remove from the verb candidate set those senses whose semantic argument cannot be present in the sentence. For instance, consider the clause "The man plays football." Suppose that we know that the verb entry *play* with sense $\langle$*play*-1$\rangle$ ("participate in sports") takes an object of type $\langle$*sport*-1$\rangle$; i.e., we have a tuple $\langle$*play*-1, *sport*-1$\rangle$ in our repository. Then, we check whether any of the possible senses of *football*—(i) *sport* or (ii) *ball*—is of type $\langle$*sport*-1$\rangle$. Here the first sense has the correct type (the second sense does not); thus we retain $\langle$*play*-1$\rangle$ as a possible sense for the verb entry *play*. Next, suppose that we consider sense $\langle$*play*-3$\rangle$ ("play on an instrument"), which according to our corpus takes $\langle$*instrument*-6$\rangle$ as argument (i.e., there is a tuple $\langle$*play*-3, *instrument*-6$\rangle$ in our VOS repository). Since none of the senses of *football* is of type $\langle$*instrument*-6$\rangle$, we can safely drop

⟨*play*-3⟩ from our candidate set. We perform this procedure for every verb sense in the candidate set.

Semantic pruning makes use of both VOS repository and the hypernym structure of the noun senses in WordNet. For each sentence, we obtain the possible senses of the direct-object argument of the verb. We then consider each candidate sense of the verb (e.g., ⟨*play*-1⟩), and check whether any of its compatible object-argument senses (from our repository) is a hypernym of any of the possible senses of its actual object argument (in the sentence); e.g., ⟨*sport*-1⟩ is a hypernym of ⟨*football*-1⟩. If so, we retain the verb's candidate sense. If not, either the candidate sense of the verb is indeed incompatible with the object argument in the sentence, or our repository is incomplete. To handle incompleteness to some extent, we also consider hyponyms of the object-argument senses in our repository; e.g., if we observe object *sport* in a sentence and have verb-sense argument ⟨*football*-1⟩ in our corpus, we consider this a match. If the hyponyms lead to a match, we retain the verb's candidate sense; otherwise, we discard it.

6 Verb-Object Sense Repository

We use three different methods to construct the repository. In particular, we harness the sense-annotated WordNet glosses[4] as well as the sense-annotated SemCor corpus (Landes et al., 1998).[5]

The major part of the VOS repository was acquired from WordNet's gloss tags. According to Atkins and Rundell (2008), noun definitions should be expressed in terms of the class to which they belong, and verb definitions should refer to the types of the subjects or objects related to the action. Based on this rationale, we extracted all noun senses that appear in the gloss of each verb sense; each of these noun senses is treated as a possible sense of the object argument of the corresponding verb sense. For example, the gloss of ⟨*play*-1⟩ is "participate in games or sports;" each noun is annotated with its senses (2 and 3 for "games", 1 for "sports"). We extract tuples ⟨*play*-1, *game*-2⟩, ⟨*play*-1, *game*-3⟩, and ⟨*play*-1, *sport*-1⟩ from this gloss. Note that we only extract direct-object arguments, i.e., we do not consider the type of the subject argument of a verb sense. Since the constituents of the predicate are much more important than the subject to determine or describe a verb sense, lexical resources rarely contain information on the subject (Atkins and Rundell, 2008). Similarly, WordNet glosses typically do not provide any information about adverbials. Overall, we collected arguments for 8,657 verb senses (out of WordNet's 13,767 verb senses) and a total of 13,050 ⟨*verb*-#, *object*-#⟩-pairs.

We leveraged the sense-annotated SemCor corpus to further extend our VOS repository. We parsed each sentence in the corpus to obtain the respective pairs of verb sense and object sense. Since sentences are often more specific than glosses, and thus less helpful for constructing our repository, we generalized the so-found object senses using a heuristic method. In particular, we first obtained all the object senses of each verb sense, and then repeatedly generalized sets of *at least two senses* that share a direct hypernym to this hypernym. The rationale is that we only want to generalize if we have some evidence that a more general sense may apply; we thus require at least two hyponyms before we generalize. Using this method, we collected arguments for 1,516 verb senses and a total of 4,131 sense pairs.

Finally, we noticed that the most frequent senses used in the English language are usually so general that their glosses do not contain any relevant semantic argument. For instance, one of the most frequent verbs is ⟨*see*-1⟩, which has gloss "perceive by ⟨*sight*-3⟩". The correct semantic argument ⟨*entity*-1⟩ is so general that it is omitted from the gloss. In fact, our gloss-tag extractor generates tuple ⟨*see*-1, *sight*-3⟩, which is incorrect. We thus manually annotated the 30 most frequent verb senses with their object argument types.

Our final repository contains arguments for 9,335 verb senses and a total of 17,181 pairs. Pairs from SemCor tend to be more specific because they refer to text occurrences. The assumption of taking the nouns of the glosses as arguments seems to be mostly correct, although some errors may be introduced. Consider the pair ⟨*play*-28, *stream*-2⟩ extracted from the gloss "discharge or direct or be discharged or directed as if in a continuous ⟨*stream*-2⟩". Also, in some cases, the glosses may refer to adverbials as in ⟨*play*-14, *location*-1⟩, taken from gloss "perform on a certain ⟨*location*-1⟩". Note that if an argument is missing from our repository, we may prune the correct sense of the verb. If, however, there is an additional, incorrect

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

[5] http://web.eecs.umich.edu/~mihalcea/downloads.html

argument in the repository, the correct verb sense is retained but pruning may be less effective.

7 Evaluation

Dataset. We tested Werdy on the SemEval-2007 coarse-grained dataset.[6] It consists of five sense-annotated documents; the sense annotations refer to a coarse-grained version of WordNet. In addition to sense annotations, the corpus also provides the corresponding KB entries (henceforth termed "gold entries") as well as a POS tag. We restrict our evaluation to verbs that act as clause heads. In total, 461 such verbs were recognized by ClausIE (Del Corro and Gemulla, 2013) and the Stanford Parser (Klein and Manning, 2003).[7]

WSD Algorithms. For the final step of Werdy, we used the KB-based WSD algorithms of Ponzetto and Navigli (2010) and It-Makes-Sense (Zhong and Ng, 2010), a state-of-the-art supervised system that was the best performer in SemEval-2007. Each method only labels entries for which it is sufficiently confident.

Simplified Extended Lesk (SimpleExtLesk). A version of Lesk (1986). Each entry is assigned the sense with highest term overlap between the entry's context (words in the sentence) and both the sense's gloss (Kilgarriff and Rosenzweig, 2000) as well as the glosses of its neighbors (Banerjee and Pedersen, 2003). A sense is output only if the overlap exceeds some threshold; we used thresholds in the range of 1–20 in our experiments. There are many subtleties and details in the implementation of SimpleExtLesk so we used two different libraries: a Java implementation of WordNet::Similarity (Pedersen et al., 2004),[8] which we modified to accept a context string, and DKPro-WSD (Miller et al., 2013) version 1.1.0, with lemmatization, removal of stop words, paired overlap enabled and normalization disabled.

Degree Centrality. Proposed by Navigli and Lapata (2010). The method collects all paths connecting each candidate sense of an entry to the set of candidate senses of the words the entry's context. The candidate sense with the highest degree in the resulting subgraph is selected. We implemented this algorithm using the Neo4j library.[9] We used a fixed threshold of 1 and vary the search depth in range 1–20. We used the candidate senses of all nouns and verbs in a sentence as context.

It-Makes-Sense (IMS). A state-of-the-art, publicly available supervised system (Zhong and Ng, 2010) and a refined version of Chan et al. (2007), which ranked first in the SemEval-2007 coarse grained task. We modified the code to accept KB entries and their candidate senses. We tested both in WordNet-2.1 and 3.0; for the later we mapped Werdy's set of candidates to WordNet-2.1.

Most Frequent Sense (MFS). Selects the most frequent sense (according to WordNet frequencies) among the set of candidate senses of an entry. If there is a tie, we do not label. Note that this procedure differs slightly from the standard of picking the entry with the smallest sense id. We do not follow this approach since it cannot handle well overlapping entries.

MFS back-off. When one of the above methods fails to provide a sense label (or provides more than one), we used the MFS method above with a threshold of 1. This procedure increased the performance in all cases.

Methodology. The disambiguation was performed with respect to coarse-grained sense clusters. The score of a cluster is the sum of the individual scores of its senses (except for IMS which provides only one answer per word); the cluster with the highest score was selected. Our source code and the results of our evaluation are publicly available[10].

The SemEval-2007 task was not designed for automatic entry recognition, for each word or multi-word expression it provides the WordNet entry and the POS tag. We proceeded as follows to handle multi-word entries. In the WSD step, we considered the candidate senses of all recognized entries that overlap with the gold entry. For example, we considered the candidate senses of entries *take*, *breath*, and *take a breath* for gold entry *take a breath*.

The SemEval-2007 task uses WordNet-2.1 but Werdy uses WordNet-3.0. We mapped both the sense keys and clusters from WordNet-2.1 to WordNet-3.0. All senses in WordNet-3.0 that could not be mapped to any cluster were consider to belong each of them to a single sense cluster. Note that this procedure is fair: for such senses

[6] The data is annotated with WordNet 2.1 senses; we converted the annotations to WordNet-3.0 using DKPro-WSD (Miller et al., 2013).

[7] Version 3.3.1, model englishRNN.ser.gz

[8] http://www.sussex.ac.uk/Users/drh21/

[9] http://www.neo4j.org/

[10] http://people.mpi-inf.mpg.de/~corrogg/

Algorithm	Gold Entry	Pruning	MFS back-off	threshold /depth	Verbs (clause heads) P	R	F1	F1 points
Degree	+	-	+	5	73.54	73.54	73.54	
Centrality	+	+	+	11	79.61	79.61	79.61	+ 6.07
	+	-	-	5	73.99	71.58	72.77	
	+	+	-	8	79.91	78.52	79.21	+ 6.44
	-	-	+	5	70.41	70.41	70.41	
	-	+	+	10	76.46	76.46	76.46	+ 6.05
	-	-	-	4	71.05	68.90	69.96	
	-	+	-	10	76.81	75.81	76.30	+ 6.34
SimpleExtLesk	+	-	+	6	77.28	75.27	76.26	
(DKPro)	+	+	+	5	81.90	80.48	**81.18**	+ 4.92
	+	-	-	1	73.70	52.28	61.17	
	+	+	-	1	81.99	64.21	72.02	**+ 10.85**
	-	-	+	5	74.33	72.57	73.44	
	-	+	+	5	79.30	77.75	**78.52**	+ 5.08
	-	-	-	1	69.85	50.54	58.65	
	-	+	-	1	78.69	62.20	69.48	+ 10.83
SimpleExtLesk	+	-	+	5	77.11	75.27	76.18	
(WordNet::Sim)	+	+	+	5	80.57	79.18	79.87	+ 3.69
	+	-	-	1	74.82	68.98	71.78	
	+	+	-	1	79.04	75.27	77.11	+ 5.33
	-	-	+	6	74.12	72.35	73.22	
	-	+	+	7	77.97	76.46	77.21	+ 3.99
	-	-	-	1	71.36	65.66	68.39	
	-	+	-	1	76.20	71.92	74.00	+ 5.61
MFS	+	-	-	1	76.61	74.62	75.60	
	+	+	-	1	80.35	78.96	79.65	+ 4.05
	-	-	-	1	73.67	71.92	72.79	
	-	+	-	1	77.75	76.24	76.99	+ 4.20
IMS	+	-	+	n.a.	79.60	79.60	79.60	
(WordNet-2.1)	+	+	+	n.a.	80.04	80.04	80.04	+ 0.44
	-	-	+	n.a.	76.21	75.05	75.63	
	-	+	+	n.a.	77.53	76.36	76.94	+ 1.31
IMS	+	-	+	n.a.	78.96	78.96	78.96	
(WordNet-3.0)	+	+	+	n.a.	79.83	79.83	79.83	+ 0.87
	-	-	+	n.a.	75.77	74.62	75.19	
	-	+	+	n.a.	77.53	76.36	76.94	+ 1.75

Table 2: Results on SemEval-2007 coarse-grained (verbs as clause heads)

the disambiguation is equivalent to a fine-grained disambiguation, which is harder.

Results. Our results are displayed in Table 2. We ran each algorithm with the gold KB entries provided by in the dataset (+ in column "gold entry) as well as the entries obtained by our method of Sec. 3 (-). We also enabled (+) and disabled (-) the pruning steps as well as the MFS back-off strategy. The highest F1 score was achieved by SimpleExtLesk (DKPro) with pruning and MFS back-off: 81.18 with gold entries and 78.52 with automatic entry recognition. In all cases, our syntactic and semantic pruning strategy increased performance (up to +10.85 F1 points). We next discuss the impact of the various steps of Werdy in detail.

Detailed Analysis. Table 3 displays step-by-step results for DKPro's SimpleExtLesk, for MFS, as well as SimpleExtLesk with MFS back-off, the best performing strategy. The table shows results when only some Werdy's steps are used. We start from a direct use of the respective algorithm with the gold entries of SemEval-2007 after each horizontal line, and then successively add the Werdy steps indicated in the table.

When no gold entries were provided, performance dropped due to the increase of sense candidates for multi-word expressions, which include the possible senses of the expression itself as well as the senses of the entry's parts that are them-

Steps Performed	threshold	P	R	F1	F1 points
SimpleExtLesk (DKPro)					
Plain with gold entries	**1**	**73.70**	**52.28**	**61.17**	
+ Entry Recognition	1	69.85	50.54	58.65	- 2.52
+ Syntactic Pruning	1	76.47	58.84	66.50	**+ 7.85**
+ Semantic Pruning	1	78.69	62.20	69.48	+ 2.98
+ Entry Recognition	1	69.85	50.54	58.65	- 2.52
+ Semantic Pruning	1	73.85	55.39	63.30	+ 4.65
+ Syntactic Pruning	1	79.33	61.21	69.10	**+ 7.93**
+ Semantic Pruning	1	81.99	64.21	72.02	+ 2.92
+ Semantic Pruning	1	78.11	56.90	65.84	+ 4.67
MFS					
Plain with gold entries	1	76.61	74.62	75.60	
+ Entry Recognition	1	73.67	71.92	72.79	- 2.81
+ Syntactic Pruning	1	75.77	74.14	74.95	+ 2.16
+ Semantic Pruning	1	77.75	76.24	76.99	+ 2.04
+ Entry Recognition	1	73.67	71.92	72.79	- 2.81
+ Semantic Pruning	1	77.09	75.43	76.25	**+ 3.46**
+ Syntactic Pruning	1	78.46	76.94	77.69	+ 2.09
+ Semantic Pruning	1	80.35	78.96	79.65	+ 1.96
+ Semantic Pruning	1	79.91	78.02	78.95	**+ 3.35**
SimpleExtLesk (DKPro) with MFS back-off					
Plain with gold entries	6	77.28	75.27	76.26	
+ Entry Recognition	6	74.33	72.57	73.44	- 2.82
+ Syntactic Pruning	5	76.65	75.00	75.82	+ 2.38
+ Semantic Pruning	5	79.30	77.75	78.52	+ 2.70
+ Entry Recognition	5	74.33	72.57	73.44	- 2.82
+ Semantic Pruning	5	78.19	76.51	77.34	**+3.90**
+ Syntactic Pruning	5	79.34	77.80	78.56	+ 2.30
+ Semantic Pruning	5	81.90	80.48	81.18	+ 2.62
+ Semantic Pruning	5	81.02	79.09	80.04	**+ 3.78**

Table 3: Step-by-step results

selves WordNet entries. Our entry recognizer tends to do a good job since it managed to correctly identify all the relevant entries except in two cases (i.e. "take up" and "get rolling"), in which the dependency parse was incorrect. The drop in F1 for our automatic entry recognition was mainly due to incorrect selection of the correct entry of a set of alternative, overlapping entries.

Syntactic pruning did not prune the correct sense in most cases. In 16 cases (with gold entries), however, the correct sense was pruned. Five of these senses were pruned due to incorrect dependency parses, which led to incorrect frame identification. In two cases, the sense was not annotated with the recognized frame in WordNet, although it seemed adequate. In the remaining cases, a general frame from WordNet was incorrectly omitted. Improvements to WordNet's frame annotations may thus make syntactic pruning even more effective.

Semantic pruning also improves performance. Here the correct sense was pruned for 11 verbs, mainly due to the noisiness and incompleteness of our VOS repository. Without using gold entries, we found in total 237 semantic matches between possible verbs senses and possible object senses (200 with gold entries). We also found that our manual annotations in the VOS repository (see Sec. 6) did not affect our experiments.

The results show that syntactic and semantic pruning are beneficial for verb sense disambiguation, but also stress the necessity to improve existing resources. Ideally, each verb sense would be annotated with both the possible clause types or syntactic patterns in which it can occur as well as the possible senses of its objects. Annotations for subjects and adverbial arguments may also be beneficial.

8 Related Work

WSD is a classification task where for every word there is a set of possible senses given by some external resource (as a KB). Two types of methods can be distinguished in WSD. Supervised systems (Dang and Palmer, 2005; Dligach and Palmer, 2008; Chen and Palmer, 2009; Zhong and Ng, 2010) use a classifier to assign senses to words, mostly relying on manually annotated data for training. In principle, these systems suffer from low coverage since the training data is usually sparse. Some authors have tried to overcome this limitation by exploiting linked resources as training data (Shen et al., 2013; Cholakov et al., 2014).

The second WSD approach corresponds to the so-called KB methods (Agirre and Soroa, 2009; Ponzetto and Navigli, 2010; Miller et al., 2012; Agirre et al., 2014). They rely on a background KB (typically WordNet or extended versions (Navigli and Ponzetto, 2012)), where related senses appear close to each other. KB-based algorithms often differ in the way the KB is explored. It has been shown that a key point to enhance performance is the amount of semantic information in the KB (Ponzetto and Navigli, 2010; Miller et al., 2012). Our framework fits this line of work since it is also unsupervised and enriches the background knowledge in order to enhance performance of standard WSD algorithms. A comprehensive overview of WSD systems can be found in Navigli (2009) and Navigli (2012).

To bring WSD to real-world applications, the mapping between text and KB entries is a fundamental first step. It has been pointed that the existence of multi-word expressions imposes multiple challenges to text understanding tasks (Sag et al., 2002). The problem has been addressed by Arranz et al. (2005) and Finlayson and Kulkarni (2011). They find multi-word entries by matching word sequences allowing some morphological and POS variations according to predefined patterns. Our method differs in that we can recognize KB entries that appear discontinuously and in that we do not select the correct entry but generate a set of potential entries.

Linguists have noted the link between verb senses and the syntactic structure and argument types (Quirk et al., 1985; Levin, 1993; Hanks, 1996), and supervised WSD systems were developed to capture this relation (Dang and Palmer, 2005; Chen and Palmer, 2009; Dligach and Palmer, 2008; Cholakov et al., 2014). In Dang and Palmer (2005) and Chen and Palmer (2009), it is shown that WSD tasks can be improved with features that capture the syntactic structure and information about verb arguments and their types. They use features as shallow named entity recognition and the hypernyms of the possible senses of the noun arguments. Dang and Palmer (2005) also included features extracted from PropBank (Palmer et al., 2005) from role labels and frames. Dligach and Palmer (2008) generated a corpus of verb and their arguments (both surface forms), which was used to incorporate a semantic feature to the supervised system.

In our work, we also incorporate syntactic and semantic information. Instead of learning the relation between the verb senses and the syntactic structure, however we incorporate it explicitly using the WordNet frames, which provide information about which verb sense should be consider for a given syntactic pattern. We also incorporate explicitly the semantic relation between each verb sense and its arguments using our VOS repository.

Different resources of semantic arguments for automatic text understanding tasks have been constructed (Baker et al., 1998; Palmer et al., 2005; Kipper et al., 2008; Gurevych et al., 2012; Nakashole et al., 2012; Flati and Navigli, 2013). In (Baker et al., 1998; Palmer et al., 2005; Kipper et al., 2008; Gurevych et al., 2012), the classification of verbs and arguments is focused toward semantic or thematic roles. Nakashole et al. (2012) uses semantic types to construct a taxonomy of binary relations and Flati and Navigli (2013) collected semantic arguments for given textual expressions. For instance, given the verb "break", they extract a pattern "break ⟨*body part*-1⟩". In contrast to existing resources, our VOS repository disambiguates both the verb sense and the senses of its arguments.

9 Conclusion

We presented Werdy, a framework for word-sense recognition and disambiguation with a particular focus on verbs and verbal phrases. Our results indicate that incorporating syntactic and semantic constraints improves the performance of verb sense disambiguation methods. This stresses the necessity of extending and improving the available syntactic and semantic resources, such as WordNet or our VOS repository.

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *Proceedings of EACL*, pages 33–41.

Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

Victoria Arranz, Jordi Atserias, and Mauro Castillo. 2005. Multiwords and word sense disambiguation. In *Computational Linguistics and Intelligent Text Processing*, volume 3406 of *Lecture Notes in Computer Science*, pages 250–262.

B. T. Sue Atkins and Michael Rundell. 2008. *The Oxford Guide to Practical Lexicography*. Oxford University Press.

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

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of IJCAI*, pages 805–810.

Yee Seng Chan, Hwee Tou Ng, and Zhi Zhong. 2007. Nus-pt: Exploiting parallel texts for word sense disambiguation in the english all-words tasks. In *Proceedings of SemEval*, pages 253–256.

Jinying Chen and Martha Palmer. 2009. Improving english verb sense disambiguation performance with linguistically motivated features and clear sense distinction boundaries. *Language Resources and Evaluation*, 43(2):181–208.

Kostadin Cholakov, Judith Eckle-Kohler, and Iryna Gurevych. 2014. Automated verb sense labelling based on linked lexical resources. In *Proceedings of EACL*, pages 68–77.

Hoa Trang Dang and Martha Palmer. 2005. The role of semantic roles in disambiguating verb senses. In *Proceedings of ACL*, pages 42–49.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *Proceedings of WWW*, pages 355–366.

Dmitriy Dligach and Martha Palmer. 2008. Improving verb sense disambiguation with automatically retrieved semantic knowledge. In *Proceedings of ICSC*, pages 182–189.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Mark Alan Finlayson and Nidhi Kulkarni. 2011. Detecting multi-word expressions improves word sense disambiguation. In *Proceedings of MWE*, pages 20–24.

Mark Alan Finlayson. 2014. Java libraries for accessing the princeton wordnet: Comparison and evaluation. In *Proceedings of GWC*.

Tiziano Flati and Roberto Navigli. 2013. Spred: Large-scale harvesting of semantic predicates. In *Proceedings of ACL*, pages 1222–1232.

Iryna Gurevych, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M. Meyer, and Christian Wirth. 2012. Uby - a large-scale unified lexical-semantic resource based on lmf. In *Proceedings of EACL*, pages 580–590.

Patrick Hanks. 1996. Contextual dependency and lexical sets. *International Journal of Corpus Linguistics*, 1(1):75–98.

Adam Kilgarriff and Joseph Rosenzweig. 2000. Framework and results for english senseval. *Computers and the Humanities*, 34(1-2):15–48.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of English verbs. *Language Resources and Evaluation*, 42(1):21–40.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*, pages 423–430.

Shari Landes, Claudia Leacock, and Randee I. Tengi, 1998. *Building Semantic Concordances*. MIT Press.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of SIGDOC*, pages 24–26.

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press.

Tristan Miller, Chris Biemann, Torsten Zesch, and Iryna Gurevych. 2012. Using distributional similarity for lexical expansion in knowledge-based word sense disambiguation. In *Proceedings of COLING*, pages 1781–1796.

Tristan Miller, Nicolai Erbs, Hans-Peter Zorn, Torsten Zesch, and Iryna Gurevych. 2013. Dkpro wsd: A generalized uima-based framework for word sense disambiguation. In *Proceedings of ACL: System Demonstrations*, pages 37–42.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: A taxonomy of relational patterns with semantic types. In *Proceedings of EMNLP*, pages 1135–1145.

Roberto Navigli and Mirella Lapata. 2010. An experimental study of graph connectivity for unsupervised word sense disambiguation. *EEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):678–692.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193(0):217–250.

Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves. 2007. Semeval-2007 task 07: Coarse-grained english all-words task. In *Proceedings of SemEval*, pages 30–35.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2):10:1–10:69.

Roberto Navigli. 2012. A quick tour of word sense disambiguation, induction and related approaches. In *Proceedings of SOFSEM*, pages 115–129.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity: Measuring the relatedness of concepts. In *Proceedings of HLT-NAACL: Demonstration Papers*, pages 38–41.

Simone Paolo Ponzetto and Roberto Navigli. 2010. Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proceedings of ACL*, pages 1522–1531.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann A. Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *Proceedings of CICLing*, pages 1–15.

Hui Shen, Razvan Bunescu, and Rada Mihalcea. 2013. Coarse to fine grained sense disambiguation in wikipedia. In *Proceedings of *SEM*, pages 22–31.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of ACL: System Demonstrations*, pages 78–83.

Multi-Resolution Language Grounding with Weak Supervision

R. Koncel-Kedziorski, Hannaneh Hajishirzi, and Ali Farhadi
University of Washington
{kedzior,hannaneh,farhadi}@washington.edu

Abstract

Language is given meaning through its correspondence with a world representation. This correspondence can be at multiple levels of granularity or *resolutions*. In this paper, we introduce an approach to multi-resolution language grounding in the extremely challenging domain of professional soccer commentaries. We define and optimize a factored objective function that allows us to leverage discourse structure and the compositional nature of both language and game events. We show that finer resolution grounding helps coarser resolution grounding, and vice versa. Our method results in an F1 improvement of more than 48% versus the previous state of the art for fine-resolution grounding[1].

1 Introduction

Language is inextricable from its context. A human language user interprets an utterance in the context of, among other things, their perception of the world. Grounded language acquisition algorithms imitate this setup: language is given meaning through its correspondence with a rich world representation. A solution to the acquisition problem must resolve several ambiguities: the segmentation of the text into meaningful units (spans of words that refer to events); determining which events are being referenced; and finding the proper alignment of events to these units.

Historically, language grounding was only possible over simple controlled domains and rigidly structured language. Current research in grounded language acquisition is moving into real-world environments (Yu and Siskind, 2013). Grounding sports commentaries in game events is a specific instance of this problem that has attracted attention (Liang et al., 2009; Snyder and Barzilay, 2007; Hajishirzi et al., 2012), in part because of the complexity of both the language and the world representation involved.

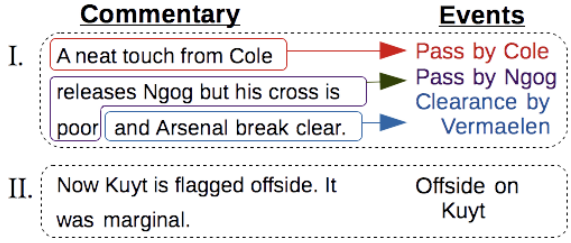


Figure 1: An example of the multiple resolutions at which soccer commentaries refer to events: The utterance level alignments are shown in the black dashed boxes. The first utterance can be further broken into the fragment-level alignments shown; the second cannot be decomposed further.

The language employed in soccer commentaries is difficult to ground due to its dense information structure, novel vocabulary and word senses, and colorful, non-traditional syntax. These challenges conspire to foil most language processing techniques including automated parsers and word-sense disambiguation systems.

In addition to the structural problems presented by the language of soccer commentaries, the problem of reference is further complicated by the fact that for game events (and other real-world phenomena) there is no standardized meaningful linguistic unit. Utterances ranging from a single word to multiple sentences can be used to refer to a single event. For example, in Figure 1 the first four words of commentary (I) refer to a single event, as does the entirety of (II).

[1]Source code and data are available at http://ssli.ee.washington.edu/tial/projects/multires/

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 386–396,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

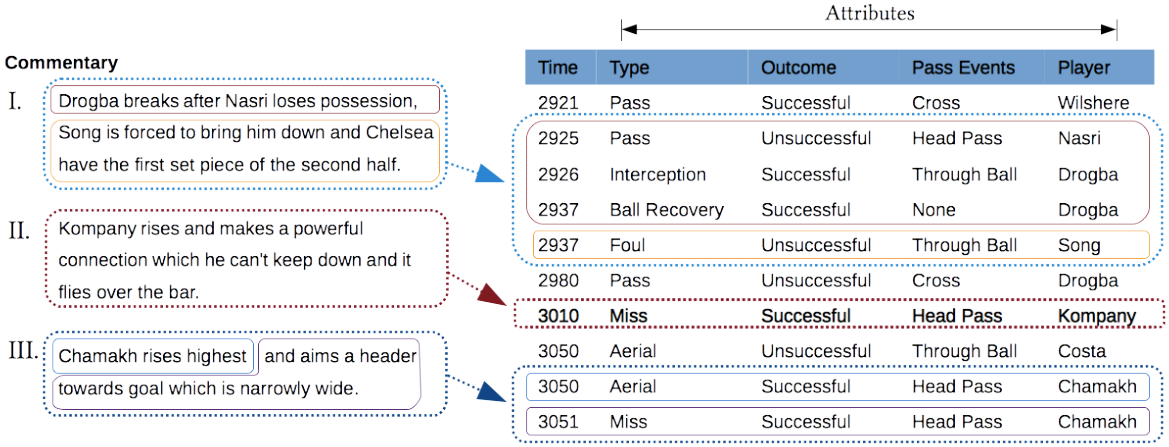


Figure 2: An example of the different levels of granularity present in the soccer data. The dashed boxes on the left denote *utterances* made by the commentators. Solid boxes denote fragments that cannot be decomposed into finer resolution alignments. The table on the right is a portion of the detailed listing of game events.

Turning our attention to Figure 2, sometimes a fragment refers to a combination of events and no further decomposition is available, such as the first fragment of commentary (I). Moreover, it is sometimes desirable to construct a complex of events by determining all the events corresponding to a particular collection of words. For instance, we would want to be able to align the whole of (I) with all the events in the corresponding dashed box. This suggests studying language grounding at multiple levels of granularity (resolutions).

We use *resolution* to describe the continuum of meaningful units which exist in human language[2]. These resolutions interact in a complicated way, with clues from different resolutions sometimes combining to produce an effect and sometimes negating one another. With enough training data, one could hope to learn the details of the interactions of various resolutions. However, the expense of producing or obtaining supervised training data at multiple resolutions is prohibitive.

To address all these complications, we introduce weakly-supervised multi-resolution language grounding. Our method makes use of a factorized objective function which allows us to model the complex interplay of resolutions. Our language model takes advantage of the discourse structure of the commentaries, making it robust enough to handle the unique language of the soccer domain. Finally, our method relies only on loose temporal co-occurrence of events and utterances as supervision and does not require expensive annotated training data.

To test our method we augment the Professional Soccer Commentary Dataset (Hajishirzi et al., 2012) with fragment-level event alignment annotations. This dataset is composed of commentaries for soccer matches paired with event logs produced by Opta Sportsdata and includes human annotated gold alignments[3]. We achieve an F1 improvement of over 48% on fragment-level alignment versus a previous state-of-the-art. We are also able to leverage the interplay of fragment- and utterance- level alignments to improve the previous state-of-the-art utterance-alignment system.

2 Challenges

Syntactic Limitations: Syntax is used to structure the information provided by an utterance, and so it seems intuitive that syntactic relations could be leveraged in this task. For example, consider utterance (III) in Figure 2. The multi-resolution grounding of (III) would provide a *segmentation* of the utterance – or a division of the utterance into the fragments which refer to separate events. In (III), there is an obvious syntactic correlate to the correct segmentation: each verb phrase within the conjunction headed by "and" identifies a separate event. Parsing (III) to an event-based semantics like that of Davidson (1967), one could associate each verb in an utterance with a game event and achieve the desired segmentation.

[2]Though it is tempting to discritize meaning in text, Chafe (1988) shows that readers imbue text with meaningful intonational patterns drawn from the potentially continuous space of auditory signals.

[3]Our updated dataset is available at `http://ssli.ee.washington.edu/tial/projects/multires/`

Unfortunately, there is a preponderance of examples such as (II) in Figure 2, where 4 verbs are used to describe a single "miss" event. (II) illustrates just one of the many difficulties of using syntactic information – elsewhere, events are referenced without an explicit verb whatsoever (such as the use of the phrase "into the books" to refer to a foul event). What is needed instead is a language model that is powerful enough to proscribe some structure yet robust enough to allow the world representation to determine which pieces of language are referring to which referent or set of referents.

Complex Interplay between Resolutions: Language refers at a variety of resolutions, and the relationship between nested reference scopes is complex. A single or few words can indicate entities or properties; full phrases are often needed to denote an action; complex events like a missed shot may take up to several phrases of narration to properly describe. A soccer commentator does not encode every detail necessary for proper alignment and segmentation into their utterances, but rather only enough to make clear to another with similar world knowledge what is meant. A language grounding method is at a severe disadvantage when faced with such implicit information.

Instead, a successful method can make heavy use of the limited lexical, phrasal, and discourse structural cues provided in an utterance, as the different resolutions rely on these different contextual clues to meaning. At finer resolutions one can rely more on the lexical meanings of the words; at medium resolutions, compositionality can be leveraged; at coarser resolutions, discourse features come into play. These cues interact in a complicated way, providing additional challenge.

Consider again Figure 2. In (III), the temporal discourse marker "and" marks the division between the fragments referring to each event. In (I) the same word (used again as a temporal discourse marker) is used to elaborate on the single "foul" event being described in the second fragment. A human (with sufficient understanding of soccer) knows that, despite being separated by the discourse marker, the phrases "bring him down" and "set piece" both refer to the foul. A language grounding algorithm that can model the interaction between such word-level and utterance-level cues can successfully segment both (I) and (III).

Supervision: For language grounding generally, and multi-resolution grounding specifically, supervised training data is expensive to produce. Also, the various grounding domains of interest are highly independent of one another (Liang et al., 2009). In the face of these issues, the ideal correspondence between language and world representation would be learned with as little supervision as possible.

3 Problem Definition

We define the problem of multi-resolution language grounding as follows: Given a temporal evolution of a world state (a sequence of events) and an overlapping natural language text (a sequence of utterances), we want to learn the best correspondences between the language and the world at different levels of granularity (Figure 2).

To set up notations, for each utterance represented as a set of words $W = \{w_1, w_2, \ldots, w_n\}$, we want a segmentation which expresses the relationship of the words to the events which they describe.

Let $\mathcal{S}$ denote a set of all possible segmentations of W. Then $\mathcal{S} = \{S | S \text{ is a segmentation of } W\}$. A segmentation S is in turn a set of non-overlapping fragments ($S = \{s_i\}$), where each fragment is a consecutive sequence of words from the utterance W. For example, for utterance (III) from Figure 2, one possible (incorrect) segmentation is $S = \{s_1, s_2, s_3\}$ for $s_1 = \{$*Chamakh rises highest*$\}$, $s_2 = \{$*and aims a header*$\}$, and $s_3 = \{$*towards goal which is narrowly wide*$\}$.

An alignment consists of a segmentation S and a mapping E from fragments of S to the set of all events $\mathbb{E}$. For example, the segmentation S could be mapped as $E = \{\langle s_1, e_2\rangle, \langle s_2, e_3\rangle, \langle s_3, e_1\rangle\}$, with e_1 being an Aerial Challenge, e_2 being a missed attempt on goal, and e_3 being an out of bounds penalty. Let $\mathcal{E} = \mathcal{S} \times \mathbb{E}$ denote the set of all possible alignments.

As we show in Figure 2, events are composed of the various attributes *Time, Type, Pass Events, Outcome*, and *Player*. For example, the aerial event in Figure 2 has the attributes and values *type:aerial, outcome:successful, pass events:head pass,* and *player:Chamakh*.

Finally, we denote the values for the attributes of each e_j as e_j^a, where a ranges over the different attributes of events as represented in the data.

We define the multi-resolution grounding of W

into $\mathbb{E}$ as the best segmentation S and alignment E that maximize the joint probability distribution:

$$\arg\max_{S \in \mathcal{S}, E \in \mathcal{E}} P(S, E|W) \quad (1)$$

This optimization[4] can be accomplished through the use of supervised learning. However, training data is expensive and tedious to produce for the grounding problem, especially at multiple resolutions. Additionally, the complexity of the language in this domain would result in very sparse associations.

Yet if we knew some of the correct fine-resolution alignments, we could use that information to produce good coarse resolution alignments, and vice versa. Therefore, we formulate a factorized form of the above objective which allows us to learn features specific to aligning at the utterance, fragment, and attribute resolutions. Our method can be optimized with only weak supervision (loose temporal alignments between utterances and a set of events occurring within a window of the utterance time).

We can evaluate such a correspondence in several ways. For each utterance, can we predict the correct events to which this utterance refers? This is the problem of utterance-level alignment.

We can also evaluate based on events: for each event, can we identify the minimal text span(s) which refers to this event? We want a tight correspondence because loose, overlapping alignments are not semantically satisfying. However, we do not want to under associate: human language makes reference at a variety of levels (the word level, the phrase level, the utterance level, and beyond). It is important to correctly identify all and only the words which correspond to a given event. This is the fragment-level alignment problem. We show that good fragment-level alignments will improve utterance-level alignment, and vice versa.

Since events are composed of their attributes, we can imagine a very fine resolution grounding of individual words to individual attributes. In fact, our solution involves producing such a grounding and composing the fragment- and utterance-level alignments therefrom.

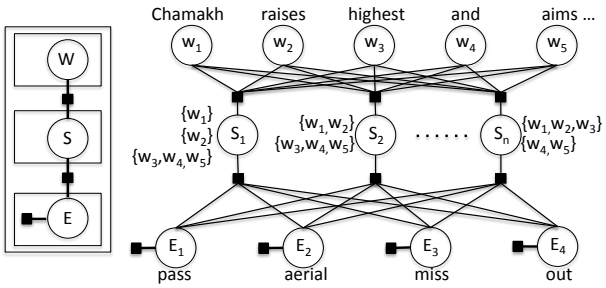


Figure 3: Factor graph for $P(S, E|W)$. Here the w_i are the words of utterance W, S_j are the possible segmentations of W, and E_k are different events.

4 Our Method

We have formulated the grounding problem as an optimization of the joint probability distribution $P(S, E|W)$, which returns the best segmentation and accompanying event alignments given an utterance W. Optimizing this function in the domain of real world language, however, is a difficult problem. Utterances are long here, and there are many events which could be grounded to each. Furthermore, the cardinality of the set of possible segmentations is combinatorially large.

Therefore we decompose Equation 1 using the factor graph depicted in Figure 3. We write the joint probability distribution as a product of the following two potential functions:

$$P(S, E|W) \stackrel{\text{def}}{=} \frac{1}{Z} \prod_{s \in S} \Psi^{\text{align}}(E, s) * \Phi^{\text{seg}}(s, W) \quad (2)$$

where Ψ^{align} is a function for scoring the alignment E for fragment s and Φ^{seg} scores how good a fragment s is for the utterance W, and Z is for normalization.

To optimize Equation 2 it is not practical to search the space of possible S, E combinations (this space is combinatorially large). However, we can optimize the factored form using dynamic programming. We first describe how to find values for each of the potentials in sections 4.1 and 4.2. In section 4.3 we describe the dynamic programming approach to optimization.

4.1 Event Alignments Given Segmentation

The potential function $\Psi^{\text{align}}(E, s)$ takes as inputs a fragment s from segmentation S and a candidate alignment E for S and returns a score for E with

[4] As this and future equations are conditioned on the set of all events $\mathbb{E}$, we omit this variable from the equations for notational simplicity.

regards to s. It is here that we produce the multi-resolution alignments; s can vary in size from a single word to a whole utterance. ψ^{align} decomposes as the following:

$$\Psi^{\text{align}}(E, s) = \Psi^{\text{prior}}(E) * \Psi^{\text{affinity}}(s, E) \quad (3)$$

where the priors (Ψ^{prior}) are confidence scores for an alignment E with the whole utterance as given by Hajishirzi et al. (2012), which fits an exemplar SVM to each utterance/event pair. An exemplar SVM is an SVM fit with one positive and many negative instances, allowing us to define an example by what it is not (Malisiewicz et al., 2011; Shrivastava et al., 2011).

Ψ^{affinity} scores the affinity between a fragment s and the event e_j to which it is aligned. We use the term affinity as a measure of the goodness of an alignment. Intuitively, a fragment s will have a higher affinity for an event e_j if s describes that event well. Formally, the affinity between s and e_j amounts to a product of the affinity between each word $w_i \in s$ and e_j. Since e_j is defined by a collection of attributes, we can compose a score for w_i with e_j from the affinity between w_i and each attribute a of e_j.

$$\begin{aligned}\Psi^{\text{affinity}}(s, E) &= \prod_{w_i \in s, e_j \in E} \psi^{\text{atr.}}(w_i, e_j) \\ &= \prod_{w_i \in s, e_j \in E} \max_a \psi(w_i, e_j^a) \quad (4)\end{aligned}$$

where e_j is the event to which s is aligned in alignment E, $\psi^{\text{atr.}}(w_i, e_j)$ is the affinity between w_i and event e_j, and $\psi(w_i, e_j^a)$ is the affinity between w_i and attribute a of e_j.

In order to determine the affinity of a word and an event attribute, we create *attribute:value classifiers* – one for each *attribute:value pair* that occurs in any event. For example, for goals we create a *type:goal* classifier, and for unsuccessful events we create an *outcome:unsuccessful* classifier.

For the categorical attributes *Type, Outcome,* and *Pass Events*, we fit a linear SVM (Fan et al., 2008) using the utterance-level alignments provided by Ψ^{prior} (the exemplar SVMs) to determine the positive and negative examples. For instance, we use all the utterances which are aligned with an event whose *type* value is "pass" as positive examples for our *type:pass* classifier, and all other utterances as negative examples.

The weight assigned to each dimension in a linear SVM describes the relative importance of that dimension in the classification process. The dimensions of our *attribute:value* SVMs are the words of the corpus, normalized for case and minus punctuation and stop words. Therefore, the affinity of a word w_i and the *attribute:value* e_j^a is the weight of the dimension corresponding to w_i in the e_j^a *attribute:value* classifier. Following others (Liang et al., 2009; Kate and Mooney, 2007), we use string matches to determine the affinity between a word and the *Player* attribute.

In order to make comparisons between the importance of a word in the decision process for different classifiers, we normalize the weight vectors for each. These *attribute:value* classifiers produce our finest resolution alignments, allowing us to define a correspondence between a single word and a single attribute of any event.

By considering e_j in terms of its attributes, we are able to compose a score for e_j with fragment s. This is a kind of double-sided compositional semantics, where both the meaningful signs (s) and their extensions (e_j) are composed of finer-resolution atomic parts (w_i and e_j^a, respectively).

4.2 Segmentations Given Utterances

The potential function $\Phi^{\text{seg}}(s, W)$ from Equation 2 returns a score for a fragment within an utterance. A segmentation can be thought of as the collection of bigrams $\langle w_i, w_{i+1} \rangle$ where w_i is the last word of a fragment which is being used to describe one event and w_{i+1} is the first word of a fragment being used to describe a different event. We will refer to such bigrams as *splitpoints*.

The function Φ^{seg} should favor fragments that begin and end at good *splitpoint* and whose intermediate bigrams are bad *splitpoints*. We formalize this as follows:

$$\Phi^{\text{seg}}(s, W) \propto \frac{\phi(w_{k-1}, w_k) * \phi(w_{k+m}, w_{k+m+1})}{\prod_{j=0}^{m-1} \phi(w_{k+j}, w_{k+j+1})}$$

where fragment s is a span of m consecutive words $\{w_k, ..., w_{k+m}\}$ from W, and ϕ is a score for how good of a *splitpoint* $\langle w_i, w_{i+1} \rangle$ would make (explained below).

Ideally, ϕ will be a classifier which can tell us if a given bigram is a good *splitpoint* for the utterance W. However, ours being an attempt at weakly-supervised learning, we have no labeled examples of correct splitpoints from which

to work. Instead, we employ linguistic knowledge to create a proxy of labels. We will use this proxy to train a classifier to discover the features of good splitpoints which can be generalized and produce a more robust system.

The proxy labeling scheme we developed is based on conservative components common to a variety of theories of discourse. Discourse theories aim to model the relationships which exist between adjacent utterances in a coherent discourse. Since we consider a sports commentary to be a coherent discourse, we can leverage results from discourse theory in producing our proxy labels.

Temporal Discourse: Events in a soccer match occur in a temporal sequence, and so it is reasonable to assume that the language used to describe them will employ temporal discourse relations to distinguish fragments describing separate events. Pitler et al. (2008) have constructed a list of discourse relations which can be easily automatically identified, including temporal discourse relations. These are indicated by the presence of *discourse markers* — alternately known as cue phrases. We hypothesize that cue phrases can be used to identify splitpoints and use them in our proxy labeling scheme. This method is not restricted to temporally related discourse: some contingency, expansion, and comparison relations are also analyzed as "easily identifiable". As such, our segmentation process can also be used to ground language into a world state where these relations would hold.

Prosodic Discourse: We also make use of prosodic discourse cues. Pierrehumbert and Hirschberg (1990) claim that intonational phrases play an important role in discourse segmentation. Therefore, we hypothesize that the edges of intonational phrases are very likely to correspond with correct splitpoints. Viewing the commentary transcriptions as a noisy channel of the actual speech signal, we can identify the intonational phrase boundaries with the punctuation inserted in the transcription process. Chafe (1988) confirms that punctuation in written language has a strong correspondence with intonational phrase boundaries, and an assumption like ours has been successfully implemented in speech synthesis systems (Black and Lenzo, 2000). Thus, we include bigrams containing punctuation as splitpoints in our proxy labels.

Feature Description for *splitpoint* classifier
Is w_i/w_{i+1} a discourse marker?
Is w_i/w_{i+1} punctuation?
Is w_i/w_{i+1} a player name?
Part of speech of w_i/w_{i+1}
Is one of w_i/w_{i+1} a dependent of the other?
Are w_i and w_{i+1} dependents of the same governor?
Dependency relations that hold across splitpoint
Height of w_i/w_{i+1} in the dependency tree
Difference in height of w_i/w_{i+1} in dependency tree
$\psi(w_i, e_j)$ of all words left versus right of splitpoint
Symmetric difference of best affinity scores for w_i/w_{i+1}
Are best affinity scores from the same event?

Table 1: Feature description for splitpoint classifier

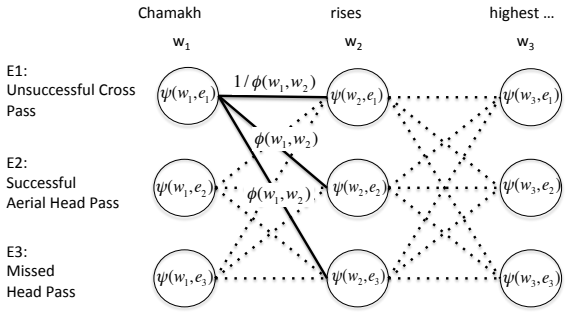


Figure 4: We use a trellis to allow for dynamic programming optimization of the objective function

Splitpoint Classifier: All other bigrams besides those above are labeled as negative examples, and a linear SVM is fit to the data. The features for the classifier include structural, discourse, and statistical features. We make use of dependency parse information from the Stanford dependency parser (De Marneffe and Manning, 2008). The full features list is explained in Table 1.

4.3 Optimization

We want to maximize the function in Equation 1, and we have explained that we can approximate this by maximizing the factored form in Equation 2. By the above methods, we can produce values for the functions Ψ^{align} and Φ^{seg}. What remains is to optimize Equation 2.

We take advantage of the factorization by using a dynamic programming approach to optimization. Figure 4 illustrates the setup. For each word w_i of the utterance, we create a column of nodes in our trellis, with one row for each event $e_j \in E$. The nodes represent the affinity of a given word w_i with event e_j. The weights on these nodes come from $\psi^{\text{atr.}}(w_i, e_j)$ described in section 4.2.

The nodes in column w_i are connected to the nodes in column w_{i+1} by edges whose weights

Method	Precision	Recall	F1
Liang et al. (2009)	0.513	0.393	0.445
Our approach	**0.603**	**0.481**	**0.535**

Table 2: Fragment-level alignments starting from gold utterance-level alignments

Method	Precision	Recall	F1
Liang et al. (2009)	0.211	0.135	0.165
Our approach	**0.235**	**0.255**	**0.245**

Table 3: Fragment-level alignments starting from raw data

are drawn from the splitpoint classifier response $\phi(w_i, w_{i+1})$. We label the edges between adjacent nodes corresponding to different events with the responses from the splitpoint classifier, and the inverse of these responses for edges connecting nodes corresponding to the same event.

We then use the Viterbi algorithm (Viterbi, 1967) to find the maximum scoring path through this trellis. The maximum scoring path optimizes Equation 2, and serves as our approximation of the optimization of Equation 1. We choose the top k diverse paths through the trellis and use the associations therein as our alignments. See Figure 5 for a detailed example of how our Viterbi path coincides with the responses from the *attribute:value* classifiers.

5 Experiments

One justification for multi-resolution language grounding would be if finer-resolution grounding improves coarser-resolution grounding and vice versa. If so, we expect that better utterance-level alignments will improve fragment-level alignments, and that in turn those fragment-level alignments will improve utterance-level alignments. We evaluate both of these hypotheses.

5.1 Experimental Setup

Dataset: We use the publicly available Professional Soccer Commentary (PSC) dataset introduced in Hajishirzi et al. (2012). This dataset is composed of professional commentaries from the 2010-2011 season of the English Premier League, along with a human-annotated data feed produced for each game by Opta Sportsdata (Opta, 2012) which describes all events occurring around the ball. Events include passes, shots, misses, cards, tackles, and other relevant game details. Each event category is defined precisely and the feed is annotated by professionals according to strict event description guidelines.

Method	Precision	Recall	F1
Liang et al. (2009)	0.327	0.418	0.367
Hajishirzi et al. (2012)	0.355	**0.576**	0.439
Our approach	**0.407**	0.520	**0.457**

Table 4: Utterance-level alignment results

The PSC also provides ground truth alignment of full utterances to events in the data feed, and for this work we have augmented it with ground truth fragment-level annotations[5].

We use data from 7 games of the PSC. These games consist of 778 utterances totaling 13,692 words. There are 12,275 events. This data is labeled with ground truth utterance- and fragment-alignments.

Metric: There are 1,295 correct utterance-to-event alignments. For evaluation we use precision, recall, and F1 of our utterance-level alignments.

The evaluation of fragment-level alignments is less straight forward. This is due to the two features of a correct fragment alignment: picking the correct fragment boundaries and associating the fragment with the correct event. We evaluate fragment-level alignment on a per word basis. We consider precision in this task to be the number of correct word to event alignments versus the total number of alignments produced by a system. Recall is the number of correct word to event alignments versus the total gold word to event alignments, of which there are 18,147.

Comparisons: We compare to two previous works: Liang et al. (2009), which produces both segmentation and alignment results; and Hajishirzi et al. (2012), which produces state-of-the-art alignments. When evaluating segmentation, we compare how well the systems perform starting from the raw dataset, and starting from gold utterance-level alignments. This allows us to isolate the segmentation process from the overall system architectures. It also gives us some insight into the effect of event priors on the segmentation and alignment processes.

[5]The full dataset is available at `http://ssli.ee.washington.edu/tial/projects/multires/`

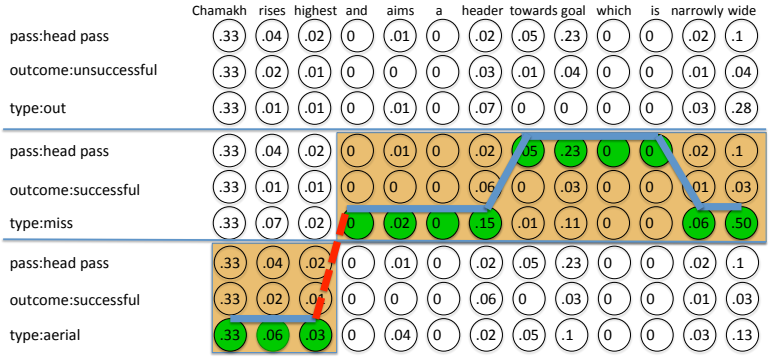


Figure 5: A successful grounding at multiple resolutions. Thin blue lines separate the *attribute:value* pairs corresponding to the three events. Values of $\psi(w_i, e_j)$ are shown on each node. The shaded bands indicate the gold fragment-level alignments. Thick line connecting the green nodes indicates the classifier responses used in the Viterbi best path through our trellis. The red dashed edge indicates a high response from the splitpoint classifier. This figure is best viewed in color.

Method	Precision	Recall	F1
Ours	0.235	0.255	0.245
- Ψ^{affinity}	0.213	0.133	0.164
- Φ^{seg}	0.205	0.232	0.218

Table 5: Ablation studies for fragment-level alignments by removing Ψ^{affinity} and Φ^{seg} from our model by replacing them with uniform function.

Method	Precision	Recall	F1
Ours	0.407	0.520	0.457
- Ψ^{affinity}	0.446	0.189	0.265
- Φ^{seg}	0.376	0.563	0.451

Table 6: Ablation studies for utterance-level alignments by removing Ψ^{affinity} and Φ^{seg} from our model by replacing them with uniform function.

5.2 Results

We evaluate our method on its alignments at the fragment-level and at the utterance-level. The results are as follows:

Fragment-level: Our results for segmentation can be seen in Tables 2 and 3. Table 2 shows the results achieved on the fragment-level alignment task using human-labeled utterance to event alignments. In this setting, all and only the correct events for each utterance are present. Still, there are several ambiguities in the data. Some fragments are aligned in the gold data with multiple events, and some are aligned to no event. Our method outperforms the previous by a large margin in terms of both precision and recall. We show below how this is due to our system's accommodation of discourse structure when making segmentation decisions and the factored form of our optimization.

Table 3 shows the results for fragment-level alignment by applying each system starting from the raw data. Here, in addition to the ambiguities mentioned above, the problem is further complicated by the fact that some correct events are missing from the alignments produced by each system and some incorrect events are included in these alignments (see Error Analysis below for details). Still our method achieves a significant improvement, with a 48% increase in F1 versus prior work.

Table 5 shows ablation results for the effect of the factors used in our optimization for fragment-level alignments. These results demonstrate the value of each factor in the fragment-level alignment process. We cannot ascribe the benefit of this method to one factor or another alone – it is their concert that improves performance.

Utterance-level: We have posited that good finer-resolution alignments will improve the coarser-resolution utterance to event alignments. Our results confirm this hypothesis. Table 4 shows our results on these alignments. We are able to improve F1 versus a state-of-the-art system which is tuned to maximize its F1 score. The majority of our improvement comes from the increased precision of our system, due to the influence of the finer-resolution fragment-level alignments on these coarser, utterance-level alignments. We provide a detailed example of this below. Ablation results are shown in Table 6.

5.3 Qualitative Analysis

A qualitative analysis of our system reveals the power of our factored objective, double-sided compositional approach, and leveraging of discourse structure. Figure 5 shows the best path through the trellis of the example sentence used in the introduction. For explanatory purposes, we have split every event into its three component attributes. This allows us to see how the *attribute:value* classifiers combine to produce an alignment.

Discourse Structure: The fragment-level alignment we have produced for this utterance is perfect: it correctly identifies the single *splitpoint* and correctly identifies each fragment with the associated event.

The identification of the splitpoint "and" comes from the fact that this word has, among other uses, a discourse connective meaning. Thus, the edges

in our trellis between different events are weighted higher than edges between the same event in the edges between the nodes for "highest" and "and", encouraging the Viterbi path to change events at this point.

Compositionality: We can see effect of the compositional approach we have taken – composing $\psi^{\text{affinity}}(s, e_j)$ from the *attribute:value* classifier scores of each $\psi(w_i, e_j^a)$ – by looking at how the best path makes use of different attributes of the same event. For the "miss" event aligned with the second part of the sentence, we can see that the best path makes use of both values from the *type:miss* and *pass event:head pass* classifiers.

Affinities: A few interesting associations are worth pointing out. First, we note that the word "header" has a stronger affinity for the *type:miss* attribute than it does for the *pass events:head pass* attribute. On first blush, this seems like a mistake in our classifier. However, we can see that even in this single trellis all three events have the *pass events:head pass* attribute. The utterance-level alignment uses this association already, aligning utterances containing the word "header" with events that have a *pass events:head pass* attribute. At a finer-resolution, it is necessary to make a different distinction between events. Our method finds that the presence of the word "header" is a stronger indicator of an event with a *type:miss* attribute, and thus this association is made.

Words that are better for the coarser-resolution association with the *pass events:head pass* attribute are "towards" and "goal". Out of the 10 utterances containing the word "towards" in the dataset, 3 of these are aligned with at least 1 *pass events:head pass* event, making this strong association a correct one. The word "goal" also has an affinity for the *pass events:head pass* attribute due to the fact that many events with this attribute are attempts on goal. This correlates with domain knowledge about soccer, because, although there may be other uses of their head by a player in the game, shots on goal are events which will nearly always be commented upon by an announcer.

Factorization: We have shown that finer-resolution fragment-level alignments can improve utterance-level alignments. From the exemplar SVMs, we are given an utterance-level alignment of the three events shown in the trellis with the utterance. This alignment is incorrect: the gold utterance alignment only includes the bottom two events. But by building an utterance-level alignment from the results of our fragment level alignment, we are left with only the two correct events. We prune the topmost event due to its failure to participate in a finer-resolution alignment.

5.4 Error Analysis

The majority of the errors made on our fragment-level alignments come in one of two flavors: Firstly, we sometimes erroneously identify a fragment as referring to an event when in truth it refers to no event. Commentators often describe facts about players or the weather or previous games which have no extension in the current game. However, our system cannot distinguish such language from the language referring to this game. This is a good avenue for future exploration.

The second set of errors we make in fragmentation are caused by bad event priors. Our current setup cannot increase recall: we can only improve the precision of the utterance-level alignments we are given. Therefore, if an event is overlooked in the first-pass of utterance-level alignments, we cannot reintroduce it through a fragment alignment. This is a direction for future work as well.

6 Related Work

Early semantic parsing work made use of fully supervised training (Zettlemoyer and Collins, 2005; Ge and Mooney, 2006; Snyder and Barzilay, 2007), but more recent work has focused on reducing the amount of supervision required (Artzi and Zettlemoyer, 2013). A few unsupervised approaches exist (Poon and Domingos, 2009; Poon, 2013), but these are specific to translating language into queries in highly structured database and cannot be applied to our more flexible domain.

There are few datasets as detailed as the Professional Soccer Commentary Dataset. Early work in understanding soccer commentaries focused on RoboCup soccer (Chen and Mooney, 2008; Chen et al., 2010; Bordes et al., 2010; Hajishirzi et al., 2011) where simple language describes each event, and events are in a one-to-one correspondence with utterances. Another dataset used for language grounding is the Weather Report Dataset (Liang et al., 2009). Here, again, however, we have mostly single utterances paired with single events, and many alignments are made via numerical string matching rather than learning lex-

ical cues. The NFL Recap dataset (Snyder and Barzilay, 2007) is also laden with numerical fact matching, and does not include the fragment-level segmentation annotation that the PSC dataset provides.

Impressive advances have been made grounding language in instructions. Branavan et al. (2009) and Vogel and Jurafsky (2010) work in the domain of computer technical support instructions, mapping language to actions using reinforcement learning. Matuszek et al. (2012b) parses simple language to robot control instructions. Our work focuses on dealing with a richer space, both in terms of the language used and the world-representation into which it is grounded, and leveraging the multiple resolutions of reference.

An exciting direction of research, closer to our own, aims to ground natural language in visual perception systems. Matuszek et al. (2012a) attempts to learn a joint model of language and object characteristics of a workplace environment. Yu and Siskind (2013) grounds moderately rich language in automatically annotated video clips. Again, the contribution of our work versus the above is in the complexity of the language with which we deal and our multi-resolution model.

7 Conclusion

The problem of grounding complex natural human language such as soccer commentaries is extremely difficult at all resolutions, and it is most challenging at finer resolutions where data is sparsest and small errors cannot be as easily normalized. Our work will help open new avenues of research into this difficult and exciting problem.

This paper presents a new method for the multi-resolution grounding of complex natural language in a detailed world representation. Our factor graph allows us to decompose the grounding problem into the more tractable subproblems of segmenting the language into fragments and aligning the fragments with the world representation. In the segmentation phase, we make use of linguistic theories of discourse to create a proxy of labels from which we learn statistical and structural features of good splitpoints. In the alignment phase, we bootstrap the learning of finer-grained correspondences between the language and the world representation with rough alignments from a state-of-the-art system. We combine these phases in a dynamic programming setup which allows us to efficiently optimize our objective.

We have shown that factoring the acquisition problem into separate alignment and segmentation phases improves performance on several evaluation metrics. We achieve considerable improvements over the previous state of the art on finer-resolution alignments in the domain of professional soccer commentaries, and we show that we can leverage groundings at one resolution to improve alignments in another.

Several extensions of this work are possible. We would like to annotate more games to improve our dataset. We could improve our model by encoding the dynamics of the environment. We did not attempt to learn this information in our process, but it is likely that modeling the event transition probabilities could provide better results. A larger future work would extend the method outlined herein to produce templates for automated commentary generation.

Acknowledgments

This research was supported in part by a grant from the NSF (IIS-1352249), and the Royalty Research Fund (RRF) at the University of Washington. The authors also wish to thank Gina-Anne Levow, Yoav Artzi, Ben Hixon, and the anonymous reviewers for their valuable feedback on this work.

References

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

Alan Black and Kevin Lenzo. 2000. Building voices in the festival speech synthesis system.

Antoine Bordes, Nicolas Usunier, and Jason Weston. 2010. Label ranking under ambiguous supervision for learning semantic correspondences. In *Proceedings of The 27th International Conference on Machine Learning*, pages 103–110.

S. R. K. Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 82–90.

Wallace Chafe. 1988. Punctuation and the prosody of written language. *Written communication*, 5(4):395–426.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th International Conference on Machine Learning*, pages 128–135.

David L. Chen, Joohyun Kim, and Raymond J. Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *Journal of Artificial Intelligence Research (JAIR)*, 37:397–435.

Donald Davidson. 1967. The logical form of action sentences. *The logic of Decision and Action*.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. *URL http://nlp. stanford. edu/software/dependencies manual. pdf*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Ruifang Ge and Raymond J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*.

Hannaneh Hajishirzi, Julia Hockenmaier, Erik T. Mueller, and Eyal Amir. 2011. Reasoning about robocup soccer narratives. In *Proceedings of the 27th conference on Uncertainty in Artificial Intelligence*, pages 291–300.

Hannaneh Hajishirzi, Mohammad Rastegari, Ali Farhadi, and Jessica K Hodgins. 2012. Semantic understanding of professional soccer commentaries. In *Proceedings of the 28th conference on Uncertainty in Artificial Intelligence*.

Rohit J. Kate and Raymond J. Mooney. 2007. Learning language semantics from ambiguous supervision. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 895–900.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99.

Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. 2011. Ensemble of exemplar-svms for object detection and beyond. In *Proceedings of the 13th International Conference on Computer Vision.*

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012a. A Joint Model of Language and Perception for Grounded Attribute Learning. In *Proc. of the 2012 International Conference on Machine Learning*, Edinburgh, Scotland, June.

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. 2012b. Learning to parse natural language commands to a robot control system. In *Proc. of the 13th International Symposium on Experimental Robotics (ISER)*, June.

Opta. 2012. http://www.optasports.com.

Janet Pierrehumbert and Julia Hirschberg. 1990. The meaning of intonational contours in the interpretation of discourse. *Intentions in Communication*, 271.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind K Joshi. 2008. Easily identifiable discourse relations. *Technical Reports (CIS)*, page 884.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1–10.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Abhinav Shrivastava, Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. 2011. Data-driven visual similarity for cross-domain image matching. *ACM Transaction of Graphics (TOG) (Proceedings of ACM SIGGRAPH ASIA)*, 30(6).

Benjamin Snyder and Regina Barzilay. 2007. Database-text alignment via structured multilabel classification. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1713–1718.

Andrew J Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269.

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

Haonan Yu and Jeffrey Mark Siskind. 2013. Grounded language learning from video described with sentences. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 53–63.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 658–666.

Incorporating Vector Space Similarity in Random Walk Inference over Knowledge Bases

Matt Gardner
Carnegie Mellon University
mg1@cs.cmu.edu

Partha Talukdar*
Indian Institute of Science
ppt@serc.iisc.in

Jayant Krishnamurthy
Carnegie Mellon University
jayantk@cs.cmu.edu

Tom Mitchell
Carnegie Mellon University
tom@cs.cmu.edu

Abstract

Much work in recent years has gone into the construction of large knowledge bases (KBs), such as Freebase, DBPedia, NELL, and YAGO. While these KBs are very large, they are still very incomplete, necessitating the use of inference to fill in gaps. Prior work has shown how to make use of a large text corpus to augment random walk inference over KBs. We present two improvements to the use of such large corpora to augment KB inference. First, we present a new technique for combining KB relations and surface text into a single graph representation that is much more compact than graphs used in prior work. Second, we describe how to incorporate vector space similarity into random walk inference over KBs, reducing the feature sparsity inherent in using surface text. This allows us to combine distributional similarity with symbolic logical inference in novel and effective ways. With experiments on many relations from two separate KBs, we show that our methods significantly outperform prior work on KB inference, both in the size of problem our methods can handle and in the quality of predictions made.

1 Introduction

Much work in recent years has gone into the construction of large knowledge bases, either by collecting contributions from many users, as with Freebase (Bollacker et al., 2008) and DBPedia (Mendes et al., 2012), or automatically from web text or other resources, as done by NELL (Carlson et al., 2010) and YAGO (Suchanek et al., 2007). These knowledge bases contain millions of real-world entities and relationships between them. However, even though they are very large, they are still very incomplete, missing large fractions of possible relationships between common entities (West et al., 2014). Thus the task of *inference* over these knowledge bases, predicting new relationships simply by examining the knowledge base itself, has become increasingly important.

A promising technique for inferring new relation instances in a knowledge base is random walk inference, first proposed by Lao and Cohen (2010). In this method, called the Path Ranking Algorithm (PRA), the knowledge base is encoded as a graph, and random walks are used to find paths that connect the source and target nodes of relation instances. These paths are used as features in a logistic regression classifier that predicts new instances of the given relation. Each path can be viewed as a horn clause using knowledge base relations as predicates, and so PRA can be thought of as a kind of discriminatively trained logical inference.

One major deficiency of random walk inference is the connectivity of the knowledge base graph—if there is no path connecting two nodes in the graph, PRA cannot predict any relation instance between them. Thus prior work has introduced the use of a text corpus to increase the connectivity of the graph used as input to PRA (Lao et al., 2012; Gardner et al., 2013). This approach is not without its own problems, however. Whereas knowledge base relations are semantically coherent and different relations have distinct meanings, this is not

* Research carried out while at the Machine Learning Department, Carnegie Mellon University.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 397–406,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

true of surface text. For example, "The Nile *flows through* Cairo" and "The Nile *runs through* Cairo" have very similar if not identical meaning. Adding a text corpus to the inference graph increases connectivity, but it also dramatically increases feature sparsity.

We introduce two new techniques for making better use of a text corpus for knowledge base inference. First, we describe a new way of incorporating the text corpus into the knowledge base graph that enables much more efficient processing than prior techniques, allowing us to approach problems that prior work could not feasibly solve. Second, we introduce the use of vector space similarity in random walk inference in order to reduce the sparsity of surface forms. That is, when following a sequence of edge types in a random walk on a graph, we allow the walk to follow edges that are semantically similar to the given edge types, as defined by some vector space embedding of the edge types. If a path calls for an edge of type "flows through", for example, we accept other edge types (such as "runs through") with probability proportional to the vector space similarity between the two edge types. This lets us combine notions of distributional similarity with symbolic logical inference, with the result of decreasing the sparsity of the feature space considered by PRA. We show with experiments using both the NELL and Freebase knowledge bases that this method gives significantly better performance than prior approaches to incorporating text data into random walk inference.

2 Graph Construction

Our method for knowledge base inference, described in Section 3, performs random walks over a graph to obtain features for a logistic regression classifier. Prior to detailing that technique, we first describe how we produce a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \mathcal{R})$ from a set of knowledge base (KB) relation instances and a set of surface relation instances extracted from a corpus. Producing a graph from a knowledge base is straightforward: the set of nodes $\mathcal{N}$ is made up of the entities in the KB; the set of edge types $\mathcal{R}$ is the set of relation types in the KB, and the typed edges $\mathcal{E}$ correspond to relation instances from the KB, with one edge of type r connecting entity nodes for each (n_1, r, n_2) triple in the KB. Less straightforward is how to construct a graph from a corpus, and how to connect that graph to the KB graph. We describe our methods for each of those below.

To create a graph from a corpus, we first preprocess the corpus to obtain a collection of *surface relations*, such as those extracted by open information extraction systems like OLLIE (Mausam et al., 2012). These surface relations consist of a pair of noun phrases in the corpus, and the verb-like connection between them (either an actual verb, as done by Talukdar et al. (2012), a dependency path, as done by Riedel et al. (2013), or OpenIE relations (Mausam et al., 2012)). The verb-like connections are naturally represented as edges in the graph, as they have a similar semantics to the knowledge base relations that are already represented as edges. We thus create a graph from these triples exactly as we do from a KB, with nodes corresponding to noun phrase types and edges corresponding to surface relation triples.

So far these two subgraphs we have created are entirely disconnected, with the KB graph containing nodes representing entities, and the surface relation graph containing nodes representing noun phrases, with no edges between these noun phrases and entities. We connect these two graphs by making use of the ALIAS relation in the KB, which links entities to potential noun phrase referents. Each noun phrase in the surface relation graph is connected to those entity nodes which the noun phrase can possibly refer to according to the KB. These edges are not the output of an entity linking system, as done by Lao et al. (2012), but express instead the notion that the noun phrase *can* refer to the KB entity. The use of an entity linking system would certainly allow a stronger connection between noun phrase nodes and entity nodes, but it would require much more preprocessing and a much larger graph representation, as each mention of each noun phrase would need its own node, as opposed to letting every mention of the same noun phrase share the same node. This graph representation allows us to add tens of millions of surface relations to a graph of tens of millions of KB relations, and perform all of the processing on a single machine.

As will be discussed in more detail in Section 4, we also allow edge types to optionally have an associated vector that ideally captures something of the semantics of the edge type.

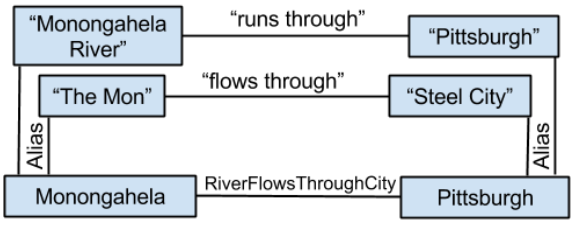
Figure 1 shows the graph constructions used in our experiments on a subset of KB and surface re-

KB Relations:
(Monongahela, RIVERFLOWSTHROUGHCITY, Pittsburgh)
(Pittsburgh, ALIAS, "Pittsburgh")
(Pittsburgh, ALIAS, "Steel City")
(Monongahela, ALIAS, "Monongahela River")
(Monongahela, ALIAS, "The Mon")

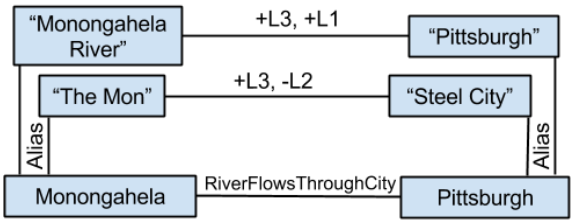
Surface Relations:
("The Mon", "flows through", "Steel City")
("Monongahela River", "runs through", "Pittsburgh")

Embeddings:
"flows through": [.2, -.1, .9]
"runs through": [.1, -.3, .8]

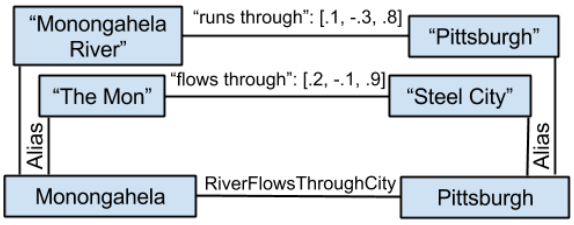
(a) An example data set.



(b) An example graph that combines a KB and surface relations.



(c) An example graph that replaces surface relations with a cluster label, as done by Gardner et al. (2013). Note, however, that the graph structure differs from that prior work; see Section 5.



(d) An example graph that uses vector space representations of surface edges, as introduced in this paper.

Figure 1: Example graph construction as used in the experiments in this paper. A graph using only KB edges is simply a subset of these graphs containing only the RIVERFLOWSTHROUGHCITY edge, and is not shown.

lations. Note that Figures 1b and 1c are shown as rough analogues of graphs used in prior work (described in more detail in Section 5), and we use them for comparison in our experiments.

3 The Path Ranking Algorithm

We perform knowledge base inference using the Path Ranking Algorithm (PRA) (Lao and Cohen, 2010). We begin this section with a brief overview of PRA, then we present our modification to the PRA algorithm that allows us to incorporate vector space similarity into random walk inference.

PRA can be thought of as a method for exploiting local graph structure to generate non-linear feature combinations for a prediction model. PRA generates a feature matrix over pairs of nodes in a graph, then uses logistic regression to classify those node pairs as belonging to a particular relation.

More formally, given a graph $\mathcal{G}$ with nodes $\mathcal{N}$, edges $\mathcal{E}$, and edge labels $\mathcal{R}$, and a set of node pairs $(s_i, t_i) \in \mathcal{D}$, one can create a connectivity matrix where rows correspond to node pairs and columns correspond to edge lables. PRA *augments* this matrix with additional columns corresponding to *sequences* of edge labels, called *path types*, and changes the cell values from representing the presence of an edge to representing the specificity of the connection that the path type makes between the node pair.

Because the feature space considered by PRA is so large (the set of all possible edge label sequences, with cardinality $\sum_{i=1}^{l} |\mathcal{R}|^i$, assuming a bound l on the maximum path length), the first step PRA must perform is *feature selection*, which is done using random walks over the graph. The second step of PRA is *feature computation*, where each cell in the feature matrix is computed using a constrained random walk that follows the path type corresponding to the feature. We now explain each of these steps in more detail.

Feature selection finds path types π that are likely to be useful in predicting new instances of the relation represented by the input node pairs . These path types are found by performing random walks on the graph $\mathcal{G}$ starting at the source and target nodes in $\mathcal{D}$, recording which paths connect some source node with its target. The edge sequences are ranked by frequency of connecting a source node to a corresponding target node, and the top k are kept.

Feature computation. Once a set of path types

is selected as features, the next step of the PRA algorithm is to compute a value for each cell in the feature matrix, corresponding to a node pair and a path type. The value computed is the probability of arriving at the target node of a node pair, given that a random walk began at the source node and was constrained to follow the path type: $\mathrm{p}(t|s, \pi)$.

Once these steps have been completed, the resulting feature matrix can be used with whatever model or learning algorithm is desired; in this and prior work, simple logistic regression has been used as the prediction algorithm.

4 Vector space random walks

Our modifications to PRA are confined entirely to the feature computation step described above; feature selection (finding potentially useful sequences of edge types) proceeds as normal, using the symbolic edge types. When computing feature values, however, we allow a walk to follow an edge that is *semantically similar* to the edge type in the path, as defined by Euclidean distance in the vector space.

More formally, consider a path type π. Recall that π is a sequence of edge types $< e_1, e_2, \ldots, e_l >$, where l is the length of the path; we will use π_i to denote the i^{th} edge type in the sequence. To compute feature values, PRA begins at some node and follows edges of type π_i until the sequence is finished and a target node has been reached. Specifically, if a random walk is at a node n with m outgoing edge types $\{e_1, e_2, \ldots, e_m\}$, PRA selects the edge type from that set which matches π_i, then selects uniformally at random from all outgoing edges of that type. If there is no match in the set, the random walk restarts from the original start node.

We modify the selection of which edge type to follow. When a random walk is at a node n with m outgoing edge types $\{e_1, e_2, \ldots, e_m\}$, instead of selecting only the edge type that matches π_i, we allow the walk to select instead an edge that is close to π_i in vector space. For each edge type at node n, we select the edge with the following probability:

$$p(e_j|\pi_i) \propto \exp(\beta \times v(e_j) \cdot v(\pi_i)),\ \forall j, 1 \leq j \leq m$$

where $v(\cdot)$ is a function that returns the vector representation of an edge type, and β is a spikiness parameter that determines how much weight to give to the vector space similarity. As β approaches infinity, the normalized exponential approximates a delta function on the closest edge type to π_i, in $\{e_1, e_2, \ldots, e_m\}$. If π_i is in the set of outgoing edges, this algorithm converges to the original PRA.

However, if π_i is not in the set of outgoing edge types at a node and all of the edge types are very dissimilar to π_i, this algorithm (with β not close to infinity) will lead to a largely uniform distribution over edge types at that node, and no way for the random walk to restart. To recover the restart behavior of the original PRA, we introduce an additional restart parameter α, and add another value to the categorical distribution before normalization:

$$p(\text{restart}|\pi_i) \propto \exp(\beta * \alpha)$$

When this *restart* type is selected, the random walk begins again, following π_1 starting at the source node. With α set to a value greater than the maximum similarity between (non-identical) edge type vectors, and β set to infinity, this algorithm exactly replicates the original PRA.

Not all edge types have vector space representations: the ALIAS relation cannot have a meaningful vector representation, and we do not use vectors to represent KB relations, finding that doing so was not useful in practice (which makes intuitive sense: KB relations are already latent representations themselves). While performing random walks, if π_i has no vector representation, we fall back to the original PRA algorithm for selecting the next edge.

We note here that when working with vector spaces it is natural to try clustering the vectors to reduce the parameter space. Each path type π is a feature in our model, and if two path types differ only in one edge type, and the differing edge types have very similar vectors, the resultant feature values will be essentially identical for both path types. It seems reasonable that running a simple clustering algorithm over these path types, to reduce the number of near-duplicate features, would improve performance. We did not find this to be the case, however; all attempts we made to use clustering over these vectors gave performance indistinguishable from not using clustering. From this we conclude that the main issue hindering performance when using PRA over these kinds of graphs is one of limited connectivity, not one of too many parameters in the model. Though the

feature space considered by PRA is very large, the number of attested features in a real graph is much smaller, and it is this sparsity which our vector space methods address.

5 Related Work

Knowledge base inference. Random walk inference over knowledge bases was first introduced by Lao and Cohen (2010). This work was improved upon shortly afterward to also make use of a large corpus, by representing the corpus as a graph and connecting it to the knowledge base (Lao et al., 2012). Gardner et al. (2013) further showed that replacing surface relation labels with a representation of a latent embedding of the relation led to improved prediction performance. This result is intuitive: the feature space considered by PRA is exponentially large, and surface relations are sparse. The relations "[river] flows through [city]" and "[river] runs through [city]" have near identical meaning, and both should be very predictive for the knowledge base relation RIVERFLOWSTHROUGHCITY. However, if one of these relations only appears in the training data and the other only appears in the test data, neither will be useful for prediction. Gardner et al. (2013) attempted to solve this issue by finding a latent symbolic representation of the surface relations (such as a clustering) and replacing the edge labels in the graph with these latent representations. This makes it more likely for surface relations seen in training data to also be seen at test time, and naturally improved performance.

This representation, however, is still brittle, as it is still a symbolic representation that is prone to mismatches between training and test data. If the clustering algorithm used is too coarse, the features will not be useful, and if it is too fine, there will be more mismatches. Also, verbs that are on the boundaries of several clusters are problematic to represent in this manner. We solve these problems by modifying the PRA algorithm to directly use vector representations of edge types during the random walk inference.

These two prior techniques are the most directly related work to what we present in this paper, and we compare our work to theirs.

Graph construction. In addition to the incorporation of vector space similarity into the PRA algorithm, the major difference between our work and the prior approaches mentioned above is in the construction of the graph used by PRA. We contrast our method of graph construction with these prior approaches in more detail below.

Lao et al. (2012) represent every word of every sentence in the corpus as a node in the graph, with edges between the nodes representing dependency relationships between the words. They then connect this graph to the KB graph using a simple entity linking system (combined with coreference resolution). The resultant graph is enormous, such that they needed to do complex indexing on the graph and use a cluster of 500 machines to perform the PRA computations. Also, as the edges represent dependency labels, not words, with this graph representation the PRA algorithm does not have access to the verbs or other predicative words that appear in the corpus, which frequently express relations. PRA only uses *edge* types as feature components, not *node* types, and so the rich information contained in the words is lost. This graph construction also would not allow the incorporation of vector space similarity that we introduced, as dependency labels do not lend themselves well to vector space representations.

Gardner et al. (2013) take an approach very similar to the one presented in Section 2, preprocessing the corpus to obtain surface relations. However, instead of creating a graph with nodes representing noun phrases, they added edges from the surface relations directly to the entity nodes in the graph. Using the ALIAS relation, as we do, they added an edge between *every possible* concept pair that could be represented by the noun phrases in a surface relation instance. This leads to some nonsensical edges added to the graph, and if the ALIAS relation has high degree (as it does for many common noun phrases in Freebase), it quickly becomes unscalable—this method of graph construction runs out of disk space when attempting to run on the Freebase experiments in Section 6. Also, in conflating entity nodes in the graph with noun phrases, they lose an important distinction that turns out to be useful for prediction, as we discuss in Section 6.4.[1]

[1]Recent notions of "universal schema" (Riedel et al., 2013) also put KB entities and noun phrases into the same conceptual space, though they opt for using noun phrases instead of the KB entities used by Gardner et al. In general this is problematic, as it relies on some kind of entity linking system as preprocessing, and cannot handle common noun references of proper entities without losing information. Our method, and that of Lao et al., skirts this issue entirely by not trying to merge KB entities with noun phrases.

Other related work. Also related to the present work is recent research on programming languages for probabilistic logic (Wang et al., 2013). This work, called ProPPR, uses random walks to locally ground a query in a small graph before performing propositional inference over the grounded representation. In some sense this technique is like a recursive version of PRA, allowing for more complex inferences than a single iteration of PRA can make. However, this technique has not yet been extended to work with large text corpora, and it does not yet appear to be scalable enough to handle the large graphs that we use in this work. How best to incorporate the work presented in this paper with ProPPR is an open, and very interesting, question.

Examples of other systems aimed at reasoning over common-sense knowledge are the CYC project (Lenat, 1995) and ConceptNet (Liu and Singh, 2004). These common-sense resources could easily be incorporated into the graphs we use for performing random walk inference.

Lines of research that seek to incorporate distributional semantics into traditional natural language processing tasks, such as parsing (Socher et al., 2013a), named entity recognition (Passos et al., 2014), and sentiment analysis (Socher et al., 2013b), are also related to what we present in this paper. While our task is quite different from these prior works, we also aim to combine distributional semantics with more traditional methods (in our case, symbolic logical inference), and we take inspiration from these methods.

6 Experiments

We perform both the feature selection step and the feature computation step of PRA using GraphChi, an efficient single-machine graph processing library (Kyrola et al., 2012). We use MALLET's implementation of logistic regression, with both L1 and L2 regularization (McCallum, 2002). To obtain negative evidence, we used a closed world assumption, treating any (source, target) pair found during the feature computation step as a negative example if it was not given as a positive example. We tuned the parameters to our methods using a coarse, manual grid search with cross validation on the training data described below. The parameters we tuned were the L1 and L2 regularization parameters, how many random walks to perform in the feature selection and computation steps of PRA, and spikiness and restart parameters for vector space walks. The results presented were not very sensitive to changes in these parameters.

	NELL	Freebase
Entities	1.2M	20M
Relation instances	3.4M	67M
Total relation types	520	4215
Relation types tested	10	24
Avg. instances/relation	810	200
SVO triples used	404k	28M

Table 1: Statistics of the data used in our experiments.

6.1 Data

We ran experiments on both the NELL and Freebase knowledge bases. The characteristics of these knowledge bases are shown in Table 1. The Freebase KB is very large; to make it slightly more manageable we filtered out relations that did not seem applicable to relation extraction, as well as a few of the largest relations.[2] This still left a very large, mostly intact KB, as can be seen in the table. For our text corpus, we make use of a set of subject-verb-object triples extracted from dependency parses of ClueWeb documents (Talukdar et al., 2012). There are 670M such triples in the data set, most of which are completely irrelevant to the knowledge base relations we are trying to predict. For each KB, we filter the SVO triples, keeping only those which can possibly connect training and test instances of the relations we used in our experiments. The number of SVO triples kept for each KB is also shown in Table 1. We obtained vector space representations of these surface relations by running PCA on the SVO matrix.

We selected 10 NELL relations and 24 Freebase relations for testing our methods. The NELL relations were hand-selected as the relations with the largest number of known instances that had a reasonable precision (the NELL KB is automatically created, and some relations have low precision). We split the known instances of these relations into 75% training and 25% testing, giving on average about 650 training instances and 160 test

[2] We removed anything under /user, /common, /type (except for the relation /type/object/type), /base, and /freebase, as not applicable to our task. We also removed relations dealing with individual music tracks, book editions, and TV epidsodes, as they are very large, very specific, and unlikely to be useful for predicting the relations in our test set.

instances for each relation.

The 24 Freebase relations were semi-randomly selected. We first filtered the 4215 relations based on two criteria: the number of relation instances must be between 1000 and 10000, and there must be no mediator in the relation.[3] Once we selected the relations, we kept all instances of each relation that had some possible connection in the SVO data.[4] This left on average 200 instances per relation, which we again split 75%-25% into training and test sets.

6.2 Methods

The methods we compare correspond to the graphs shown in Figure 1. The KB method uses the original PRA algorithm on just the KB relations, as presented by Lao and Cohen (2010). KB + SVO adds surface relations to the graph (Figure 1b). We present this as roughly analogous to the methods introduced by Lao et al. (2012), though with some significant differences in graph representation, as described in Section 5. KB + Clustered SVO follows the methods of Gardner et al. (2013), but using the graph construction introduced in this paper (Figure 1c; their graph construction techniques would have made graphs too large to be feasible for the Freebase experiments). KB + Vector SVO is our method (Figure 1d).

6.3 Evaluation

As evaluation metrics, we use *mean average precision* (MAP) and *mean reciprocal rank* (MRR), following recent work evaluating relation extraction performance (West et al., 2014). We test significance using a paired permutation test.

The results of these experiments are shown in Table 2 and Table 3. In Table 4 we show average precision for every relation tested on the NELL KB, and we show the same for Freebase in Table 5.

6.4 Discussion

We can see from the tables that KB + Vector SVO (the method presented in this paper) significantly outperforms prior approaches in both MAP and MRR. We believe that this is due to the reduction in feature sparsity enabled by using vector space instead of symbolic representations (as that is the only real difference between KB + Clustered SVO and KB + Vector SVO), allowing PRA to make better use of path types found in the training data. When looking at the results for individual relations in Table 4 and Table 5, we see that KB + Vector SVO outperforms other methods on the majority of relations, and it is a close second when it does not.

We can also see from the results that mean average precision seems a little low for all methods tested. This is because MAP is computed as the precision of *all* possible correct predictions in a ranked list, where precision is counted as 0 if the correct prediction is not included in the list. In other words, there are many relation instances in our randomly selected test set that are not inferrable from the knowledge base, and the low recall hurts the MAP metric. MRR, which judges the precision of the top prediction for each relation, gives us some confidence that the main issue here is one of recall, as MRR is reasonably high, especially on the NELL KB. As further evidence, if we compute average precision for each query node (instead of for each relation), excluding queries for which the system did not return any predictions, MAP ranges from .29 (KB) to .45 (KB + Vector SVO) on NELL (with around 30% of queries having no prediction), and from .40 (KB) to .49 (KB +

Method	MAP	MRR
KB	0.193	0.635
KB + SVO	0.218	0.763
KB + Clustered SVO	0.276	0.900
KB + Vector SVO	**0.301**	**0.900**

Table 2: Results on the NELL knowledge base. The bolded line is significantly better than all other results with $p < 0.025$.

Method	MAP	MRR
KB	0.278	0.614
KB + SVO	0.294	0.639
KB + Clustered SVO	0.326	0.651
KB + Vector SVO	**0.350**	**0.670**

Table 3: Results on the Freebase knowledge base. The bolded line is significantly better than all other results with $p < 0.0002$.

[3] A mediator in Freebase is a reified relation instance meant to handle n-ary relations, for instance /film/performance. PRA in general, and our implementation of it in particular, needs some modification to be well-suited to predicting relations with mediators.

[4] We first tried randomly selecting instances from these relations, but found that the probability of selecting an instance that benefited from an SVO connection was negligible. In order to make use of the methods we present, we thus restricted ourselves to only those that had a possible SVO connection.

Relation	KB	KB + SVO	KB + Clustered SVO	KB + Vector SVO
ActorStarredInMovie	0.000	0.032	0.032	**0.037**
AthletePlaysForTeam	0.200	0.239	0.531	**0.589**
CityLocatedInCountry	0.126	0.169	0.255	**0.347**
JournalistWritesForPublication	0.218	0.254	0.291	**0.319**
RiverFlowsThroughCity	0.000	0.001	0.052	**0.076**
SportsTeamPositionForSport	0.217	**0.217**	0.178	0.180
StadiumLocatedInCity	0.090	0.156	0.275	**0.321**
StateHasLake	0.000	0.000	**0.000**	0.000
TeamPlaysInLeague	0.934	0.936	**0.947**	0.939
WriterWroteBook	0.144	0.179	0.195	**0.202**

Table 4: Average precision for each relation tested on the NELL KB. The best performing method on each relation is bolded.

Relation	KB	KB + SVO	KB + C-SVO	KB + V-SVO
/amusement_parks/park/rides	0.000	0.009	0.004	**0.013**
/architecture/architect/structures_designed	0.072	0.199	0.257	**0.376**
/astronomy/constellation/contains	0.004	**0.017**	0.000	0.008
/automotive/automotive_class/examples	0.003	0.001	0.002	**0.006**
/automotive/model/automotive_class	0.737	0.727	0.742	**0.768**
/aviation/airline/hubs	0.322	0.286	0.298	**0.336**
/book/literary_series/author_s	0.798	0.812	0.818	**0.830**
/computer/software_genre/software_in_genre	0.000	0.001	0.001	**0.001**
/education/field_of_study/journals_in_this_discipline	0.001	**0.003**	0.003	0.001
/film/film/rating	**0.914**	0.905	0.914	0.905
/geography/island/body_of_water	0.569	0.556	0.580	**0.602**
/geography/lake/basin_countries	0.420	0.361	0.409	**0.437**
/geography/lake/cities	0.111	0.134	**0.177**	0.175
/geography/river/cities	0.030	0.038	0.045	**0.066**
/ice_hockey/hockey_player/hockey_position	0.307	0.243	0.222	**0.364**
/location/administrative_division/country	0.989	0.988	**0.991**	0.989
/medicine/disease/symptoms	0.061	**0.078**	0.068	0.067
/medicine/drug/drug_class	**0.169**	0.164	0.135	0.157
/people/ethnicity/languages_spoken	0.134	**0.226**	0.188	0.223
/spaceflight/astronaut/missions	0.010	0.186	0.796	**0.848**
/transportation/bridge/body_of_water_spanned	0.534	0.615	0.681	**0.727**
/tv/tv_program_creator/programs_created	0.164	0.179	0.163	**0.181**
/visual_art/art_period_movement/associated_artists	0.044	0.040	**0.046**	0.037
/visual_art/visual_artist/associated_periods_or_movements	0.276	**0.295**	0.282	0.290

Table 5: Average precision for each relation tested on the Freebase KB. The best performing method on each relation is bolded. For space considerations, "Clustered SVO" is shortened to "C-SVO" and "Vector SVO" is shortened to "V-SVO" in the table header.

Vector SVO) on Freebase, (where 21% of queries gave no prediction). Our methods thus also improve MAP when calculated in this manner, but it is not an entirely fair metric,[5] so we use standard MAP to present our main results.

One interesting phenomenon to note is a novel use of the ALIAS relation in some of the relation models. The best example of this was found with the relation `/people/ethnicity/languages_spoken`. A high-weighted feature when adding surface relations was the edge sequence <ALIAS, ALIAS INVERSE>. This edge sequence reflects the fact that languages frequently share a name with the group of people that speaks them (e.g., Maori, French). And because PRA can generate compositional features, we also find the following edge sequence for the same relation: </people/ethnicity/included_in_group, ALIAS, ALIAS INVERSE>. This feature captures the same notion that languages get their names from groups of people, but applies it to subgroups within an ethnicity. These features would be very difficult, perhaps impossible, to include in systems that do not distinguish between noun phrases and knowledge base entities, such as the graphs constructed by Gardner et al. (2013), or typical relation extraction systems, which generally only work with noun phrases after performing a heuristic entity linking.

7 Conclusion

We have offered two main contributions to the task of knowledge base inference. First, we have presented a new technique for combining knowledge base relations and surface text into a single graph representation that is much more compact than graphs used in prior work. This allowed us to apply methods introduced previously to much larger problems, running inference on a single machine over the entire Freebase KB combined with tens of millions of surface relations. Second, we have described how to incorporate vector space similarity into random walk inference over knowledge bases, reducing the feature sparsity inherent in using surface text. This allows us to combine distributional similarity with symbolic logical inference in novel and effective ways. With experiments on many relations from two separate knowledge bases, we have shown that our methods significantly outperform prior work on knowledge base inference.

The code and data used in the experiments in this paper are available at http://rtw.ml.cmu.edu/emnlp2014_vector_space_pra/.

[5] MAP is intended to include some sense of recall, but excluding queries with no predictions removes that and opens the metric to opportunistic behavior.

Acknowledgments

This research has been supported in part by DARPA under contract number FA8750-13-2-0005, by NSF under grant 31043,18,1121946, and by generous support from Yahoo! and Google.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*.

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

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Aapo Kyrola, Guy Blelloch, and Carlos Guestrin. 2012. Graphchi: Large-scale graph computation on just a pc. In *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 31–46.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of EMNLP-CoNLL*.

Douglas B Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Hugo Liu and Push Singh. 2004. Conceptnet: a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit.

Pablo N. Mendes, Max Jakob, and Christian Bizer. 2012. Dbpedia for nlp: A multilingual cross-domain knowledge base. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013a. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

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

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of WWW*.

Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. 2012. Acquiring temporal constraints between relations. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 992–1001. ACM.

William Yang Wang, Kathryn Mazaitis, and William W. Cohen. 2013. Programming with personalized pagerank: A locally groundable first-order probabilistic logic. In *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management*, CIKM '13, pages 2129–2138, New York, NY, USA. ACM.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *WWW*.

Composition of Word Representations Improves Semantic Role Labelling

Michael Roth and **Kristian Woodsend**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
{mroth,kwoodsen}@inf.ed.ac.uk

Abstract

State-of-the-art semantic role labelling systems require large annotated corpora to achieve full performance. Unfortunately, such corpora are expensive to produce and often do not generalize well across domains. Even in domain, errors are often made where syntactic information does not provide sufficient cues. In this paper, we mitigate both of these problems by employing distributional word representations gathered from unlabelled data. While straight-forward word representations of predicates and arguments improve performance, we show that further gains are achieved by composing representations that model the interaction between predicate and argument, and capture full argument spans.

1 Introduction

The goal of *semantic role labelling* (SRL) is to discover the relations that hold between a predicate and its arguments in a given input sentence (e.g., "who" did "what" to "whom", "when", "where", and "how"). This semantic knowledge at the predicate-argument level is required by inference-based NLP tasks in order to identify meaning-preserving transformations, such as active/passive, verb alternations and nominalizations. Several manually-build semantic resources, including FrameNet (Ruppenhofer et al., 2010) and PropBank (Palmer et al., 2005), have been developed with the goal of documenting and providing examples of such transformations and how they preserve semantic role information. Given that labelled corpora are inevitably restricted in size and coverage, and that syntactic cues are not by themselves unambiguous or sufficient, the success of systems that automatically provide corresponding analyses has been limited in practice.

Recent work on SRL has explored approaches that can leverage unlabelled data, following a semi-supervised (Fürstenau and Lapata, 2012; Titov and Klementiev, 2012) or unsupervised learning paradigm (Abend et al., 2009; Titov and Klementiev, 2011). Unlabelled data provides additional statistical strength and can lead to more consistent models. For instance, latent representations of words can be computed, based on distributional similarity or language modelling, which can be used as additional features during traditional supervised learning. Although we would expect that extra features would improve classifier performance, this seems in part counter-intuitive. Just because one word has a specific representation does not mean that it should be assigned a specific argument label. Instead, one would expect a more complex interplay between predicate, argument and the context they appear in.

In this paper, we investigate the impact of distributional word representations for SRL. Initially, we augment the feature space with word representations for a predicate and its argument head. Furthermore, we use a compositional approach to model a representation of the full argument, by composing a joint representation of all words in the argument span, and we also investigate the interaction between predicate and argument, using a compositional representation of the dependency path. We demonstrate the benefits of these compositional features using a state-of-the-art semantic role labeller, which we evaluate on the English part of the CoNLL-2009 data set.

2 Related Work

Research into using distributional information in SRL dates back to Gildea and Jurafsky (2002), who used distributions over verb-object co-occurrence clusters to improve coverage in argument classification. The distribution of a word over these soft clusters assignments was added as

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 407–413,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

features to their classifier. The SRL system by Croce et al. (2010) combines argument clustering based on co-occurrence frequencies with a language model. Collobert et al. (2011) used distributional word representations in a neural network model that can update representations during training. Zapirain et al. (2013) suggested distributional information as a basis for a selectional preference model that can be used as a single additional feature for classifying potential arguments. Most recently, Hermann et al. (2014) used distributional word representations within pre-defined syntactic contexts as input to a classifier which learns to distinguish different predicate senses.

A complementary line of research explores the representation of sequence information. Prominent examples are the works by Deschacht and Moens (2009) and Huang and Yates (2010) who learned and applied Hidden Markov Models to assign state variables to words and word spans, which serve as supplementary features for classification. One drawback of this approach is that state variables are discrete and the number of states (i.e., their granularity) has to be chosen in advance.

The popularity of distributional methods for word representation has been a motivation for developing representations of larger constructions such as phrases and sentences, and there have been several proposals for computing the meaning of word combinations in vector spaces. Mitchell and Lapata (2010) introduced a general framework where composition is formulated as a function f of two vectors u and v. Depending on how f is chosen, different composition models arise, the simplest being an additive model where $f(u, v) = u + v$. To capture relational functions, Baroni and Zamparelli (2010) expanded on this approach by representing verbs, adjectives and adverbs by matrices which can modify the properties of nouns (represented by vectors). Socher et al. (2012) combined word representations with syntactic structure information, through a recursive neural network that learns vector space representations for multi-word phrases and sentences. An empirical comparison of these composition methods was provided in (Blacoe and Lapata, 2012).

In this work, we use type-based continuous representations of words to compose representations of multiple word sequences and spans, which can then be incorporated directly as features into SRL systems.

Distributional Feature	Computation
Argument a	$\vec{a}$
Predicate p	$\vec{p}$
Predicate-argument **Interaction**	$\vec{a} + \vec{p}$
Argument **Span** $w_1 \dots w_n$	$\Sigma_i \vec{w_i}$
Dependency **Path** from a to p	$\Sigma_{w \in \text{path}(a,p)} \vec{w}$

Table 1: Features based on distributional word representations and additive composition. Vector $\vec{w}$ denotes the representation of word w.

3 Method

Following the set-up of the CoNLL shared task in 2009, we consider predicate-argument structures that consist of a verbal or nominal predicate p and PropBank-labelled arguments $a_i \in \{a_1 \dots a_n\}$, where each a_i corresponds to the head word of the phrase that constitutes the respective argument. Traditional semantic role labelling approaches compute a set of applicable features on each pair $\langle p, a_i \rangle$, such as the observed lemma type of a word and the grammatical relation to its head, that serve as indicators for a particular role label.

The disadvantage of this approach lies in the fact that indicator features such as word and lemma type are often sparse in training data and hence do not generalize well across domains. In contrast, features based on distributional representations (e.g., raw co-occurrence frequencies) can be computed for every word, given that it occurs in some unlabelled corpus. In addition to this obvious advantage for out-of-domain settings, distributional representations can provide a more robust input signal to the classifier, for instance by projecting a matrix of co-occurrence frequencies to a lower-dimensional space. We hence hypothesize that such features enable the model to become more robust out-of-domain, while providing higher precision in-domain.

Although simply including the components of a word representation as features to a classifier can lead to immediate improvements in SRL performance, this observation seems in part counter-intuitive. Just because one word has a specific representation does not mean that it should be assigned a specific argument label. In fact, one would expect a more complex interplay between the representation of an argument a_i and the context it appears in. To model aspects of this interplay, we define an extended set of features that

further includes representations for the combination of p and a_i, the set of words in the dependency path between p and a_i, and the set of words in the full span of a_i. We compute additive compositional representations of multiple words, using the simplest method of Mitchell and Lapata (2010) where the composed representation is the uniformly weighted sum of each single representation. Our full set of feature types based on distributional word representations is listed in Table 1.

4 Experimental Setup

We evaluate the impact of different types of features by performing experiments on a benchmark dataset for semantic role labelling. To assess the gains of distributional representations realistically, we incorporate the features described in Section 3 into a state-of-the-art SRL system. The following paragraphs summarize the details of our experimental setup.

Semantic Role Labeller. In all our experiments, we use the publicly available system by Björkelund et al. (2010).[1] This system combines the first-ranked SRL system and the first-ranked syntactic parser in the CoNLL 2009 shared task for English (Björkelund et al., 2009; Bohnet, 2010). To the best of our knowledge, this combination represents the current state-of-the-art for semantic role labelling following the PropBank/NomBank paradigm (Palmer et al., 2005; Meyers et al., 2004). To re-train and evaluate models with different feature sets, we use the same training, development and test sets as provided in the CoNLL shared task (Hajič et al., 2009). Although the employed system features a full syntactic-semantic parsing pipeline, we only modify the feature sets of the two components directly related to the actual role labelling task, namely argument identification and argument classification.

Word Representations. As a baseline, we simply added as features the word representations of the predicate and argument head involved in a classification decision (first two lines in Table 1). We experimented with a range of publicly available sets of word representations, including embeddings from various neural language models (Mnih and Hinton, 2009; Collobert et al., 2011; Huang et al., 2012), eigenvectors (Dhillon et al., 2011), Brown clusters (Brown et al., 1992), and post-processed co-occurrence counts (Lebret and Collobert, 2014). Results on the development set for various off-the-shelf representations are shown in Table 2. The numbers reveal that any kind of word representation can be employed to improve results. We choose to perform all follow-up experiments using the 50-dimensional embeddings induced by Turian et al. (2010), using the method by Collobert et al., as they led to slightly better results in F_1-score than other representations. No significant differences were observed, however, using other types of representations or vector sizes.

Development	dims	P	R	F_1
None	–	86.1	81.0	83.5
Brown clusters[2]	320	86.2	81.3	83.7
Neural LM[2]	50	86.2	**81.4**	**83.7**
Neural LM+Global[3]	50	86.2	81.4	83.7
HLBL[2]	50	**86.3**	81.3	83.7
H-PCA[4]	50	86.2	81.3	83.7
Eigenwords[5]	50	86.2	81.3	83.6

Table 2: Results on the CoNLL-2009 development set, using off-the-shelf word representations for predicates and argument as additional features. Performance numbers in percent.

5 Results

We evaluate our proposed set of additional features on the CoNLL-2009 in-domain and out-of-domain test sets, using the aforementioned SRL system and word representations. All results are computed using the system's built-in preprocessing pipeline and re-trained models for argument identification and classification. We report labelled precision, recall and semantic F1-score as computed by the official scorer.

The upper part of Table 3 shows SRL performance on the in-domain CoNLL-2009 test set, with and without (**Original**) additional features based on distributional representations. The results reveal that any type of additional feature helps to improve precision and recall in this setting (from 85.2% F_1-score up to 85.5%), with significant gains for 4 of the 5 additional features (computed using a randomization test; cf. Yeh, 2000). Interestingly, we find that the features do not seem

[1] http://code.google.com/p/mate-tools/
[2] http://metaoptimize.com/projects/wordreprs/
[3] http://ai.stanford.edu/%7eehhuang/
[4] http://lebret.ch/words/
[5] http://www.cis.upenn.edu/%7eungar/eigenwords/

In-domain	P	R	F_1
Original	87.4	83.1	85.2
Original + Argument	87.6	83.3	85.4**
Original + Predicate	87.4	83.2	85.2
Original + Interaction	87.5	83.3	85.3**
Original + Span	**87.6**	**83.5**	**85.5****
Original + Path	87.5	83.4	85.4**
Original + All	87.6	83.4	85.5**

Out-of-domain	P	R	F_1
Original	76.9	71.7	74.2
Original + Argument	77.4	71.9	74.5
Original + Predicate	77.3	72.2	74.7*
Original + Interaction	77.2	72.0	74.5
Original + Span	77.3	72.3	74.7*
Original + Path	77.2	72.3	74.7*
Original + All	**77.5**	**73.0**	**75.2****

Table 3: Results on both CoNLL-2009 test sets. All numbers in percent. Significant differences from Original in terms of F_1-score are marked by asterisks (* $p<0.05$, ** $p<0.01$).

to have a cumulative effect here, as indicated by the results with all features (**+All**, 85.5% F_1). We conjecture that this is due to the high volume of existing in-domain training data, which renders our full feature set redundant. To test this conjecture, we further assess performance on the out-of-domain test set of the CoNLL-2009 shared task.

The results for the out-of-domain experiment are summarized in the lower part of Table 3. We again observe that each single feature type improves classification, with absolute gains being slightly higher than in the in-domain setting. More interestingly though, we find that the complete feature set boosts performance even further, achieving an overall gain in precision and recall of 0.6 and 1.3 percentage points, respectively. The resulting F_1-score of 75.2 lies even higher than the top score for this particular data set reported in the CoNLL shared task (Zhao et al., 2009; 74.6 F_1).

We next investigate the benefits of compositional representations over features for single words by assessing their impact on the overall result in an ablation study. Table 4 shows results of ablation tests performed for the three compositional feature types **Interaction**, **Span** and **Path** on the out-of-domain test set. The results reveal

Out-of-domain	P	R	F_1
Original	76.9	71.7	74.2
Full (Original+All)	**77.5**	**73.0**	**75.2**
Full −Interaction	77.2	72.5	74.8
Full −Span	77.2	72.3	74.7
Full −Path	77.6	72.3	74.8

Table 4: Results of an ablation study over features based on compositional representations. All numbers in percent.

a considerable loss in recall, indicating the importance of including compositional word representations and confirming our intuition that they can provide additional gains over simple type-level representations. In the next section, we discuss this result in more detail and provide examples of improved classification decisions.

6 Discussion

As a more detailed qualitative analysis, we examined the impact of word representations on SRL performance with respect to different argument labels and predicate types. Results on the in-domain data set, shown in the upper part of Table 5, suggest that most improvements in terms of precision are gained for verbal predicates, while nominal predicates primarily benefit from higher recall. One reason for the latter observation might be that arguments of nominal predicates are generally much harder to identify for the **Original** model, as the cues provided by indicator features on words and syntax are often inconclusive. For verbal predicates, the word representations mainly provide reinforcing signals to the classifier, improving its precision at a slight cost of recall.

The results on the out-of-domain data set provide more insights regarding the suitability of word representations for generalization. As shown in the lower half of Table 5, the additional features on average have a positive impact on precision and recall. For verbal predicates, we observe only one case, namely A0, in which improvements in recall came with a decline in precision. Regarding nominal predicates, the trend is similar to what we have seen in the in-domain setting, with most gains being achieved in terms of recall.

Apart from assessing quantitative effects, we further examined cases that directly show the qualitative gains of the compositional features defined

	Sentence with *predicate* and [gold argument$_{\text{label}}$]	**Original**	Features required for correction
(1)	He did not resent [their$_{\text{A0}}$] *supervision*	A1	**Interaction**
(2)	[He$_{\text{A1}}$] is getting plenty of *rest*	no label	**Interaction**, **Path**
(3)	[He$_{\text{A0}}$] rose late and went down to *have* breakfast.	no label	**Path**
(4)	He was able to *sit* [for hours$_{\text{AM-TMP}}$].	A2	**Span**
(5)	Because he had *spoken* [too softly$_{\text{AM-MNR}}$].	AM-TMP	**Span**

Table 6: Example sentences in which distributional features compensated for errors made by **Original**.

In-domain	verbal		nominal	
Label	P	R	P	R
A0	+0.4	+0.4	−0.1	+2.4
A1	+0.2	−0.4	+0.6	+1.5
A2	+1.7	−1.5	–	+2.5
AM-ADV	+0.8	+0.2	−9.9	−3.1
AM-DIS	+0.3	−3.2	–	–
AM-LOC	+0.8	+1.1	+0.6	+3.0
AM-MNR	−0.5	−1.2	+2.7	+0.3
AM-TMP	−1.2	−0.7	−1.9	+3.3

Out-of-domain	verbal		nominal	
Label	P	R	P	R
A0	−0.9	+2.5	−2.5	−0.4
A1	+1.7	+0.8	+1.0	+3.7
A2	+1.4	+0.7	−2.5	+3.2
AM-ADV	+5.6	+0.7	–	–
AM-DIS	+7.3	–	–	–
AM-LOC	+0.7	+2.4	–	+15.0
AM-MNR	+6.4	+10.5	+9.7	+10.7
AM-TMP	+1.6	+1.8	−6.7	+1.1

Table 5: Differences in precision and recall per argument label and predicate word category. All numbers represent absolute percentage points.

in Section 3. Table 6 lists examples from the out-of-domain data set that were misclassified by the **Original** model but could be correctly predicted using our enhanced feature set. As illustrated by Examples (1) and (2), the **Interaction** feature seems to help recall by guiding classification decisions towards more meaningful and complete structures.

Improvements using the **Path** feature can be observed in cases where nested syntactic structures need to be processed, as required in Example (2). In another instance, Example (3), the following path is predicted between argument and predicate: *He* $\xrightarrow{\text{SBJ}}$ *rose* $\xleftarrow{\text{COORD}}$ *and* $\xleftarrow{\text{CONJ}}$ *went* $\xleftarrow{\text{OPRD}}$ *to* $\xleftarrow{\text{IM}}$ *have*. Such cases are particularly problematic for the **Original** model because long and potentially erroneous paths are sparse in the training data.

Further gains in performance are achieved using the **Span** feature, which enables the model to better handle infrequent and out-of-vocabulary words occurring in an argument span, including "hours" and "softly" in Example (4) and (5), respectively.

7 Conclusions

In this paper, we proposed to enhance the feature space of a state-of-the-art semantic role labeller by applying and composing distributional word representations. Our results indicate that combining such features with standard syntactic cues leads to more precise and more robust models, with significant improvements both in-domain and out-of-domain. Ablation tests on an out-of-domain data set have shown that gains in recall are mostly due to features based on composed representations. Given the novelty of these features for SRL, we believe that this insight is remarkable and deserves further investigation. In future work, we plan to apply more sophisticated models of compositionality to better represent predicate-argument structures and to guide classification decisions towards outcomes that are semantically more plausible. We anticipate that this line of research will also be of interest for a range of related tasks beyond traditional SRL, including predicate-argument structure alignment (Roth and Frank, 2012) and implicit argument linking (Gerber and Chai, 2012).

Acknowledgements

This work has been supported by the FP7 Collaborative Project S-CASE (Grant Agreement No 610717) funded by the European Commission (Michael Roth), and by EPSRC (EP/K017845/1) in the framework of the CHIST-ERA READERS project (Kristian Woodsend).

References

Omri Abend, Roi Reichart, and Ari Rappoport. 2009. Unsupervised argument identification for semantic role labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 28–36, Suntec, Singapore, August.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193, Cambridge, MA, October. Association for Computational Linguistics.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 43–48. Association for Computational Linguistics.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstration Volume*, pages 33–36, Beijing, China, August. Coling 2010 Organizing Committee.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556, Jeju Island, Korea, July. Association for Computational Linguistics.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China, August.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

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

Danilo Croce, Cristina Giannone, Paolo Annesi, and Roberto Basili. 2010. Towards open-domain semantic role labeling. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 237–246, Uppsala, Sweden, July. Association for Computational Linguistics.

Koen Deschacht and Marie-Francine Moens. 2009. Semi-supervised semantic role labeling using the Latent Words Language Model. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 21–29, Singapore, August.

Paramveer Dhillon, Dean P Foster, and Lyle H Ungar. 2011. Multi-view learning of word embeddings via CCA. In *Advances in Neural Information Processing Systems*, pages 199–207.

Hagen Fürstenau and Mirella Lapata. 2012. Semi-supervised semantic role labeling via structural alignment. *Computational Linguistics*, 38(1):135–171.

Matthew Gerber and Joyce Chai. 2012. Semantic Role Labeling of Implicit Arguments for Nominal Predicates. *Computational Linguistics*, 38(4):755–798.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1448–1458, Baltimore, Maryland, June. Association for Computational Linguistics.

Fei Huang and Alexander Yates. 2010. Open-domain semantic role labeling by modeling word spans. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 968–978, Uppsala, Sweden, July.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea, July.

Rémi Lebret and Ronan Collobert. 2014. Word embeddings through hellinger pca. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 482–490, Gothenburg, Sweden, April. Association for Computational Linguistics.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The nombank project: An interim report. In A. Meyers, editor, *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, Massachusetts, USA, May.

Jeff Mitchell and Mirella Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388–1429.

Andriy Mnih and Geoffrey Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*, volume 21, pages 1081–1088.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Michael Roth and Anette Frank. 2012. Aligning predicate argument structures in monolingual comparable texts: A new corpus for a new task. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 218–227, Montreal, Canada, June.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2010. *FrameNet II: Extended Theory and Practice*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea, July. Association for Computational Linguistics.

Ivan Titov and Alexandre Klementiev. 2011. A bayesian model for unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1445–1455, Portland, Oregon, USA, June.

Ivan Titov and Alexandre Klementiev. 2012. Semi-supervised semantic role labeling: Approaching from an unsupervised perspective. In *Proceedings of COLING 2012*, pages 2635–2652, Mumbai, India, December.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden, July. Association for Computational Linguistics.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 947–953, Saarbrücken, Germany, August.

Beñat Zapirain, Eneko Agirre, Lluís Màrquez, and Mihai Surdeanu. 2013. Selectional preferences for semantic role classification. *Computational Linguistics*, 39(3):631–663.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 61–66, Boulder, Colorado, USA, June.

Automatic Domain Assignment for Word Sense Alignment

Tommaso Caselli
TrentoRISE / Via Sommarive, 18
38123 Povo, Italy
t.caselli@gmail.com

Carlo Strapparava
FBK / Via Sommarive, 18
38123 Povo, Italy
strappa@fbk.eu

Abstract

This paper reports on the development of a hybrid and simple method based on a machine learning classifier (Naive Bayes), Word Sense Disambiguation and rules, for the automatic assignment of WordNet Domains to nominal entries of a lexicographic dictionary, the Senso Comune De Mauro Lexicon. The system obtained an F1 score of 0.58, with a Precision of 0.70. We further used the automatically assigned domains to filter out word sense alignments between MultiWordNet and Senso Comune. This has led to an improvement in the quality of the sense alignments showing the validity of the approach for domain assignment and the importance of domain information for achieving good sense alignments.

1 Introduction and Problem Statement

Lexical knowledge, i.e. how words are used and express meaning, plays a key role in Natural Language Processing. Lexical knowledge is available in many different forms, ranging from unstructured terminologies (i.e. word list), to full fledged computational lexica and ontologies (e.g. WordNet (Fellbaum, 1998)). The process of creation of lexical resources is costly both in terms of money and time. To overcome these limits, semi-automatic approaches have been developed (e.g. MultiWordNet (Pianta et al., 2002)) with different levels of success. Furthermore, important information is scattered in different resources and difficult to use. Semantic interoperability between resources could represent a viable solution to allow reusability and develop more robust and powerful resources. Word sense alignment (WSA) qualifies as the preliminary requirement for achieving this goal (Matuschek and Gurevych, 2013).

WSA aims at creating lists of pairs of senses from two, or more, (lexical-semantic) resources which denote the same meaning. Different approaches to WSA have been proposed and they all share some common elements, namely: i.) the extensive use of sense descriptions of the words (e.g. WordNet glosses); and ii.) the extension of the basic sense descriptions with additional information such as hypernyms, synonyms and domain or category labels.

The purpose of this work is two folded: first, we experiment on the automatic assignment of domain labels to sense descriptions, and then, evaluate the impact of this information for improving an existing sense aligned dataset for nouns. Previous works has demonstrated that domain labels are a good feature for obtaining high quality alignments of entries (Navigli, 2006; Toral et al., 2009; Navigli and Ponzetto, 2012). The WordNet (WN) Domains (Magnini and Cavaglia, 2000; Bentivogli et al., 2004) have been selected as reference domain labels. We will use as candidate lexico-semantic resources to be aligned two Italian lexica, namely, MultiWordNet (MWN) and the Senso Comune De Mauro Lexicon (SCDM) (Vetere et al., 2011).

The two resources differ in terms of modelization: the former, MWN, is an Italian version of WN obtained through the "expand model" (Vossen, 1996) and perfectly aligned to Princeton WN 1.6, while the latter, SCDM, is a machine readable dictionary obtained from a paper-based reference lexicographic dictionary, De Mauro GRADIT. Major issues for WSA of the lexica concern the following aspects:

- SCMD has no structure of word senses (i.e. no taxonomy, no synonymy relations, no distinction between core senses and subsenses for polysemous entries) unlike MWN;
- SCDM has no domain or category labels associated to senses (with the exception of specific terminological entries) unlike MWN;
- the Italian section of MWN has only 2,481 glosses in Italian over 28,517 synsets for nouns (i.e. 8.7%).

The remainder of this paper is organized as follows: Section 2 will report on the methodology and experiments implemented for the automatic assignment of the WN Domains to the SCDM entries. Section 3 will describe the dataset used for the evaluation of the WSA experiments and the use of the WN Domains for filtering the sense alignments. Finally, Section 4 illustrates conclusion and future work.

2 Methodology and Experiments

The WN Domains consist of a set of 166 hierarchically organized labels which have been associated to each

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 414–418,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Classifiers	P	R	F1	10-Fold F1
NaiveBayes$_{lemma}$	0.77	0.58	0.66	0.66
MaxEnt$_{lemma}$	0.70	0.49	0.58	0.63
NaiveBayes$_{wsd}$	0.77	0.58	0.66	0.69
MaxEnt$_{wsd}$	0.74	0.54	0.62	0.67

Table 2: Results for the Naive Bayes and Maximum Entropy binary classifiers.

synset[1] and express a subject field label (e.g. SPORT, MEDICINE). A special label, FACTOTUM, has been used for those synsets which can appear in almost all subject fields.

The identification of a domain label to the nominal entries in the SCDM Lexicon is based the "One Domain per Discourse" (ODD) hypothesis applied to the sense descriptions. We have used a reduced set of domains labels (45 normalized domains) following (Magnini et al., 2001).

To assign the WN domain label to the SCDM entries, we have developed a hybrid method: first a binary classifier is applied to the SCDM sense descriptions to discriminate between two domain values, FACTOTUM and OTHER, where the OTHER value includes all remaining 44 normalized domains. After this, all entries classified with the OTHER value are analyzed by a rule based system and associated with a specific domain label (i.e. SPORT, MEDICINE, FOOD ...).

2.1 Classifier and feature selection

We have developed a training set by manually aligning noun senses between the two lexica. The sense alignment allows us to associate all the information of a synset to a corresponding entry in the SCDM lexicon, including the WN Domain label. Concerning the test set, we have used an existing dataset of aligned noun pairs as in (Caselli et al., 2014). We report in Table 1 the figures for the training and test sets. Multiple alignments with the same domain label have been excluded from the training set.

Characteristics	Training Set	Test Set
# lemmas	131	46
# of aligned pairs	369	166
# of SCDM senses	747	216
# of MWN synsets	675	229
# SCDM with WN Domain label	350	118

Table 1: Training and test sets for the classifier.

In order for the classifier to predict the binary domain labels (FACTOTUM and OTHER), each sense description of the SCDM Lexicon has been represented by means of a two-dimensional feature vector (e.g. for training data: BINARY_DOMAIN_LABEL GENERIC:val SPECIFIC:val). Feature values have been obtained through two strategies:

- lemma label: we extract all normalized domain labels associated to each sense of each lemma in the sense description from MWN. The value of the feature GENERIC corresponds to the sum of the FACTOTUM labels. The value of the feature SPECIFIC corresponds to the sum of all other specific domain labels (e.g. MEDICINE, SPORT etc.) after they have been collapsed into a single value (i.e. NOT-FACTOTUM).

- word sense label: for each sense description, we have first performed Word Sense Disambiguation by means of an adapted version to Italian of the UKB package[2] (Agirre et al., 2010; Agirre et al., 2014)[3]. Only the highest ranked synset, and associated WN Domain(s), was retained as good. Similarly to the lemma label strategy, the sum of the domain label FACTOTUM is assigned to the feature GENERIC, while the sum of all other domain labels collapsed into the single value NOT-FACTOTUM is assigned to the feature SPECIFIC.

We experimented with two classifiers: Naive Bayes and Maximum Entropy as implemented in the MALLET package (McCallum, 2002). We illustrate the results in Table 2. The classifiers have been evaluated with respect to standard Precision (P), Recall (R) and F1 against the test set. Ten-fold cross validation has been performed on the training set as well. Classifiers trained with the first strategy will be associated with the label *lemma*, while those trained with the second strategy with the label *wsd*.

Both classifiers obtains good results with respect to the test data in terms of Precision and Recall. The Naive Bayes classifier outperforms the Maximum Entropy one in both training approaches, suggesting better generalization capabilities even in presence of a small training set and basic features. The role of WSD has a positive impact, namely for the Maximum Entropy classifier (Precision +4 points, Recall +5 points with respect to the lemma label). Although such a positive effect of the WSD does not emerge for the Naive Bayes classifier with respect to the test set, we can still observe an improvement over the ten-fold cross validation (F1= 0.69 *vs*. F1=0.66). We finally selected the

[1]The full set of labels and hierarchy is available at http://wndomains.fbk.eu/hierarchy.html

[2]Available at http://ixa2.si.ehu.es/ukb/

[3]We used the WN Multilingual Central Repository as knowledge base and the MWN entries as dictionary

predictions of Naive Bayes$_{wsd}$ classifier as input to the rule-based system as it provides the highest scores.

2.2 Rules for WN Domain assignment

The rule based classifier for final WN Domain assignment works as follows:

- lemmatized and word sense disambiguated lemmas in the sense descriptions are associated with the corresponding WN Domains from MWN;
- frequency counts on the WN Domain labels is applied; the most frequent WN Domain is assigned as the correct WN Domain of the nominal entry;
- in case two or more WN Domains have same frequency, the following assignment strategy is applied: if the frequency scores of the WN Domains is equal to 1, the value FACTOTUM is selected; on the contrary, if the frequency score is higher than 1, all WN Domain labels are retained as good.

We report the results on final domain assignment in Table 3. The final system, NaiveBayes+Rules, has been compared to two baselines. Both baselines apply frequency counts over the WN Domains labels of the lemmas of the sense descriptions for the entire set of the 45 normalized domain values, including the FACTOTUM label, as explained in Section2. The Baseline$_{lemma}$ assigns the domain by taking into account every WN Domain associated to each lemma. On the other hand, the Baseline$_{wsd}$ selects only the WN Domain of sense disambiguated lemmas. WSD for the second baseline has been performed by applying the same method described in Section 2.1. The results of both baselines have high values for Precision (0.58 for Baseline$_{lemma}$, 0.70 for Baseline$_{wsd}$). We consider this as a further support to the validity of the ODD hypothesis which seems to hold even for text descriptions like dictionary glosses which normally use generic lexical items to illustrate word senses. It is also interesting to notice that WSD on its own has a positive impact in Baseline$_{wsd}$ system for the assignment of specific domain labels (F1=0.53).

The hybrid system performs better than both baselines in terms of F1 scores (F1=0.58 *vs.* F1=0.45 for Baseline$_{lemma}$ *vs.* F1=0.53 for Baseline$_{wsd}$). However, both the hybrid system and the Baseline$_{wsd}$ obtain the same Precision. To better evaluate the performance of our hybrid approach, we computed the paired t-test. The results of the hybrid system are statistically significant with respect to the Baseline$_{lemma}$ ($p < 0.05$) and for Recall only when compared to the Baseline$_{wsd}$.

To further analyze the difference between the hybrid system and the Baseline$_{wsd}$, we performed an error analysis on their outputs. We have identified that the hybrid system is more accurate in the prediction of the FACTOTUM class with respect to the baseline. In particular, the accuracy of the hybrid system on this class is 79% while that of the baseline is only 65%. In addition to this, the hybrid system provides better results in terms of Recall (R=0.50 *vs.* R=0.43). Although comparable, the hybrid system provides more accurate results with respect to the baseline.

System	P	R	F1
NaiveBayes$_{wsd}$+Rules	0.70†	0.50†∗	0.58†
Baseline$_{lemma}$	0.58	0.36	0.45
Baseline$_{wsd}$	0.70	0.43	0.53

Table 3: Results of WN Domain Assignment over the SDCM entries. Statistical significance of the NaiveBayes+Rules system has been marked with a † for the Baseline$_{lemma}$ and with a ∗ for the Baseline$_{wsd}$

3 Domain Filtering for WSA

This section reports on the experiments for improving existing WSA for nouns between SDCM and MWN. In this work we have used the same dataset and alignment methods as in (Caselli et al., 2014), shortly described here:

- Lexical Match: for each word *w* and for each sense *s* in the given resources $R \in \{$MWN, SCDM$\}$, we constructed a sense descriptions d_R(s) as a bag of words in Italian. The alignment is based on counting the number of overlapping tokens between the two strings, normalized by the length of the strings;
- Cosine Similarity: we used the Personalized Page Rank (PPR) algorithm (Agirre et al., 2010) with WN 3.0 as knowledge base extended with the "Princeton Annotated Gloss Corpus". Once the PPR vector pairs are obtained, the alignment is obtained on the basis of the cosine score for each pair[4].

The dataset consists of 166 pairs of aligned senses from MWN and SCDM for 46 nominal lemmas (see also column "*Test set*" in Table 1). Overall, SCDM covers 53.71The main difference with respect to (Caselli et al., 2014) is that the proposed alignments have been additionally filtered on the basis of the output of the WN domain system (NaiveBayes$_{wsd}$+Rules). In particular, for each aligned pair which was considered as good in (Caselli et al., 2014), we have applied a further filtering based on the WN domain system results as follows: if two senses are aligned but do not have the same domain, they are excluded from the WSA results, otherwise they are retained. Table 4 illustrates

[4]The vectors for the SCDM entries were obtained by, first, applying Google Translate API to get the English translations and, then, PPR over WN 3.0.

System	P	R	F1
LexicalMatch	0.76 (0.69)	0.27 (0.44)	0.40 (0.55)
Cosine_noThreshold	0.27 (0.12)	**0.47 (0.94)**	0.35 (0.21)
Cosine > 0.1	0.77 (0.52)	0.21 (0.32)	0.33 (0.40)
Cosine > 0.2	**0.87 (0.77)**	0.14 (0.21)	0.24 (0.33)
LexicalMatch+Cosine > 0.1	0.73 (na)	0.40 (na)	**0.51 (na)**
LexicalMatch+Cosine > 0.2	**0.77 (0.67)**	0.37 (0.61)	0.50 (0.64)

Table 4: Results for WSA of nouns with domain filtering.

the results of the WSA approaches with domain filters. We report in brackets the results from (Caselli et al., 2014). The filtering based on WN Domains has a big impact on Precision and contributes to increase the quality of the aligned senses. Although, in general, we have a downgrading of the performance with respect to Recall, the increase in Precision will reduce the manual post-processing effort to fully aligned the two resources[5]. Furthermore, it is interesting to notice that, when merging together the results of the pre-filtered alignments from the two alignment approaches (LexicalMatch+Cosine > 0.1 and LexicalMatch+Cosine > 0.2), we still have a very high Precision (> 0.70) and an increase in Recall (> 0.40) with respect to the results of each approach. Finally, we want to point out that what was reported as the best alignment results in (Caselli et al., 2014), namely LexicalMatch+Cosine > 0.2, can be obtained, at least for Precision, with a lower filtering cut-off threshold on the Cosine Similarity approach (i.e cut-off threshold at or higher than 0.1)

4 Conclusions and Future Work

This work describes a hybrid approach based on a Naive Bayes classifier, Word Sense Disambiguation and rules for assigning WN Domains to nominal sense descriptions of a lexicographic dictionary, the Senso Comune De Mauro Lexicon. The assignment of domain labels has been used to improve WSA results on nouns between the Senso Comune Lexicon and MultiWordNet. The results support some observations, namely: i.) domain filtering plays an important role in WSA, namely as a strategy to exclude wrong alignments (false positives) and improve the quality of the aligned pairs; ii.) the method we have proposed is a viable approach for automatically enriching existing lexical resources in a reliable way; and iii.) the ODD hypothesis also apply to sense descriptions.

An advantage of our approach is its simplicity. We have used features based on frequency counts and obtained good results, with a Precision of 0.70 for automatic WN Domain assignment. Nevertheless, an important role is played by Word Sense Disambiguation. The use of domain labels obtained from sense disambiguated lemmas improves both the results of the classifier and those of the rules. The absence of statistical significance with respect to the Baseline$_{wsd}$ is not to be considered as a negative result. As the error analysis has showed, the classifier mostly contributes to the identification of the FACTOTUM value, which tends to be overestimated even with sense disambiguated lemmas, and to Recall. We are planning to extend this work to include domain clusters to improve the domain assignment results, namely in terms of Recall.

[5]The F1 of 0.64 in (Caselli et al., 2014) is obtained with a Precision of 0.67, suggesting that some alignments are false positives

Acknowledgments

One of the author wants to thank Vrije Univerisiteit Amsterdam for sponsoring the attendance to the EMNLP conference.

References

Eneko Agirre, Montse Cuadros, German Rigau, and Aitor Soroa. 2010. Exploring knowledge bases for similarity. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

Luisa Bentivogli, Pamela Forner, Bernardo Magnini, and Emanuele Pianta. 2004. Revising the wordnet domains hierarchy: semantics, coverage and balancing. In *Proceedings of the Workshop on Multilingual Linguistic Resources*, pages 101–108. Association for Computational Linguistics.

Tommaso Caselli, Carlo Strapparava, Laure Vieu, and Guido Vetere. 2014. Aligning an italianwordnet with a lexicographic dictionary: Coping with limited data. In *Proceedings of the Seventh Global WordNet Conference*, pages 290–298.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. MIT Press.

Bernardo Magnini and Gabriela Cavaglia. 2000. Integrating subject field codes into wordnet. In *Proceedings of the conference on International Language Resources and Evaluation (LREC 2000)*.

Bernardo Magnini, Carlo Strapparava, Giovanni Pezzulo, and Alfio Gliozzo. 2001. Using domain information for word sense disambiguation. In *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 111–114. Association for Computational Linguistics.

Michael Matuschek and Iryna Gurevych. 2013. Dijkstra-wsa: A graph-based approach to word sense alignment. *Transactions of the Association for Computational Linguistics (TACL)*, 2:to appear.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit.

Rada Mihalcea. 2007. Using Wikipedia for automatic word sense disambiguation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, Rochester, New York.

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

Roberto Navigli. 2006. Meaningful clustering of senses helps boost word sense disambiguation performance. In *Proceedings of the* 44^{th} *Annual Meeting of the Association for Computational Linguistics joint with the* 21^{st} *International Conference on Computational Linguistics (COLING-ACL)*, Sydney, Australia.

Elisabeth Niemann and Iryna Gurevych. 2011. The peoples web meets linguistic knowledge: Automatic sense alignment of Wikipedia and WordNet. In *Proceedings of the 9th International Conference on Computational Semantics*, pages 205–214, Singapore, January.

Emanuele Pianta, Luisa Bentivogli, and Cristian Girardi. 2002. MultiWordNet: developing an aligned multilingual database. In *First International Conference on Global WordNet*, Mysore, India.

German Rigau and Agirre Eneko. 1995. Disambiguating bilingual nominal entries against WordNet. In *Proceedings of workshop The Computational Lexicon, 7th European Summer School in Logic, Language and Information*, Barcelona, Spain.

Adriana Roventini, Nilda Ruimy, Rita Marinelli, Marisa Ulivieri, and Michele Mammini. 2007. Mapping concrete entities from PAROLE-SIMPLE-CLIPS to ItalWordNet: Methodology and results. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, Prague, Czech Republic, June.

Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. 2005. Automatic assignment of Wikipedia encyclopedic entries to WordNet synsets. In *Proceedings of the Third international conference on Advances in Web Intelligence*, AWIC'05, Berlin, Heidelberg. Springer-Verlag.

Antonio Toral, Oscar Ferrández, Eneko Aguirre, and Rafael Munoz. 2009. A study on linking and disambiguating wikipedia categories to wordnet using text similarity. *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2009)*.

Guido Vetere, Alessandro Oltramari, Isabella Chiari, Elisabetta Jezek, Laure Vieu, and Fabio Massimo Zanzotto. 2011. Senso Comune, an open knowledge base for italian. *JTraitement Automatique des Langues*, 53(3):217–243.

Piek Vossen. 1996. Right or wrong: Combining lexical resources in the eurowordnet project. In *Euralex*, volume 96, pages 715–728. Citeseer.

Nothing like Good Old Frequency: Studying Context Filters for Distributional Thesauri

Muntsa Padró,[♣] Marco Idiart[♡], Carlos Ramisch[♢], Aline Villavicencio[♣]
[♣]Institute of Informatics, Federal University of Rio Grande do Sul (Brazil)
[♡]Institute of Physics, Federal University of Rio Grande do Sul (Brazil)
[♢]Aix Marseille Université, CNRS, LIF UMR 7279, 13288, Marseille (France)
muntsa.padro@inf.ufrgs.br, marco.idiart@gmail.com,
carlos.ramisch@lif.univ-mrs.fr, avillavicencio@inf.ufrgs.br

Abstract

Much attention has been given to the impact of informativeness and similarity measures on distributional thesauri. We investigate the effects of context filters on thesaurus quality and propose the use of cooccurrence frequency as a simple and inexpensive criterion. For evaluation, we measure thesaurus agreement with WordNet and performance in answering TOEFL-like questions. Results illustrate the sensitivity of distributional thesauri to filters.

1 Introduction

Large-scale *distributional thesauri* created automatically from corpora (Grefenstette, 1994; Lin, 1998; Weeds et al., 2004; Ferret, 2012) are an inexpensive and fast alternative for representing semantic relatedness between words, when manually constructed resources like WordNet (Fellbaum, 1998) are unavailable or lack coverage. To construct a distributional thesaurus, the (collocational or syntactic) contexts in which a target word occurs are used as the basis for calculating its similarity with other words. That is, two words are similar if they share a large proportion of contexts.

Much attention has been devoted to refining thesaurus quality, improving informativeness and similarity measures (Lin, 1998; Curran and Moens, 2002; Ferret, 2010), identifying and demoting bad neighbors (Ferret, 2013), or using more relevant contexts (Broda et al., 2009; Biemann and Riedl, 2013). For the latter in particular, as words vary in their collocational tendencies, it is difficult to determine how informative a given context is. To remove uninformative and noisy contexts, filters have often been applied like pointwise mutual information (PMI), lexicographer's mutual information (LMI) (Biemann and Riedl, 2013), t-score (Piasecki et al., 2007) and z-score (Broda et al., 2009). However, the selection of a measure and of a threshold value for these filters is generally empirically determined. We argue that these filtering parameters have a great influence on the quality of the generated thesauri.

The goal of this paper is to quantify the impact of context filters on distributional thesauri. We experiment with different filter methods and measures to assess context significance. We propose the use of simple cooccurrence frequency as a filter and show that it leads to better results than more expensive measures such as LMI or PMI. Thus we propose a cheap and effective way of filtering contexts while maintaining quality.

This paper is organized as follows: in §2 we discuss evaluation of distributional thesauri. The methodology adopted in the work and the results are discussed in §3 and §4. We finish with some conclusions and discussion of future work.

2 Related Work

In a nutshell, the standard approach to build a distributional thesaurus consists of: (i) the *extraction of contexts* for the target words from corpora, (ii) the application of an *informativeness measure* to represent these contexts and (iii) the application of a *similarity measure* to compare sets of contexts. The contexts in which a target word appears can be extracted in terms of a window of cooccurring (content) words surrounding the target (Freitag et al., 2005; Ferret, 2012; Erk and Pado, 2010) or in terms of the syntactic dependencies in which the target appears (Lin, 1998; McCarthy et al., 2003; Weeds et al., 2004). The informativeness of each context is calculated using measures like PMI, and t-test while the similarity between contexts is calculated using measures like Lin's (1998), cosine, Jensen-Shannon divergence, Dice or Jaccard.

Evaluation of the quality of distributional thesauri is a well know problem in the area (Lin,

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 419–424,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

1998; Curran and Moens, 2002). For instance, for intrinsic evaluation, the agreement between thesauri has been examined, looking at the average similarity of a word in the thesauri (Lin, 1998), and at the overlap and rank agreement between the thesauri for target words like nouns (Weeds et al., 2004). Although much attention has been given to the evaluation of various informativeness and similarity measures, a careful assessment of the effects of filtering on the resulting thesauri is also needed. For instance, Biemann and Riedl (2013) found that filtering a subset of contexts based on LMI increased the similarity of a thesaurus with WordNet. In this work, we compare the impact of using different types of filters in terms of thesaurus agreement with WordNet, focusing on a distributional thesaurus of English verbs. We also propose a frequency-based saliency measure to rank and filter contexts and compare it with PMI and LMI.

Extrinsic evaluation of distributional thesauri has been carried out for tasks such as English lexical substitution (McCarthy and Navigli, 2009), phrasal verb compositionality detection (McCarthy et al., 2003) and the WordNet-based synonymy test (WBST) (Freitag et al., 2005). For comparative purposes in this work we adopt the latter.

3 Methodology

We focus on thesauri of *English verbs* constructed from the BNC (Burnard, 2007)[1]. Contexts are extracted from syntactic dependencies generated by RASP (Briscoe et al., 2006), using nouns (heads of NPs) which have subject and direct object relations with the target verb. Thus, each target verb is represented by a set of *triples* containing (i) the verb itself, (ii) a context noun and (iii) a syntactic relation (object, subject). The thesauri were constructed using Lin's (1998) method. Lin's version of the distributional hypothesis states that two words (verbs v_1 and v_2 in our case) are similar if they share a large proportion of contexts weighted by their information content, assessed with PMI (Bansal et al., 2012; Turney, 2013).

In the literature, little attention is paid to context filters. To investigate their impact, we compare two kinds of filters, and before calculating similarity using Lin's measure, we apply them to remove potentially noisy triples:

- **Threshold (th):** we remove triples that occur less than a threshold th. Threshold values vary from 1 to 50 counts per triple.
- **Relevance (p):** we keep only the top p most relevant contexts for each verb, were relevance is defined according to the following measures: (a) frequency, (b) PMI, and (c) LMI (Biemann and Riedl, 2013). Values of p vary between 10 and 1000.

In this work, we want to answer two questions: (a) Do more selective filters improve *intrinsic evaluation* of thesaurus? and (b) Do they also help in *extrinsic evaluation*?

For *intrinsic evaluation*, we determine agreement between a distributional thesaurus and WordNet as the path similarities for the first k distributional neighbors of a verb. A single score is obtained by averaging the similarities of all verbs with their k first neighbors. The higher this score is, the closer the neighbors are to the target in WordNet, and the better the thesaurus. Several values of k were tested and the results showed exactly the same curve shapes for all values, with WordNet similarity decreasing linearly with k. For the remainder of the paper we adopt $k = 10$, as it is widely used in the literature.

For *extrinsic evaluation*, we use the WBST set for verbs (Freitag et al., 2005) with 7,398 questions and an average polysemy of 10.4. The task consists of choosing the most suitable synonym for a word among a set of four options. The thesaurus is used to rank the candidate answers by similarity scores, and select the first one as the correct synonym. As discussed by Freitag et al. (2005), the upper bound reached by English native speakers is 88.4% accuracy, and simple lower bounds are 25% (random choice) and 34.5% (always choosing the most frequent option).

4 Results

Figure 1 shows average WordNet similarities for thesauri built filtering by frequency threshold th and by p most frequent contexts. Table 1 summarizes the parametrization leading to the best WordNet similarity for each kind of filter. In all cases we show the results obtained for different frequency ranges[2] as well as the results when averaging over all verbs.

[1] Even though larger corpora are available, we use a traditional carefully constructed corpus with representative samples of written English to control the quality of the thesaurus.

[2] In order to study the influence of verb frequency on the results, we divide the verbs in three groups: high-frequency

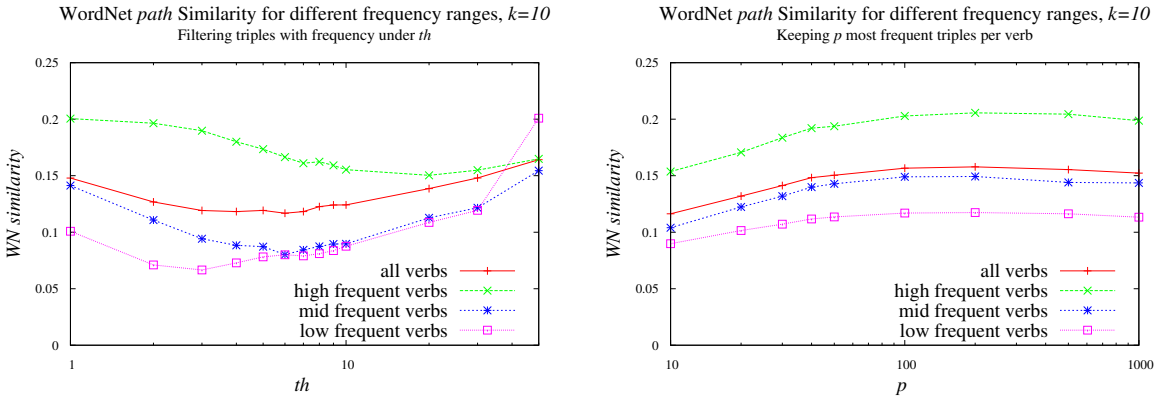


Figure 1: WordNet scores for verb frequency ranges, filtering by frequency threshold th (left) and p most frequent contexts (right).

Filter	All verbs		Frequency range Low		Mid		High	
No filter	-	0.148	-	0.101	-	0.144	-	0.198
Filter low freq. contexts	$th = 50$	**0.164**	$th = 50$	0.202	$th = 50$	0.154	$th = 1$	0.200
Keep p contexts (freq.)	$p = 200$	**0.158**	$p = 500$	0.138	$p = 200$	0.149	$p = 200$	0.206
Keep p contexts (PMI)	$p = 1000$	0.139	$p = 1000$	0.101	$p = 1000$	0.136	$p = 1000$	0.181
Keep p contexts (LMI)	$p = 200$	0.155	$p = 100$	0.112	$p = 200$	0.147	$p = 200$	0.208

Table 1: Best scores obtained for each filter for all verbs and frequency ranges. Scores are given in terms of WordNet path. Confidence interval is arround $\pm$ 0.002 in all cases.

When using a threshold filter (Figure 1 left), high values lead to better performance for mid- and low-frequency verbs. This is because, for high th values, there are few low and mid-frequency verbs left, since a verb that occurs less has less chances to be seen often in the same context. The similarity for verbs with no contexts over the frequency threshold cannot be assessed and as a consequence those verbs are not included in the final thesaurus. As Figure 2 shows, the number of verbs decreases much faster for low and mid frequency verbs when th increases.[3] For example, for $th = 50$, there are only 7 remaining low-frequency verbs in the thesaurus and these tend to be idiosyncratic multiword expressions. One example is *wreak*, and the only triple containing this verb that appeared more than 50 times is *wreak havoc* (71 occurrences). The neighbors of this verb are *cause* and *play*, which yield a good similarity score in WordNet. Therefore, although higher thresholds result in higher similarities for low and mid-frequency verbs, this comes at a cost, as the number of verbs included in the thesaurus decreases considerably.

($||v|| \geq 500$), mid-frequency ($150 \leq ||v|| < 500$) and low-frequency ($||v|| < 150$).

[3]For p most salient contexts, the number of verbs does not vary and is the same shown in Figure 2 for $th = 1$ (no filter).

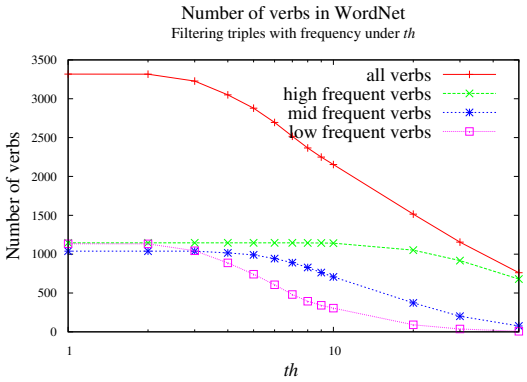


Figure 2: Number of verbs per frequency ranges when filtering by context frequency threshold th

As expected, the best performance is obtained for high-frequency verbs and no filter, since it results in more context information per verb. Increasing th decreases similarity due to the removal of some of these contexts. In average, higher th values lead to better overall similarity among the frequency ranges (from 0.148 with $th = 1$ to 0.164 with $th = 50$). The higher the threshold, the more high-frequency verbs will prevail in the thesauri, for which the WordNet path similarities are higher.

On the other hand, when adopting a relevance

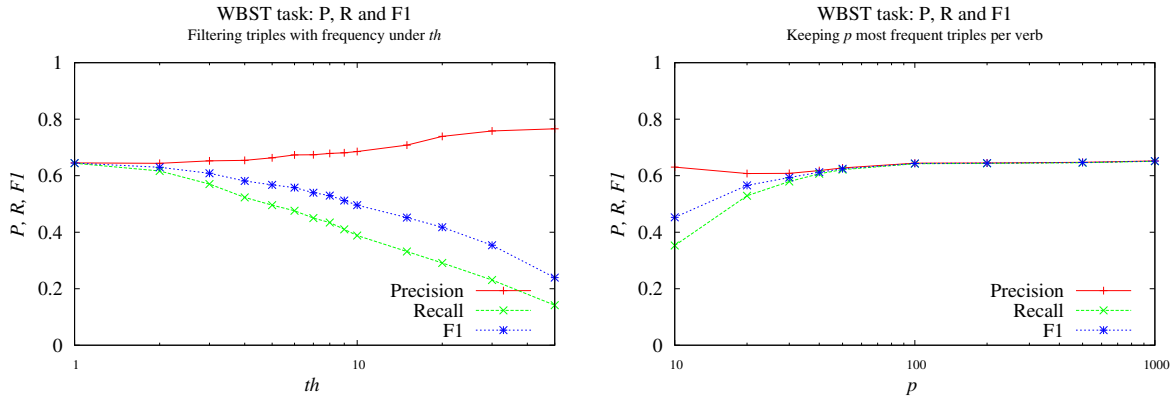


Figure 3: WBST task scores filtering by frequency threshold th (left) and p most frequent contexts (right).

filter of keeping the p most relevant contexts for each verb (Figure 1 right), we obtain similar results, but more stable thesauri. The number of verbs remains constant, since we keep a fixed number of contexts for each verb and verbs are not removed when the threshold is modified. WordNet similarity increases as more contexts are taken into account, for all frequency ranges. There is a maximum around $p = 200$, though larger values do not lead to a drastic drop in quality. This suggests that the noise introduced by low-frequency contexts is compensated by the increase of informativeness for other contexts. An ideal balance is reached by the lowest possible p that maintains high WordNet similarity, since the lower the p the faster the thesaurus construction.

In terms of saliency measure, when keeping only the p most relevant contexts, sorting them with PMI leads to much worse results than LMI or frequency, as PMI gives too much weight to infrequent combinations. This is consistent with results of Biemann and Riedl (2013). Regarding LMI versus frequency, the results using the latter are slightly better (or with no significant difference, depending on the frequency range). The advantage of using frequency instead of LMI is that it makes the process simpler and faster while leading to equal or better performance in all frequency ranges. Therefore for the extrinsic evaluation using WBST task, we use frequency to select the p most relevant contexts and then compute Lin's similarity using only those contexts.

Figure 3 shows the performance of the thesauri in the WBST task in terms of precision, recall and F1.[4] For precision, the best filter is to remove contexts occurring less than th times, but, this also leads to poor recall, since many verbs are left out of the thesauri and their WSBT questions cannot be answered. On the other hand, keeping the most relevant p contexts leads to more stable results and when p is high (right plot), they are similar to those shown in the left plot of Figure 3.

4.1 Discussion

The answer to our questions in Section 3 is *yes*, more selective filters improve intrinsic and extrinsic thesaurus quality. The use of both filtering methods results in thesauri in which the neighbors of target verbs are closer in WordNet and get better scores in TOEFL-like tests. However, the fact that filtering contexts with frequency under th removes verbs in the final thesaurus is a drawback, as highlighted in the extrinsic evaluation on the WBST task.

Furthermore, we demonstrated that competitive results can be obtained keeping only the p most relevant contexts per verb. On the one hand, this method leads to much more stable thesauri, with the same verbs for all values of p. On the other hand, it is important to highlight that the best results to assess the relevance of the contexts are obtained using frequency while more sophisticated filters such as LMI do not improve thesaurus quality. Although an LMI filter is relatively fast compared to dimensionality reduction techniques such as singular value decomposition (Landauer and Dumais, 1997), it is still considerably more expensive than a simple frequency filter.

In short, our experiments indicate that a reason-

[4]Filters based on LMI and PMI were also tested with the same results as intrinsic evaluation: sorting contexts by frequency leads to better results.

able trade-off between noise, coverage and computational efficiency is obtained for $p = 200$ most frequent contexts, as confirmed by intrinsic and extrinsic evaluation. Frequency threshold th is not recommended: it degrades recall because the contexts for many verbs are not frequent enough. This result is useful for extracting distributional thesauri from very large corpora like the UKWaC (Ferraresi et al., 2008) by proposing an alternative that minimizes the required computational resources while efficiently removing a significant amount of noise.

5 Conclusions and Future Work

In this paper we addressed the impact of filters on the quality of distributional thesauri, evaluating a set of standard thesauri and different filtering methods. The results suggest that the use of filters and their parameters greatly affect the thesauri generated. We show that it is better to use a filter that selects the most relevant contexts for a verb than to simply remove rare contexts. Furthermore, the best performance was obtained with the simplest method: frequency was found to be a simple and inexpensive measure of context salience. This is especially important when dealing with large amounts of data, since computing LMI for all contexts would be computationally costly. With our proposal to keep just the p most frequent contexts per verb, a great deal of contexts are cheaply removed and thus the computational power required for assessing similarity is drastically reduced.

As future work, we plan to use these filters to build thesauri from larger corpora. We would like to generalize our findings to other syntactic configurations (e.g. noun-adjective) as well as to other similarity and informativeness measures. For instance, ongoing experiments indicate that the same parameters apply when Lin's similarity is replaced by cosine. Finally, we would like to compare the proposed heuristics with more sophisticated filtering strategies like singular value decomposition (Landauer and Dumais, 1997) and non-negative matrix factorization (Van de Cruys, 2009).

Acknowledgments

We would like to thank the support of projects CAPES/COFECUB 707/11, PNPD 2484/2009, FAPERGS-INRIA 1706-2551/13-7, CNPq 312184/2012-3, 551964/2011-1, 482520/2012-4 and 312077/2012-2.

References

Mohit Bansal, John DeNero, and Dekang Lin. 2012. Unsupervised translation sense clustering. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 773–782, Montréal, Canada, June. Association for Computational Linguistics.

Chris Biemann and Martin Riedl. 2013. Text: Now in 2D! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1).

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In James Curran, editor, *Proc. of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 77–80, Sidney, Australia, Jul. ACL.

Bartosz Broda, Maciej Piasecki, and Stan Szpakowicz. 2009. Rank-based transformation in measuring semantic relatedness. In *Proceedings of the 22nd Canadian Conference on Artificial Intelligence: Advances in Artificial Intelligence*, Canadian AI '09, pages 187–190, Berlin, Heidelberg. Springer-Verlag.

Lou Burnard. 2007. User Reference Guide for the British National Corpus. Technical report, Oxford University Computing Services, Feb.

James R. Curran and Marc Moens. 2002. Improvements in automatic thesaurus extraction. In *Proc.of the ACL 2002 Workshop on Unsupervised Lexical Acquisition*, pages 59–66, Philadelphia, Pennsylvania, USA. ACL.

Katrin Erk and Sebastian Pado. 2010. Exemplar-based models for word meaning in context. In *Proc. of the ACL 2010 Conference Short Papers*, pages 92–97, Uppsala, Sweden, Jun. ACL.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. MIT Press, May. 423 p.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating UKWaC, a very large web-derived corpus of English. In *In Proceedings of the 4th Web as Corpus Workshop (WAC-4*.

Olivier Ferret. 2010. Testing semantic similarity measures for extracting synonyms from a corpus. In *Proc. of the Seventh LREC (LREC 2010)*, pages 3338–3343, Valetta, Malta, May. ELRA.

Olivier Ferret. 2012. Combining bootstrapping and feature selection for improving a distributional thesaurus. In *ECAI*, pages 336–341.

Olivier Ferret. 2013. Identifying bad semantic neighbors for improving distributional thesauri. In *Proc. of the 51st ACL (Volume 1: Long Papers)*, pages 561–571, Sofia, Bulgaria, Aug. ACL.

Dayne Freitag, Matthias Blume, John Byrnes, Edmond Chow, Sadik Kapadia, Richard Rohwer, and Zhiqiang Wang. 2005. New experiments in distributional representations of synonymy. In Ido Dagan and Dan Gildea, editors, *Proc. of the Ninth CoNLL (CoNLL-2005)*, pages 25–32, University of Michigan, MI, USA, Jun. ACL.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Springer, Norwell, MA, USA.

Thomas K Landauer and Susan T. Dumais. 1997. A solution to platos problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, pages 211–240.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proc. of the 36th ACL and 17th COLING, Volume 2*, pages 768–774, Montreal, Quebec, Canada, Aug. ACL.

Diana McCarthy and Roberto Navigli. 2009. The english lexical substitution task. *Language Resources and Evaluation*, 43(2):139–159.

Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In Francis Bond, Anna Korhonen, Diana McCarthy, and Aline Villavicencio, editors, *Proc. of the ACL Workshop on MWEs: Analysis, Acquisition and Treatment (MWE 2003)*, pages 73–80, Sapporo, Japan, Jul. ACL.

Maciej Piasecki, Stanislaw Szpakowicz, and Bartosz Broda. 2007. Automatic selection of heterogeneous syntactic features in semantic similarity of polish nouns. In *Proceedings of the 10th international conference on Text, speech and dialogue*, TSD'07, pages 99–106, Berlin, Heidelberg. Springer-Verlag.

Peter D. Turney. 2013. Distributional semantics beyond words: Supervised learning of analogy and paraphrase. 1:353–366.

Tim Van de Cruys. 2009. A non-negative tensor factorization model for selectional preference induction. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 83–90, Athens, Greece, March. Association for Computational Linguistics.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proc. of the 20th COLING (COLING 2004)*, pages 1015–1021, Geneva, Switzerland, Aug. ICCL.

Aligning English Strings with Abstract Meaning Representation Graphs

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, Kevin Knight
Information Sciences Institute
Department of Computer Science
University of Southern California
{damghani,yanggao,ulf,knight}@isi.edu

Abstract

We align pairs of English sentences and corresponding Abstract Meaning Representations (AMR), at the token level. Such alignments will be useful for downstream extraction of semantic interpretation and generation rules. Our method involves linearizing AMR structures and performing symmetrized EM training. We obtain 86.5% and 83.1% alignment F score on development and test sets.

1 Introduction

Banarescu et al. (2013) describe a semantics bank of English sentences paired with their logical meanings, written in Abstract Meaning Representation (AMR). The designers of AMR leave open the question of how meanings are derived from English sentences (and vice-versa), so there are no manually-annotated alignment links between English words and AMR concepts. This paper studies how to build such links automatically, using co-occurrence and other information. Automatic alignments may be useful for downstream extraction of semantic interpretation and generation rules.

AMRs are directed, acyclic graphs with labeled edges, e.g., the sentence *The boy wants to go* is represented as:

```
(w / want-01
  :arg0 (b / boy)
  :arg1 (g / go-01
           :arg0 b))
```

We have hand-aligned a subset of the 13,050 available AMR/English pairs. We evaluate our automatic alignments against this gold standard. A sample hand-aligned AMR is here ("~n" specifies a link to the nth English word):

```
the boy wants to go
(w / want-01~3
  :arg0 (b / boy~2)
  :arg1 (g / go-01~5
           :arg0 b))
```

This alignment problem resembles that of statistical machine translation (SMT). It is easier in some ways, because AMR and English are highly cognate. It is harder in other ways, as AMR is graph-structured, and children of an AMR node are unordered. There are also fewer available training pairs than in SMT.

One approach is to define a generative model from AMR graphs to strings. We can then use EM to uncover hidden derivations, which alignments weakly reflect. This approach is used in string/string SMT (Brown et al., 1993). However, we do not yet have such a generative graph-to-string model, and even if we did, there might not be an efficient EM solution. For example, in syntax-based SMT systems (Galley et al., 2004), the generative tree/string transduction story is clear, but in the absence of alignment constraints, there are too many derivations and rules for EM to efficiently consider.

We therefore follow syntax-based SMT custom and use string/string alignment models in aligning our graph/string pairs. However, while it is straightforward to convert syntax trees into strings data (by taking yields), it is not obvious how to do this for unordered AMR graph elements. The example above also shows that gold alignment links reach into the internal nodes of AMR.

Prior SMT work (Jones et al., 2012) describes alignment of semantic graphs and strings, though their experiments are limited to the GeoQuery domain, and their methods are not described in detail. Flanigan et al (2014) describe a heuristic AMR/English aligner. While heuristic aligners can achieve good accuracy, they will not automatically improve as more AMR/English data comes

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 425–429,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

online.

The contributions of this paper are:

- A set of gold, manually-aligned AMR/English pairs.
- An algorithm for automatically aligning AMR/English pairs.
- An empirical study establishing alignment accuracy of 86.5% and 83.1% F score for development and test sets respectively.

2 Method

We divide the description of our method into three parts: preprocessing, training, and postprocessing. In the preprocessing phase, we linearize the AMR graphs to change them into strings, clean both the AMR and English sides by removing stop words and simple stemming, and add a set of corresponding AMR/English token pairs to the corpus to help the training phase. The training phase is based on IBM models, but we modify the learning algorithm to learn the parameters symmetrically. Finally, in the postprocessing stage we rebuild the aligned AMR graph. These components are described in more detail below.

2.1 Preprocessing

The first step of the preprocessing component is to linearize the AMR structure into a string. In this step we record the original structure of nodes in the graph for later reconstruction of AMR. AMR has a rooted graph structure. To linearize this graph we run a depth first search from the root and print each node as soon as it it visited. We print but not expand the nodes that are seen previously. For example the AMR:

```
(w / want-01
  :arg0 (b / boy)
  :arg1 (g / go-01
           :arg0 b))
```

is linearized into this order: *w / want-01 :arg0 b / boy :arg1 g / go-01 :arg0 b*.

Note that semantically related nodes often stay close together after linearization.

After linearizing the AMR graph into a string, we perform a series of preprocessing steps including lowercasing the letters, removing stop words, and stemming.

The AMR and English stop word lists are generated based on our knowledge of AMR design. We know that tokens like *an*, *the* or *to be* verbs will very rarely align to any AMR token; similarly, AMR role tokens like *:arg0*, *:quant*, *:opt1* etc. as well as the *instance-of* token /, and tokens like *temporal-quantity* or *date-entity* rarely align to any English token. We remove these tokens from the parallel corpus, but remember their position to be able to convert the resulting string/string alignment back into a full AMR graph/English string alignment. Although some stopwords participate in gold alignments, by removing them we will buy a large precision gain for some recall cost.

We remove the word sense indicator and quotation marks for AMR concepts. For instance we will change *want-01* to *want* and *"ohio"* to *ohio*. Then we stem AMR and English tokens into their first four letters, except for role tokens in AMR. The purpose of stemming is to normalize English morphological variants so that they are easier to match to AMR tokens. For example English tokens *wants*, *wanting*, *wanted*, and *want* as well as the AMR token *want-01* will all convert to *want* after removing the AMR word sense indicator and stemming.

In the last step of preprocessing, we benefit from the fact that AMR concepts and their corresponding English ones are frequently cognates. Hence, after stemming, an AMR token often can be translated to a token spelled similarly in English. This is the case for English token *want* and AMR token *want* in the previous paragraph. To help the training model learn from this fact, we extend our sentence pair corpus with the set of AMR/English token pairs that are spelled identically after preprocessing. Also, for English tokens that can be translated into multiple AMR tokens, like *higher* and *high :degree more* we add the corresponding string/string pairs to the corpus. This set is extracted from existing lexical resources, including lists of comparative/superlative adjectives, negative words, etc.

After preprocessing, the AMR at the start of this section will change into: *want boy go* and the sentence *The boy wants to go* changes into *boy want to go*, and we will also add the identity pairs *want/want*, *boy/boy*, and *go/go* to the corpus.

2.2 Training

Our training method is based on IBM word alignment models (Brown et al., 1993). We modify the objective functions of the IBM models to en-

courage agreement between learning parameters in English-to-AMR and AMR-to-English directions of EM. The solution of this objective function can be approximated in an extremely simple way that requires almost no extra coding effort.

Assume that we have a set of sentence pairs $\{(E, A)\}$, where each E is an English sentence and each A is a linearized AMR. According to IBM models, A is generated from E through a generative story based on some parameters.

For example, in IBM Model 2, given E we first decide the length of A based on some probability $l = p(len(A)|len(E))$, then we decide the distortions based on a distortion table: $d = p(i|j, len(A), len(E))$. Finally, we translate English tokens into AMR ones based on a translation table $t = p(a|e)$ where a and e are AMR and English tokens respectively.

IBM models estimate these parameters to maximize the conditional likelihood of the data: $\theta_{A|E} = argmax L_{\theta_{A|E}}(A|E)$ or $\theta_{E|A} = argmax L_{\theta_{E|A}}(E|A)$ where θ denotes the set of parameters. The conditional likelihood is intrinsic to the generative story of IBM models. However, word alignment is a symmetric problem. Hence it is more reasonable to estimate the parameters in a more symmetric manner.

Our objective function in the training phase is:

$$\theta_{A|E}, \theta_{E|A} = argmax L_{\theta_{A|E}}(A|E) + L_{\theta_{E|A}}(E|A)$$
$$\text{subject to} \quad \theta_{A|E}\theta_E = \theta_{E|A}\theta_A = \theta_{A,E}$$

We approximate the solution of this objective function with almost no change to the existing implementation of the IBM models. We relax the constraint to $\theta_{A|E} = \theta_{E|A}$, then apply the following iterative process:

1. Optimize the first part of the objective function: $\theta_{A|E} = argmax L_{\theta_{A|E}}(A|E)$ using EM
2. Satisfy the constraint: set $\theta_{E|A} \propto \theta_{A|E}$
3. Optimize the second part of the objective function: $\theta_{E|A} = argmax L_{\theta_{E|A}}(E|A)$ using EM
4. Satisfy the constraint: set $\theta_{A|E} \propto \theta_{E|A}$
5. Iterate

Note that steps 1 and 3 are nothing more than running the IBM models, and steps 2 and 4 are just initialization of the EM parameters, using tables from the previous iteration. The initialization steps only make sense for the parameters that involve both sides of the alignment (i.e., the translation table and the distortion table). For the translation table we set $t_{E|A}(e|a) = t_{A|E}(a|e)$ for English and AMR tokens e and a and then normalize the t table. The distortion table can also be initialized in a similar manner. We initialize the fertility table with its value in the previous iteration.

Previously Liang et al. (2006) also presented a symmetric method for training alignment parameters. Similar to our work, their objective function involves summation of conditional likelihoods in both directions; however, their constraint is on agreement between predicted alignments while we directly focus on agreement between the parameters themselves. Moreover their method involves a modification of the E step of EM algorithm which is very hard to implement for IBM Model 3 and above.

After learning the parameters, alignments are computed using the Viterbi algorithm in both directions of the IBM models. We tried merging the alignments of the two directions using methods like grow-diag-final heuristic or taking intersection of the alignments and adding some high probability links in their union. But these methods did not help the alignment accuracy.

2.3 Postprocessing

The main goal of the postprocessing component is to rebuild the aligned AMR graph. We first insert words removed as stop words into their positions, then rebuild the graph using the recorded original structure of the nodes in the AMR graph.

We also apply a last modification to the alignments in the postprocessing. Observing that pairs like *worker* and *person :arg0-of work-01* appear frequently, and in all such cases, all the AMR tokens align to the English one, whenever we see any of AMR tokens *person*, *product*, *thing* or *company* is followed by *arg0-of*, *arg1-of* or *arg2-of* followed by an AMR concept, we align the two former tokens to what the concept is aligned to.

3 Experiments

3.1 Data Description

Our data consists of 13,050 publicly available AMR/English sentence pairs[1]. We have hand

[1]LDC AMR release 1.0, Release date: June 16, 2014 https://catalog.ldc.upenn.edu/LDC2014T12

aligned 200 of these pairs to be used as development and test sets[2]. We train the parameters on the whole data. Table 1 presents a description of the data. We do not count parenthesis, slash and AMR variables as AMR tokens. Role tokens are those AMR tokens that start with a colon. They do not represent any concept, but provide a link between concepts. For example in:

```
(w / want-01
  :arg0 (b / boy)
  :arg1 (g / go-01
           :arg0 b))
```

the first *:arg0* states that the first argument of the concept *wanting* is the *boy* and the second argument is *going*.

	train	dev	test
Sent. pairs	13050	100	100
AMR tokens	465 K	3.8 K (52%)	2.3 K (%55)
AMR role tokens	226 K	1.9 K (23%)	1.1 K (%22)
ENG tokens	248 K	2.3 K (76%)	1.7 K (%74)

Table 1: AMR/English corpus. The number in parentheses is the percent of the tokens aligned in gold annotation. Almost half of AMR tokens are role tokens, and less than a quarter of role tokens are aligned.

3.2 Experiment Results

We use MGIZA++ (Gao and Vogel, 2008) as the implementation of the IBM models. We run Model 1 and HMM for 5 iterations each, then run our training algorithm on Model 4 for 4 iterations, at which point the alignments become stable. As alignments are usually many to one from AMR to English, we compute the alignments from AMR to English in the final step.

Table 2 shows the alignment accuracy for Model 1, HMM, Model 4, and Model 4 plus the modification described in section 2.2 (Model 4+).

The alignment accuracy on the test set is lower than the development set mainly because it is intrinsically a harder set, as we only made small modifications to the system based on the development set. Recall error due to stop words is one difference.

[2]The development and test AMR/English pairs can be found in /data/split/dev/amr-release-1.0-dev-consensus.txt and /data/split/test/amr-release-1.0-test-consensus.txt, respectively. The gold alignments are not included in these files but are available separately.

	model	precision	recall	F score
Dev	Model 1	70.9	71.1	71.0
	HMM	87.6	80.1	83.7
	Model 4	89.7	80.4	84.8
	Model 4+	94.1	80.0	86.5
Test	Model 1	74.8	71.8	73.2
	HMM	83.8	73.8	78.5
	Model 4	85.8	74.9	80.0
	Model 4+	92.4	75.6	83.1

Table 2: Results on different models. Our training method (Model 4+) increases the F score by 1.7 and 3.1 points on dev and test sets respectively.

Table 3 breaks down precision, recall, and F score for role and non-role AMR tokens, and also shows in parentheses the amount of recall error that was caused by removing either side of the alignment as a stop word.

	token type	precision	recall	F score
Dev	role	77.1	48.7	59.7
	non-role	97.2	88.2	92.5
	all	94.1	80.0 (34%)	86.5
Test	role	71.0	37.8	49.3
	non-role	95.5	84.7	89.8
	all	92.4	75.6 (36%)	83.1

Table 3: Results breakdown into role and non-role AMR tokens. The numbers in the parentheses show the percent of recall errors caused by removing aligned tokens as stop words.

While the alignment method works very well on non-role tokens, it works poorly on the role tokens. Role tokens are sometimes matched with a word or part of a word in the English sentence. For example *:polarity* is matched with the *un* part of the word *unpopular*, *:manner* is matched with most adverbs, or even in the pair:

```
thanks
(t / thank-01
    :arg0 (i / i)
    :arg1 (y / you))
```

all AMR tokens including *:arg0* and *:arg1* are matched to the only English word *thanks*. Inconsistency in aligning role tokens has made this a hard problem even for human experts.

4 Conclusions and Future Work

In this paper we present the first set of manually aligned English/AMR pairs, as well as the first published system for learning the alignments between English sentences and AMR graphs that provides a strong baseline for future research in this area. As the proposed system learns the alignments automatically using very little domain knowledge, it can be applied in any domain and for any language with minor adaptations.

Computing the alignments between English sentences and AMR graphs is a first step for extraction of semantic interpretation and generation rules. Hence, a natural extension to this work will be automatically parsing English sentences into AMR and generating English sentences from AMR.

Acknowledgments

This work was supported by DARPA contracts HR0011-12-C-0014 and FA-8750-13-2-0045. The authors would like to thank David Chiang, Tomer Levinboim, and Ashish Vaswani (in no particular order) for their comments and suggestions.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Linguistic Annotation Workshop (LAW VII-ID), ACL.*

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

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *ACL.*

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *HLT-NAACL.*

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing Workshop, ACL.*

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *COLING.*

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *HLT-NAACL.*

A Shortest-path Method for Arc-factored Semantic Role Labeling

Xavier Lluís
TALP Research Center
Universitat Politècnica de Catalunya
xlluis@cs.upc.edu

Xavier Carreras
Xerox Research Centre Europe
xavier.carreras@xrce.xerox.com

Lluís Màrquez
ALT Research Group
Qatar Computing Research Institute
lmarquez@qf.org.qa

Abstract

We introduce a Semantic Role Labeling (SRL) parser that finds semantic roles for a predicate together with the syntactic paths linking predicates and arguments. Our main contribution is to formulate SRL in terms of shortest-path inference, on the assumption that the SRL model is restricted to arc-factored features of the syntactic paths behind semantic roles. Overall, our method for SRL is a novel way to exploit larger variability in the syntactic realizations of predicate-argument relations, moving away from pipeline architectures. Experiments show that our approach improves the robustness of the predictions, producing arc-factored models that perform closely to methods using unrestricted features from the syntax.

1 Introduction

Semantic role labeling (SRL) consists of finding the arguments of a predicate and labeling them with semantic roles (Gildea and Jurafsky, 2002; Màrquez et al., 2008). The arguments fill roles that answer questions of the type "who" did "what" to "whom", "how", and "why" for a given sentence predicate. Most approaches to SRL are based on a pipeline strategy, first parsing the sentence to obtain a syntactic tree and then identifying and classifying arguments (Gildea and Jurafsky, 2002; Carreras and Màrquez, 2005).

SRL methods critically depend on features of the syntactic structure, and consequently parsing mistakes can harm the quality of semantic role predictions (Gildea and Palmer, 2002). To alleviate this dependence, previous work has explored k-best parsers (Johansson and Nugues, 2008), combination systems (Surdeanu et al., 2007) or joint syntactic-semantic models (Johansson, 2009; Henderson et al., 2008; Lluís et al., 2013).

In this paper we take a different approach. In our scenario SRL is the end goal, and we assume that syntactic parsing is only an intermediate step to extract features to support SRL predictions. In this setting we define a model that, given a predicate, identifies each of the semantic roles together with the syntactic path that links the predicate with the argument. Thus, following previous work (Moschitti, 2004; Johansson, 2009), we take the syntactic path as the main source of syntactic features, but instead of just conditioning on it, we predict it together with the semantic role. The main contribution of this paper is a formulation of SRL parsing in terms of efficient shortest-path inference, under the assumption that the SRL model is restricted to arc-factored features of the syntactic path linking the argument with the predicate.

Our assumption —that features of an SRL model should factor over dependency arcs— is supported by some empirical frequencies. Table 1 shows the most frequent path patterns on CoNLL-2009 (Hajič et al., 2009) data for several languages, where a path pattern is a sequence of ascending arcs from the predicate to some ancestor, followed by descending arcs to the argument. For English the distribution of path patterns is rather simple: the majority of paths consists of a number of ascending arcs followed by zero or one descending arc. Thus a common strategy in SRL systems, formulated by Xue and Palmer (2004), is to look for arguments in the ancestors of the predicate and their direct descendants. However, in Czech and Japanese data we observe a large portion of paths with two or more descending arcs, which makes it difficult to characterize the syntactic scope in which arguments are found. Also, in the datasets for German, Czech and Chinese the three most frequent patterns cover over the 90% of all arguments. In contrast, Japanese exhibits much more variability and a long tail of infrequent types

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 430–435,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

English			German			Czech			Chinese			Japanese		
$\sum$%	%	path	$\sum$%	%	path	$\sum$%	%	path	$\sum$%	%	path	$\sum$%	%	path
63.63	63.6298	↓	77.22	77.2202	↓	63.90	63.8956	↓	78.09	78.0949	↓	37.20	37.1977	↓↓
73.97	10.3429	↑↓	93.51	16.2854	↑↓	86.26	22.3613	↓↓	85.36	7.26962	↑↓	51.52	14.3230	↓
80.63	6.65915	∘	97.43	3.92111	↑↑↓	90.24	3.98078	↑↓	91.27	5.90333	↑↑↓	60.79	9.27270	↓↓↓
85.97	5.33352	↑	98.19	0.76147	↓↓	93.95	3.71713	↓↓↓	95.93	4.66039	↑↑	70.03	9.23857	↑
90.78	4.81104	↑↑↓	98.70	0.51640	↑↑↑↓	95.48	1.52168	↑↓↓	97.53	1.60392	↑	74.17	4.13359	↓↓↓↓
93.10	2.31928	↑↑↑↓	99.17	0.46096	↑	96.92	1.44091	↑	98.28	0.75086	↑↑↑↓	76.76	2.59117	↑↑
95.19	2.09043	↑↑	99.43	0.26841	↑↓↓	97.68	0.76714	↑↑↓	98.77	0.48734	↓↓	78.82	2.06111	↑↑↓↓
96.26	1.07468	↑↑↑↑↓	99.56	0.12837	↑↑↓↓	98.28	0.59684	↓↓↓↓	99.13	0.36270	↑↑↑	80.85	2.03381	↓↓↓↓↓
97.19	0.92482	↓↓	99.67	0.10503	↑↑↑↑↓	98.60	0.31759	↑↓↓↓	99.45	0.31699	↑↑↑↑↓	82.66	1.80631	↑↓↓
97.93	0.74041	↑↑↑	99.77	0.10503	↑↑	98.88	0.28227	↑↑↓↓	99.72	0.27041	↑↑↑↑	83.71	1.05558	↑↑↑
98.41	0.48565	↑↑↑↑↑↓	99.82	0.04960	↓↓↓	99.15	0.26721	↑↑↑↓	99.82	0.10049	↓↓↓	84.74	1.02828	↑↑↑↓↓
98.71	0.29769	↑↑↑↑	99.87	0.04960	↑↑↑	99.27	0.12430	↓↓↓↓↓	99.86	0.03623	↑↓↓	85.68	0.93500	↑↑↓↓↓
98.94	0.22733	↑↑↑↑↑↓	99.90	0.02626	∘	99.37	0.10103	↑↑↑↑↓	99.89	0.02890	↑↑↓↓	86.61	0.93273	↓↓↓↓↓
99.11	0.17805	↑↓↓	99.92	0.02042	↑↑↑↓↓	99.47	0.09747	↑↑	99.92	0.02890	↑↑↑↑↓	87.29	0.68249	↑↑↑↑↓
99.27	0.15316	↓↓↓	99.94	0.02042	↑↑↑↑↓	99.56	0.08515	↑↑↓↓↓	99.94	0.02846	∘	87.90	0.60969	↑↓↓↓
99.39	0.12065	↑↑↑↑↑	99.95	0.01459	↑↑↓↓↓	99.63	0.07419	↑↑↑↓↓	99.96	0.02070	↑↑↑↑↑	88.47	0.56646	↑↑↓↓↓
99.50	0.11024	↑↑↓↓	99.96	0.01167	↓↓↓↓	99.69	0.05667	↑↓↓↓↓	99.97	0.00992	↑↑↓↓↓	89.01	0.53689	↓↓↓↓↓↓
99.60	0.09931	↑↑↑↑↑↑↑↓	99.97	0.00875	↑↓↓↓	99.73	0.04216	↑↑↑↑↑↓	99.98	0.00733	↑↑↑↑↑↑↓	89.49	0.48684	↑↑↑↓↓↓
99.65	0.05283	↑↓↓↓	99.98	0.00875	↑↑↑↑↑↑↓	99.76	0.02875	↑↑↑↓↓↓	99.99	0.00431	↑↑↑↑↓↓	89.94	0.45044	↑↑↑↑

Table 1: Summary of the most frequent paths on the CoNLL-2009 Shared Task datasets. ↑ indicates that we traverse a syntactic dependency upwards from a modifier to a head. ↓ is for dependencies following a descending head to modifier edge. The symbol ∘ represents that the argument is the predicate itself. We exclude from this table Catalan and Spanish as predicates and arguments are always trivially related by a single syntactic dependency that descends.

of patterns. In general it is not feasible to capture path patterns manually, and it is not desirable that a statistical system depends on rather sparse non-factored path features. For this reason in this paper we explore arc-factored models for SRL.

Our method might be specially useful in applications were we are interested in some target semantic role, i.e. retrieving agent relations for some verb, since it processes semantic roles independently of each other. Our method might also be generalizable to other kinds of semantic relations which strongly depend on syntactic patterns such as relation extraction in information extraction or discourse parsing.

2 Arc-factored SRL

We define an SRL parsing model that retrieves predicate-argument relations based on arc-factored syntactic representations of paths connecting predicates with their arguments. Throughout the paper we assume a fixed sentence $\mathbf{x} = x_1, \ldots, x_n$ and a fixed predicate index p. The SRL output is an indicator vector $\mathbf{z}$, where $\mathbf{z}_{r,a} = 1$ indicates that token a is filling role r for predicate p. Our SRL parser performs $\operatorname{argmax}_{\mathbf{z} \in \mathcal{Z}(\mathbf{x},p)} s(\mathbf{x}, p, \mathbf{z})$, where $\mathcal{Z}(\mathbf{x}, p)$ defines the set of valid argument structures for p, and $s(\mathbf{x}, p, \mathbf{z})$ computes a plausibility score for $\mathbf{z}$ given $\mathbf{x}$ and p. Our first assumption is that the score function factors over role-argument pairs:

$$s(\mathbf{x}, p, \mathbf{z}) = \sum_{z_{r,a}=1} s(\mathbf{x}, p, r, a) \quad . \qquad (1)$$

Then we assume two components in the model, one that scores the role-argument pair alone, and another that considers the best ($\max$) syntactic dependency path $\boldsymbol{\pi}$ that connects the predicate p with the argument a:

$$s(\mathbf{x}, p, r, a) = s_0(\mathbf{x}, p, r, a) + \max_{\boldsymbol{\pi}} s_{\text{syn}}(\mathbf{x}, p, r, a, \boldsymbol{\pi}) \quad . \qquad (2)$$

The model does not assume access to the syntactic structure of $\mathbf{x}$, hence in Eq. (2) we locally retrieve the maximum-scoring path for an argument-role pair. A path $\boldsymbol{\pi}$ is a sequence of dependencies $\langle h, m, l \rangle$ where h is the head, m the modifier and l the syntactic label. We further assume that the syntactic component factors over the dependencies in the path:

$$s_{\text{syn}}(\mathbf{x}, p, r, a, \boldsymbol{\pi}) = \sum_{\langle h,m,l \rangle \in \boldsymbol{\pi}} s_{\text{syn}}(\mathbf{x}, p, r, a, \langle h, m, l \rangle) \; . \qquad (3)$$

This will allow to employ efficient shortest-path inference, which is the main contribution of this paper and is described in the next section. Note that since paths are locally retrieved per role-argument pair, there is no guarantee that the set of paths across roles forms a (sub)tree.

As a final note, in this paper we follow Lluís et al. (2013) and consider a constrained space of valid argument structures $\mathcal{Z}(\mathbf{x}, p)$: (a) each role is realized at most once, and (b) each token fills at most one role. As shown by Lluís et al. (2013), this can be efficiently solved as a linear assign-

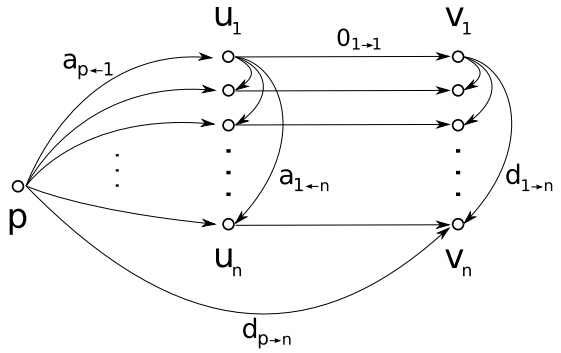


Figure 1: Graph representing all possible syntactic paths from a single predicate to their arguments. We find in this graph the best SRL using a shortest-path algorithm. Note that many edges are omitted for clarity reasons. We labeled the nodes and arcs as follows: p is the predicate and source vertex; $u_1, \ldots, u_n$ are tokens reachable by an ascending path; $v_1, \ldots, v_n$ are tokens reachable by a ascending path (possibly empty) followed by a descending path (possibly empty); $a_{i \leftarrow j}$ is an edge related to an *ascending* dependency from node u_i to node u_j; $d_{i \rightarrow j}$ is a *descending* dependency from node v_i to node v_j; $0_{i \rightarrow i}$ is a 0-weighted arc that connects the ascending portion of the path ending at u_i with the descending portion of the path starting at v_i.

ment problem as long as the SRL model factors over role-argument pairs, as in Eq. (1).

3 SRL as a Shortest-path Problem

We now focus on solving the maximization over syntactic paths in Eq. (2). We will turn it into a minimization problem which can be solved with a polynomial-cost algorithm, in our case a shortest-path method. Assume a fixed argument and role, and define $\theta_{\langle h,m,l \rangle}$ to be a *non-negative penalty* for the syntactic dependency $\langle h, m, l \rangle$ to appear in the predicate-argument path. We describe a shortest-path method that finds the path of arcs with the smaller penalty:

$$\min_{\boldsymbol{\pi}} \sum_{\langle h,m,l \rangle \in \boldsymbol{\pi}} \theta_{\langle h,m,l \rangle} \quad . \tag{4}$$

We find these paths by appropriately constructing a weighted graph $G = (V, E)$ that represents the problem. Later we show how to adapt the arc-factored model scores to be non-negative penalties, such that the solution to Eq. (4) will be the negative of the maximizer of Eq. (2).

It remains only to define the graph construction where paths correspond to arc-factored edges weighted by θ penalties. We start by noting that any path from a predicate p to an argument v_i is formed by a number of *ascending* syntactic arcs followed by a number of *descending* arcs. The ascending segment connects p to some ancestor q (q might be p itself, which implies an empty ascending segment); the descending segment connects q with v_i (which again might be empty). To compactly represent all these possible paths we define the graph as follows (see Figure 1):

1. Add node p as the *source* node of the graph.
2. Add nodes $u_1, \ldots, u_n$ for every token of the sentence except p.
3. Link every pair of these nodes u_i, u_j with a directed edge $a_{i \leftarrow j}$ weighted by the corresponding ascending arc, namely $\min_l \theta_{\langle j,i,l \rangle}$. Also add ascending edges from p to any u_i weighted by $\min_l \theta_{\langle i,p,l \rangle}$. So far we have a connected component representing all ascending path segments.
4. Add nodes $v_1, \ldots, v_n$ for every token of the sentence except p, and add edges $d_{i \rightarrow j}$ between them weighted by descending arcs, namely $\min_l \theta_{\langle i,j,l \rangle}$. This adds a second strongly-connected component representing descending path segments.
5. For each i, add an edge from u_i to v_i with weight 0. This ensures that ascending and descending path segments are connected consistently.
6. Add direct descending edges from p to all the v_i nodes to allow for only-descending paths, weighted by $\min_l \theta_{\langle p,i,l \rangle}$.

Dijkstra's algorithm (Dijkstra, 1959) will find the optimal path from predicate p to all tokens in time $O(V^2)$ (see Cormen et al. (2009) for an in-depth description). Thus, our method runs this algorithm for each possible role of the predicate, obtaining the best paths to all arguments at each run.

4 Adapting and Training Model Scores

The shortest-path problem is undefined if a negative cycle is found in the graph as we may indefinitely decrease the cost of a path by looping over this cycle. Furthermore, Dijkstra's algorithm requires all arc scores to be non-negative penalties. However, the model in Eq. (3) computes plausibility scores for dependencies, not penalties. And, if we set this model to be a standard feature-based linear predictor, it will predict unrestricted real-valued scores.

One approach to map plausibility scores to penalties is to assume a log-linear form for our

model. Let us denote by $\bar{x}$ the tuple $\langle \mathbf{x}, p, r, a \rangle$, which we assume fixed in this section. The log-linear model predicts:

$$\Pr(\langle h, m, l \rangle \mid \bar{x}) = \frac{\exp\{\mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle)\}}{Z(\bar{x})}\,, \tag{5}$$

where $\mathbf{f}(\bar{x}, \langle h, m, l \rangle)$ is a feature vector for an arc in the path, $\mathbf{w}$ are the parameters, and $Z(\bar{x})$ is the normalizer. We can turn predictions into non-negative penalties by setting $\theta_{\langle h,m,l \rangle}$ to be the negative log-probability of $\langle h, m, l \rangle$; namely $\theta_{\langle h,m,l \rangle} = -\mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle) + \log Z(\bar{x})$. Note that $\log Z(\bar{x})$ shifts all values to the non-negative side.

However, log-linear estimation of $\mathbf{w}$ is typically expensive since it requires to repeatedly compute feature expectations. Furthermore, our model as defined in Eq. (2) combines arc-factored path scores with path-independent scores, and it is desirable to train these two components jointly. We opt for a mistake-driven training strategy based on the Structured Averaged Perceptron (Collins, 2002), which directly employs shortest-path inference as part of the training process.

To do so we predict plausibility scores for a dependency directly as $\mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle)$. To map scores to penalties, we define

$$\theta_0 = \max_{\langle h,m,l \rangle} \mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle)$$

and we set

$$\theta_{\langle h,m,l \rangle} = -\mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle) + \theta_0 \quad .$$

Thus, θ_0 has a similar purpose as the log-normalizer $Z(\bar{x})$ in a log-linear model, i.e., it shifts the negated scores to the positive side; but in our version the normalizer is based on the max value, not the sum of exponentiated predictions as in log-linear models. If we set our model function to be

$$s_{\text{syn}}(\bar{x}, \langle h, m, l \rangle) = \mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle) - \theta_0$$

then the shortest-path method is exact.

5 Experiments

We present experiments using the CoNLL-2009 Shared Task datasets (Hajič et al., 2009), for the verbal predicates of English. Evaluation is based on precision, recall and F_1 over correct predicate-argument relations[1]. Our system uses the feature set of the state-of-the-art system by Johansson (2009), but ignoring the features that do not factor over single arcs in the path.

The focus of these experiments is to see the performance of the shortest-path method with respect to the syntactic variability. Rather than running the method with the full set of possible dependency arcs in a sentence, i.e. $O(n^2)$, we only consider a fraction of the most likely dependencies. To do so employ a probabilistic dependency-based model, following Koo et al. (2007), that computes the distribution over head-label pairs for a given modifier, $\Pr(h, l \mid \mathbf{x}, m)$. Specifically, for each modifier token we only consider the dependencies or heads whose probability is above a factor γ of the most likely dependency for the given modifier. Thus, $\gamma = 1$ selects only the most likely dependency (similar to a pipeline system, but without enforcing tree constraints), and as γ decreases more dependencies are considered, to the point where $\gamma = 0$ would select all possible dependencies. Table 2 shows the ratio of dependencies included with respect to a pipeline system for the development set. As an example, if we set $\gamma = 0.5$, for a given modifier we consider the most likely dependency and also the dependencies with probability larger than 1/2 of the probability of the most likely one. In this case the total number of dependencies is 10.3% larger than only considering the most likely one.

Table 3 shows results of the method on development data, when training and testing with different γ values. The general trend is that testing with the most restricted syntactic graph results in the best performance. However, we observe that as we allow for more syntactic variability during training, the results largely improve. Setting $\gamma = 1$ for both training and testing gives a semantic F_1 of 75.9. This configuration is similar to a pipeline approach but considering only factored features. If we allow to train with $\gamma = 0.1$ and we test with $\gamma = 1$ the results improve by 1.96 points to a semantic F_1 of 77.8 points. When syntactic variability is too large, e.g., $\gamma = 0.01$, no improvements are observed.

Finally, table 4 shows results on the verbal English WSJ test set using our best configuration

[1] Unlike in the official CoNLL-2009 evaluation, in this work we exclude the predicate sense from the features and the evaluation.

Threshold γ	1	0.9	0.5	0.1	0.01
Ratio	1	1.014	1.103	1.500	2.843

Table 2: Ratio of additional dependencies in the graphs with respect to a single-tree pipeline model ($\gamma = 1$) on development data.

Threshold	**prec** (%)	**rec** (%)	$\mathbf{F_1}$
training γ = 1			
1	77.91	73.97	75.89
0.9	77.23	74.17	75.67
0.5	73.30	**75.03**	74.16
0.1	58.22	68.75	63.05
0.01	32.83	53.69	40.74
training γ = 0.5			
1	81.17	73.57	77.18
0.9	80.74	73.78	77.10
0.5	78.40	74.79	76.55
0.1	65.76	71.61	68.56
0.01	42.95	57.68	49.24
training γ = 0.1			
1	**84.03**	72.52	**77.85**
0.9	83.76	72.66	77.82
0.5	82.75	73.33	77.75
0.1	77.25	72.20	74.64
0.01	63.90	65.98	64.92
training γ = 0.01			
1	81.62	69.06	74.82
0.9	81.45	69.19	74.82
0.5	80.80	69.80	74.90
0.1	77.92	68.94	73.16
0.01	74.12	65.92	69.78

Table 3: Results of our shortest-path system for different number of allowed dependencies showing precision, recall and F_1 on development set for the verbal predicates of the *English* language.

from the development set. We compare to the state-of-the art system by Zhao et al. (2009) that was the top-performing system for the English language in SRL at the CoNLL-2009 Shared Task. We also show the results for a shortest-path system trained and tested with $\gamma = 1$. In addition we include an equivalent pipeline system using all features, both factored and non-factored, as defined in Johansson (2009). We observe that by not being able to capture non-factored features the final performance drops by 1.6 F_1 points.

6 Conclusions

We have formulated SRL in terms of shortest-path inference. Our model predicts semantic roles together with associated syntactic paths, and assumes an arc-factored representation of the path. This property allows for efficient shortest-path algorithms that, given a predicate and a role, retrieve the most likely argument and its path.

System	**prec(%)**	**rec(%)**	$\mathbf{F_1}$
Zhao et al. 2009	86.91	81.22	83.97
Non-factored	86.96	75.92	81.06
Factored $\gamma = 1$	79.88	76.12	77.96
Factored best	85.26	74.41	79.46

Table 4: Test set results for verbal predicates of the in-domain *English* dataset. The configurations are labeled as follows. *Factored* $\gamma = 1$: our shortest-path system trained and tested with $\gamma = 1$, similar to a pipeline system but without enforcing tree constraints and restricted to arc-factored features. *Factored best*: our shortest-path system with the best results from table 3. *Non-factored*: an equivalent pipeline system that includes both factored and non-factored features.

In the experimental section we prove the feasibility of the approach. We observe that arc-factored models are in fact more restricted, with a drop in accuracy with respect to unrestricted models. However, we also observe that our method largely improves the robustness of the arc-factored method when training with a degree of syntactic variability. Overall, ours is a simple strategy to bring arc-factored models close to the performance of unrestricted models. Future work should explore further approaches to parse partial syntactic structure specific to some target semantic relations.

Acknowledgments

This work was financed by the European Commission for the XLike project (FP7-288342); and by the Spanish Government for projects Tacardi (TIN2012-38523-C02-00) and Skater (TIN2012-38584-C06-01). For a large part of this work Xavier Carreras was at the Universitat Politècnica de Catalunya under a Ramón y Cajal contract (RYC-2008-02223).

References

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in*

Natural Language Processing, pages 1–8. Association for Computational Linguistics, July.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*. The MIT Press.

Edsger W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, September.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 239–246, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), June 4-5*, Boulder, Colorado, USA.

James Henderson, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proceedings of CoNLL-2008 Shared Task*.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic–semantic analysis with propbank and nombank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 183–187, Manchester, England, August. Coling 2008 Organizing Committee.

Richard Johansson. 2009. Statistical bistratal dependency parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 561–569, Singapore, August. Association for Computational Linguistics.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, Prague, Czech Republic, June. Association for Computational Linguistics.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint Arc-factored Parsing of Syntactic and Semantic Dependencies. *Transactions of the Association for Computational Linguistics (TACL)*, 1(1):219–230, May.

Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. 2008. Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics*, 34(2):145–159, June.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow statistic parsing. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 335–342, Barcelona, Spain, July.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 88–94, Barcelona, Spain, July. Association for Computational Linguistics.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 61–66, Boulder, Colorado, June. Association for Computational Linguistics.

Semantic Kernels for Semantic Parsing

Iman Saleh
Faculty of Computers and Information
Cairo University
iman.saleh@fci-cu.edu.eg

Alessandro Moschitti, Preslav Nakov, Lluís Màrquez, Shafiq Joty
ALT Research Group
Qatar Computing Research Institute
{amoschitti,pnakov,lmarquez,sjoty}@qf.org.qa

Abstract

We present an empirical study on the use of semantic information for Concept Segmentation and Labeling (CSL), which is an important step for semantic parsing. We represent the alternative analyses output by a state-of-the-art CSL parser with tree structures, which we rerank with a classifier trained on two types of semantic tree kernels: one processing structures built with words, concepts and Brown clusters, and another one using semantic similarity among the words composing the structure. The results on a corpus from the restaurant domain show that our semantic kernels exploiting similarity measures outperform state-of-the-art rerankers.

1 Introduction

Spoken Language Understanding aims to interpret user utterances and to convert them to logical forms or, equivalently, to database queries, which can then be used to satisfy the user's information needs. This process is known as Concept Segmentation and Labeling (CSL), also called *semantic parsing* in the speech community: it maps utterances into meaning representations based on semantic constituents. The latter are basically word sequences, often referred to as concepts, attributes or semantic tags. CSL makes it easy to convert spoken questions such as "cheap lebanese restaurants in doha with take out" into database queries.

First, a language-specific semantic parser tokenizes, segments and labels the question:

[*Price* cheap] [*Cuisine* lebanese] [*Other* restaurants in] [*City* doha] [*Other* with] [*Amenity* take out]

Then, label-specific normalizers are applied to the segments, with the option to possibly relabel mislabeled segments:

[*Price* low] [*Cuisine* lebanese] [*City* doha] [*Amenity* carry out]

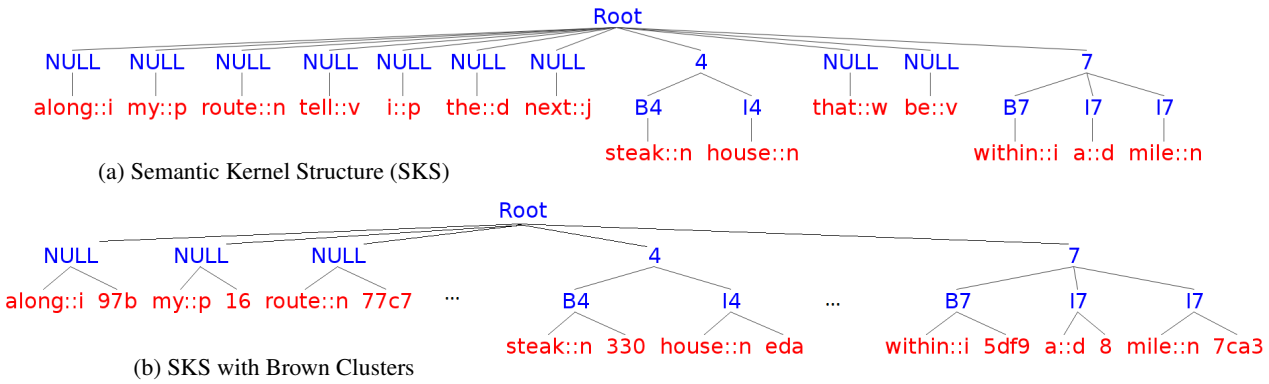
Finally, a database query is formed from the list of labels and values, and is then executed against the database, e.g., MongoDB; a backoff mechanism may be used if the query has not succeeded.

```
{$and [{cuisine:"lebanese"},{city:"doha"},
  {price:"low"},{amenity:"carry out"}]}
```

The state-of-the-art of CSL is represented by conditional models for sequence labeling such as Conditional Random Fields (CRFs) (Lafferty et al., 2001) trained with simple morphological and lexical features. The basic CRF model was improved by means of reranking (Moschitti et al., 2006; Dinarelli et al., 2012) using structural kernels (Moschitti, 2006). Although these methods exploited sentence structure, they did not use syntax at all. More recently, we applied shallow syntactic structures and discourse parsing with slightly better results (Saleh et al., 2014). However, the most obvious models for semantic parsing, i.e., rerankers based on semantic structural kernels (Bloehdorn and Moschitti, 2007b), had not been applied to semantic structures yet.

In this paper, we study the impact of semantic information conveyed by Brown Clusters (BCs) (Brown et al., 1992) and semantic similarity, while also combining them with innovative features. We use reranking, similarly to (Saleh et al., 2014), to select the best hypothesis annotated with concepts predicted by a local model. The competing hypotheses are represented as innovative trees enriched with the semantic concepts and BC labels. The trees can capture dependencies between sentence constituents, concepts and BCs. However, extracting explicit features from them is rather difficult as their number is exponentially large. Thus, we rely on (*i*) Support Vector Machines (Joachims, 1999) to train the reranking functions and on (*ii*) structural kernels (Moschitti, 2010; Moschitti, 2012; Moschitti, 2013) to automatically encode tree fragments that represent syntactic and semantic dependencies from words and concepts.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 436–442,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics



(a) Semantic Kernel Structure (SKS)

(b) SKS with Brown Clusters

Figure 1: CSL structures: standard and with Brown Clusters.

We further apply a semantic kernel (SK), namely the Smoothed Partial Tree Kernel (Croce et al., 2011), which uses the lexical similarity between the tree nodes, while computing the substructure space. This is the first time that SKs are applied to reranking hypotheses. This (*i*) makes the global sentence structure along with concepts available to the learning algorithm, and (*ii*) enables computing the similarity between lexicals in syntactic patterns that are enriched by concepts.

We tested our models on the *Restaurant* domain. Our results show that: *(i)* The basic CRF parser, which uses semi-Markov CRF, or semi-CRF (Sarawagi and Cohen, 2004), is already very accurate; it achieves F_1 scores over 83%, making any further improvement very hard. (*ii*) The upper-bound performance of the reranker is very high as well, i.e., the correct annotation is generated in the list of the first 100 hypotheses in 98.72% of the cases. (*iii*) SKs significantly improve over the semi-CRF baseline and our previous state-of-the-art reranker exploiting shallow syntactic patterns (Saleh et al., 2014), as shown by extensive comparisons using several systems. (*iv*) Making BCs effective requires a deeper study.

2 Related Work

One of the early approaches to CSL was that of Pieraccini et al. (1991), where the word sequences and *concepts* were modeled using Hidden Markov Models (HMMs) as observations and hidden states, respectively. Generative models were exploited by Seneff (1989) and Miller et al. (1994), who used stochastic grammars for CSL. Other discriminative models followed such preliminary work, e.g., (Rubinstein and Hastie, 1997; Santafé et al., 2007; Raymond and Riccardi, 2007). CRF-based models are considered to be the state of the art in CSL (De Mori et al., 2008).

Another relevant line of research are the semantic kernels, i.e., kernels that use lexical similarity between features. One of the first that applyed LSA was (Cristianini et al., 2002), whereas (Bloehdorn et al., 2006; Basili et al., 2006) used WordNet. Semantic structural kernels of the type we use in this paper were first introduced in (Bloehdorn and Moschitti, 2007a; Bloehdorn and Moschitti, 2007b). The most advanced model based on tree kernels, which we also use in this paper, is the Smoothed PTK (Croce et al., 2011).

3 Reranking for CSL

Reranking is applied to a list of N annotation hypotheses, which are generated and sorted by the probability to be globally correct as estimated using local classifiers or global classifiers that only use local features. Then, a reranker, typically a meta-classifier, tries to select the best hypothesis from the list. The reranker can exploit global information, and specifically, the dependencies between the different concepts, which are made available by the local model. We use semi-CRFs for the local model as they yield the highest accuracy in CSL (when using a single model) and preference reranking for the global reranker.

3.1 Preference Reranking (PR)

PR uses a classifier $\mathcal{C}$, which takes a pair of hypotheses $\langle H_i, H_j \rangle$ and decides whether H_i is better than H_j. Given a training question Q, positive and negative examples are built for training the classifier. Let H_1 be the hypothesis with the lowest error rate with respect to the gold standard among all hypotheses generated for question Q. We adopt the following approach for example generation: the pairs $\langle H_1, H_i \rangle$ $(i = 2, 3, \ldots, N)$ are positive examples, while $\langle H_i, H_1 \rangle$ are considered negative.

At testing time, given a new question Q′, $\mathcal{C}$ classifies all pairs $\langle H_i, H_j \rangle$ generated from the annotation hypotheses of Q′: a positive classification is a vote for H_i, otherwise the vote is for H_j, where the classifier score can be used as a weighted vote. H_k are then ranked according to the number (sum) of the votes (weighted by score) they receive.

We build our reranker with SVMs using the following kernel: $K(\langle H_1, H_2 \rangle, \langle H'_1, H'_2 \rangle) = \phi(\langle H_1, H_2 \rangle) \cdot \phi(\langle H'_1, H'_2 \rangle) \triangleq \big(\phi(H_1) - \phi(H_2)\big) \cdot \big(\phi(H'_1) - \phi(H'_2)\big) = \phi(H_1)\phi(H'_1) + \phi(H_2)\phi(H'_2) - \phi(H_1)\phi(H'_2) - \phi(H_2)\phi(H'_1) = S(H_1, H'_1) + S(H_2, H'_2) - S(H_1, H'_2) - S(H_2, H'_1)$. We consider H as a tuple $\langle T, \vec{v} \rangle$ composed of a tree T and a feature vector $\vec{v}$. Then, we define $S(H, H') = S_{\text{TK}}(T, T') + S_{\text{v}}(\vec{v}, \vec{v}')$, where S_{TK} computes one of the tree kernel functions defined in 3.2 and 3.3; and S_{v} is a kernel (see 3.4), e.g., linear, polynomial, Gaussian, etc.

3.2 Tree kernels (TKs)

TKs measure the similarity between two structures in terms of the number of substructures they share. We use two types of tree kernels: (*i*) Partial Tree Kernel (PTK), which can be effectively applied to both constituency and dependency parse trees (Moschitti, 2006). It generates all possible connected tree fragments, e.g., sibling nodes can be also separated and can be part of different tree fragments: a fragment is any possible tree path, and other tree paths are allowed to depart from its nodes. Thus, it can generate a very rich feature space. (*ii*) The smoothed PTK or semantic kernel (SK) (Croce et al., 2011), which extends PTK by allowing soft matching (i.e., via similarity computation) between nodes associated with different but related lexical items. The node similarity can be derived from manually annotated resources, e.g., WordNet or Wikipedia, as well as using corpus-based clustering approaches, e.g., latent semantic analysis (LSA), as we do in this paper.

3.3 Semantic structures

Tree kernels allow us to compute structural similarities between two trees; thus, we engineered a special structure for the CSL task. In order to capture the structural dependencies between the semantic tags,[1] we use a basic tree (see for example Figure 1a), where the words of a sentence are tagged with their semantic tags.

More specifically, the words in the sentence constitute the leaves of the tree, which are in turn connected to the pre-terminals containing the semantic tags in BIO notation ('B'=*begin*, 'I'=*inside*, 'O'=*outside*). The BIO tags are then generalized in the upper level, and joined to the Root node. Additionally, part-of-speech (POS) tags[2] are added to each word by concatenating it with the string "::L", where L is the first letter of the POS-tags of the words, e.g., *along*, *my* and *route*, receive i, p and n, which are the first letters of the POS-tags IN, PRN and NN, respectively. SK applied to the above structure can generate powerful semantic patterns such as [Root [4-Cuisine [similar_to(*stake house*)]][7-Loc [similar_to(*within a mile*)]]], e.g., for correctly labeling new clauses like *Pizza Parlor in three kilometers*. The BC labels, represented as cluster IDs, are simply added as siblings of words as shown in Fig. 1b.

3.4 Feature Vectors

For the sake of comparison, we also devoted some effort towards engineering a set of features to be used in a flat feature-vector representation. These features can be used in isolation to learn the reranking function, or in combination with the kernel-based approach (as a composite kernel using a linear combination). They belong to the following four categories: (*i*) CRF-based: these include the basic features used to train the initial semi-CRF model; (*ii*) n-gram based: we collected 3- and 4-grams of the output label sequence at the level of concepts, with artificial tags inserted to identify the start ('S') and end ('E') of the sequence.[3] (*iii*) Probability-based, computing the probability of the label sequence as an average of the probabilities at the word level in the N-best list; and (*iv*) DB-based: a single feature encoding the number of results returned from the database when constructing a query using the conjunction of all semantic segments in the hypothesis.

4 Experiments

The experiments aim at investigating the role of feature vectors, PTK, SK and BCs in reranking. We first describe the experimental setting and then we move into the analysis of the results.

[1] They are associated with the following IDs: 0-Other, 1-Rating, 2-Restaurant, 3-Amenity, 4-Cuisine, 5-Dish, 6-Hours, 7-Location, and 8-Price.

[2] We use the Stanford tagger (Toutanova et al., 2003).

[3] For instance, if the output sequence is *Other-Rating-Other-Amenity* the 3-gram patterns would be: S-*Other-Rating, Other-Rating-Other, Rating-Other-Amenity*, and *Other-Amenity*-E.

	Train	Devel.	Test	Total
semi-CRF	6,922	739	1,521	9,182
Reranker	7,000	3,695	7,605	39,782

Table 1: Number of instances and pairs used to train the semi-CRF and rerankers, respectively.

4.1 Experimental setup

Dataset. In our experiments, we used questions annotated with semantic tags, which were collected through crowdsourcing on Amazon Mechanical Turk and made available[4] by McGraw et al. (2012). We split the dataset into training, development and test sets. Table 1 shows the number of examples and example pairs we used for the semi-CRF and the reranker, respectively. We subsequently split the training data randomly into 10 folds. We used cross-validation, i.e., iteratively training with 9 folds and annotating the remaining fold, in order to generate the N-best lists of hypotheses for the entire training dataset. We computed the 100-best hypotheses for each example. We then used the development dataset to test and tune the hyper-parameters of our reranking model. The results on the development set, which we will present in Section 4.2 below, were obtained using semi-CRF and reranking models trained on the training set.

Data representation. Each hypothesis is represented by a semantic tree, a feature vector (explained in Section 3), and two extra features: (*i*) the semi-CRF probability of the hypothesis, and (*ii*) its reciprocal rank in the N-best list.

Learning algorithm. We used the SVM-Light-TK[5] to train the reranker with a combination of tree kernels and feature vectors (Moschitti, 2006; Joachims, 1999). We used the default parameters and a linear kernel for the feature vectors. As a baseline, we picked the best-scoring hypothesis in the list, i.e., the output by the regular semi-CRF parser. The setting is exactly the same as that described in (Saleh et al., 2014).

Evaluation measure. In all experiments, we used the harmonic mean of precision and recall (F_1) (van Rijsbergen, 1979), computed at the token level and micro-averaged across the different semantic types.[6]

[4]http://groups.csail.mit.edu/sls/downloads/restaurant/

[5]http://disi.unitn.it/moschitti/Tree-Kernel.htm

[6]We do not consider 'Other' to be a semantic type; thus, we did not include it in the F_1 calculation.

N	1	2	**5**	10	100
F_1	83.03	87.76	**92.63**	95.23	98.72

Table 2: Oracle F_1 score for N-best lists.

Brown Clusters. Clustering groups of similar words together provides a way of generalizing them. In this work, we explore the use of Brown clusters (Brown et al., 1992) in both feature vectors and tree kernels. The Brown clustering algorithm uses an n-gram class model. It first assigns each word to a distinct cluster, and then it merges different clusters in a bottom-up fashion. The merge step is done in a way that minimizes the loss in average mutual information between clusters. The outcome is hierarchical clustering, which we use in our reranking algorithm. To create the Brown clusters, we used the Yelp dataset of reviews.[7] It contains 335,022 reviews about 15,585 businesses; 5,575 of the businesses and 233,839 of the reviews are restaurant-related. This dataset is very similar to the dataset of queries about restaurants we use in our experiments.

Similarity matrix for SK. We compute the lexical similarity for SK by applying LSA (Furnas et al., 1988) to Tripadvisor data. The dataset and the exact procedure for creating the LSA matrix are described in (Castellucci et al., 2013; Croce and Previtali, 2010).

4.2 Results

Oracle accuracy. Table 2 shows the oracle F_1 score for N-best lists of different lengths, i.e., the F_1 that is achieved by picking the best candidate in the N-best list for various values of N. Considering 5-best lists yields an increase in oracle F_1 of almost ten absolute points. Going up to 10-best lists only adds 2.5 extra F_1 points. The complete 100-best lists add 3.5 extra F_1 points, for a total of 98.72. This very high value is explained by the fact that often the total number of different annotations for a given question is smaller than 100. In our experiments, we will focus on 5-best lists.

Baseline accuracy. We computed F_1 for the semi-CRF model on both the development and the test sets, obtaining 83.86 and 83.03, respectively.

Learning Curves. The semantic information in terms of BCs or semantic similarity derived by LSA can have a major impact in case of data scarcity. Therefore, we trained our reranking models with increasing sizes of training data.

[7]http://www.yelp.com/dataset_challenge/

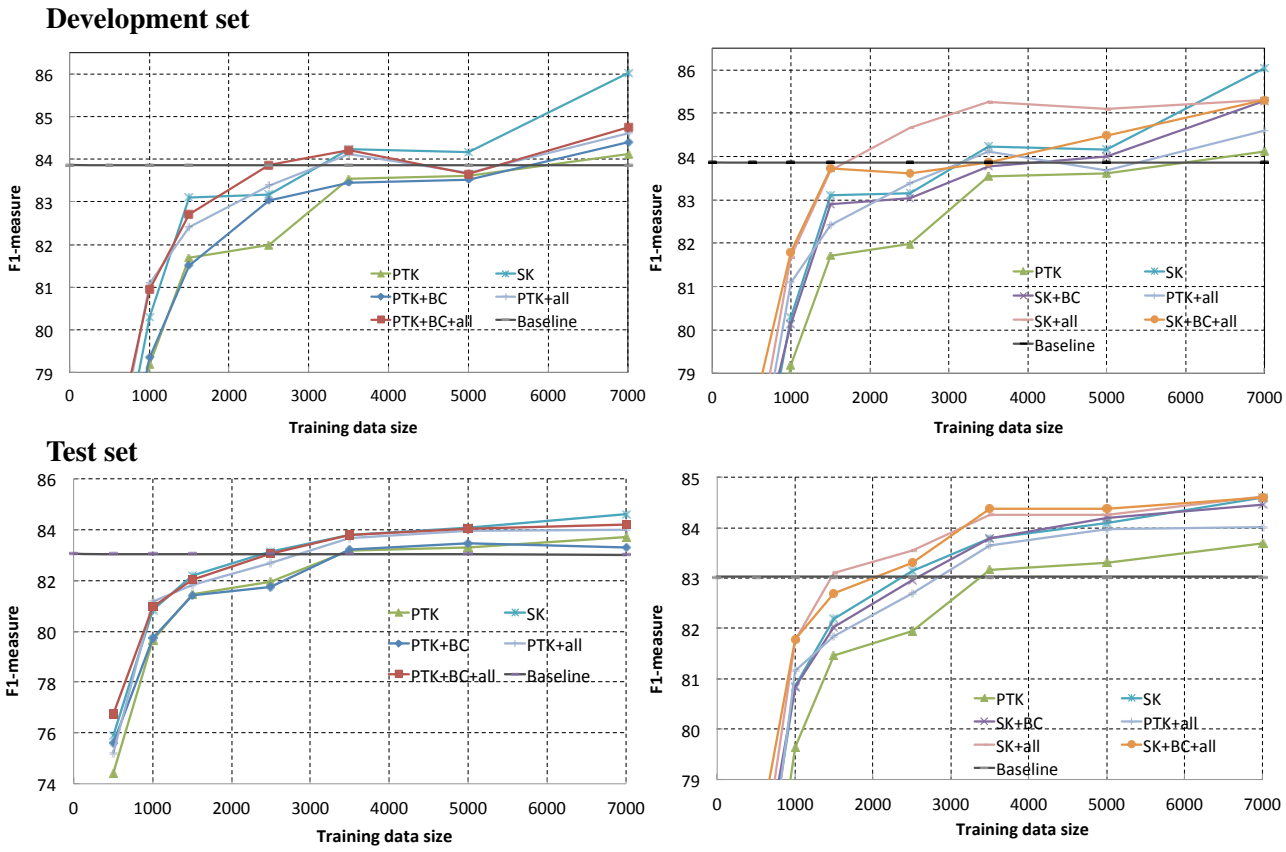


Figure 2: Learning curves for different reranking models on the development and on the testing sets.

The first two graphs in Fig. 2 show the plots on the development set whereas the last two are computed on the test set. The reranking models reported are Baseline, PTK, PTK+BC, PTK+all (features), PTK+BC+all, SK, SK+BC, SK+all and SK+BC+all.[8] We can see that: (*i*) PTK alone, i.e., without semantic information, has the lowest accuracy; (*ii*) BCs do not improve significantly any model; (*iii*) SK almost always achieves the highest accuracy; (*iv*) PTK+all (i.e., the model also using features) improves on PTK, but its accuracy is lower than for any model using SK, i.e., using semantic similarity; and (*v*) all features provide an initial boost to SK, but as soon as the data increases, their impact decreases.

5 Conclusion and Future Work

In summary, the learning curves clearly show the good generalization ability of SK, which improve the CRF baseline using little data (∼3,000). The semantic kernel significantly improves over the semi-CRF baseline and our previous state-of-the-art reranker exploiting shallow syntactic patterns (Saleh et al., 2014), which corresponds to PTK+all in the above comparison.

[8]Models are split between 2 plots in order to ease reading.

The improvement falls between 1-2 absolute percent points. This is remarkable as (*i*) it corresponds to ∼10% relative error reduction, and (*ii*) the state-of-the-art baseline system is very difficult to beat, as confirmed by the low impact of traditional features and BCs. Although the latter can generalize over concepts and words, their use is not straightforward, resulting in no improvement.

In the future, we plan to investigate the use of semantic similarity from distributional and other sources (Mihalcea et al., 2006; Padó and Lapata, 2007), e.g., Wikipedia (Strube and Ponzetto, 2006; Mihalcea and Csomai, 2007), Wiktionary (Zesch et al., 2008), WordNet (Pedersen et al., 2004; Agirre et al., 2009), FrameNet, VerbNet (Shi and Mihalcea, 2005), BabelNet (Navigli and Ponzetto, 2010), and LSA, and for different domains.

Acknowledgments

This research is part of the Interactive sYstems for Answer Search (Iyas) project, conducted by the Arabic Language Technologies (ALT) group at Qatar Computing Research Institute (QCRI) within the Qatar Foundation. We would like to thank Danilo Croce, Roberto Basili and Giuseppe Castellucci for helping and providing us with the similarity matrix for the semantic kernels.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27, Boulder, Colorado, June.

Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2006. A semantic kernel to classify texts with very few training examples. *Informatica (Slovenia)*, 30(2):163–172.

Stephan Bloehdorn and Alessandro Moschitti. 2007a. Combined syntactic and semantic kernels for text classification. In *Advances in Information Retrieval - Proceedings of the 29th European Conference on Information Retrieval (ECIR 2007)*, pages 307–318, Rome, Italy.

Stephan Bloehdorn and Alessandro Moschitti. 2007b. Structure and semantics for expressive text kernels. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM 2007)*, pages 861–864, Lisbon, Portugal.

Stephan Bloehdorn, Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2006. Semantic kernels for text classification based on topological measures of feature similarity. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 06)*, pages 808–812, Hong Kong.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Giuseppe Castellucci, Simone Filice, Danilo Croce, and Roberto Basili. 2013. UNITOR: Combining Syntactic and Semantic Kernels for Twitter Sentiment Analysis. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 369–374, Atlanta, Georgia, USA.

Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. 2002. Latent Semantic Kernels. *Journal of Intelligent Information Systems*, 18(2):127–152.

Danilo Croce and Daniele Previtali. 2010. Manifold learning for the semi-supervised induction of framenet predicates: An empirical investigation. In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, pages 7–16, Uppsala, Sweden.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1034–1046, Edinburgh, Scotland, UK.

Renato De Mori, Frederic Béchet, Dilek Hakkani-Tür, Michael McTear, Giuseppe Riccardi, and Gokhan Tur. 2008. Spoken Language Understanding. *IEEE Signal Processing Magazine*, 25:50–58.

Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. 2012. Discriminative reranking for spoken language understanding. *IEEE Transactions on Audio, Speech and Language Processing*, 20(2):526–539.

G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. 1988. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '88)*, pages 465–480, New York, USA.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, USA.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 282–289, Williamstown, MA, USA.

Ian McGraw, Scott Cyphers, Panupong Pasupat, Jingjing Liu, and Jim Glass. 2012. Automating crowd-supervised learning for spoken language systems. In *Proceedings of the 13th Annual Conference of the International Speech Communication Association (INTERSPEECH 2012)*, pages 2473–2476, Portland, OR, USA.

Rada Mihalcea and Andras Csomai. 2007. Wikify! linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management (CIKM 2007)*, pages 233–242, Lisbon, Portugal.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1 (AAAI 2006)*, pages 775–780, Boston, MA, USA.

Scott Miller, Richard Schwartz, Robert Bobrow, and Robert Ingria. 1994. Statistical Language Processing using Hidden Understanding Models. In *Proceedings of the workshop on Human Language Technology (HLT 1994)*, pages 278–282, Plainsboro, NJ, USA.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2006. Semantic role labeling via tree kernel

joint inference. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 61–68, New York City, USA.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of the 17th European Conference on Machine Learning (ECML 2006)*, pages 318–329, Berlin, Germany.

Alessandro Moschitti. 2010. Kernel engineering for fast and easy design of natural language applications. In *Coling 2010: Kernel Engineering for Fast and Easy Design of Natural Language Applications–Tutorial notes*, pages 1–91, Beijing, China.

Alessandro Moschitti. 2012. State-of-the-art kernels for natural language processing. In *Tutorial Abstracts of ACL 2012*, page 2, Jeju Island, Korea.

Alessandro Moschitti. 2013. Kernel-based learning to rank with syntactic and semantic structures. In *Tutorial abstracts of the 36th Annual ACM SIGIR Conference*, page 1128, Dublin, Ireland.

Roberto Navigli and Simone Paolo Ponzetto. 2010. Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the association for computational linguistics (ACL 2010)*, pages 216–225, Uppsala, Sweden.

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

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity - measuring the relatedness of concepts. In *HLT-NAACL 2004: Demonstration Papers*, pages 38–41, Boston, Massachusetts, USA.

Roberto Pieraccini, Esther Levin, and Chin-Hui Lee. 1991. Stochastic Representation of Conceptual Structure in the ATIS Task. In *Proceedings of the Fourth Joint DARPA Speech and Natural Language Workshop*, pages 121–124, Los Altos, CA, USA.

Christian Raymond and Giuseppe Riccardi. 2007. Generative and Discriminative Algorithms for Spoken Language Understanding. In *Proceedings of the 8th Annual Conference of the International Speech Communication Association (INTERSPEECH 2007)*, pages 1605–1608, Antwerp, Belgium, August.

Y. Dan Rubinstein and Trevor Hastie. 1997. Discriminative vs Informative Learning. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-1997)*, pages 49–53, Newport Beach, CA, USA.

Iman Saleh, Scott Cyphers, Jim Glass, Shafiq Joty, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2014. A study of using syntactic and semantic structures for concept segmentation and labeling. In *Proceedings of the 25th International Conference on Computational Linguistics*, COLING '14, pages 193–202, Dublin, Ireland.

G. Santafé, J.A. Lozano, and P. Larrañaga. 2007. Discriminative vs. Generative Learning of Bayesian Network Classifiers. In *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2007)*, pages 453–546, Hammamet, Tunisia.

Sunita Sarawagi and William W. Cohen. 2004. Semimarkov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems 17 (NIPS 2004)*, Vancouver, British Columbia, Canada.

Stephanie Seneff. 1989. TINA: A Probabilistic Syntactic Parser for Speech Understanding Systems. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP-89)*, pages 711–714, Glasgow, UK.

Lei Shi and Rada Mihalcea. 2005. Putting pieces together: Combining framenet, verbnet and wordnet for robust semantic parsing. In *Computational Linguistics and Intelligent Text Processing*, pages 100–111. Springer Berlin Heidelberg.

Michael Strube and Simone Paolo Ponzetto. 2006. Wikirelate! computing semantic relatedness using wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06)*, pages 1419–1424, Boston, Massachusetts, USA.

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

Cornelis J. van Rijsbergen. 1979. *Information Retrieval*. Butterworth-Heinemann Newton, MA, USA.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Using wiktionary for computing semantic relatedness. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI'08)*, pages 861–866, Chicago, Illinois,USA.

An I-vector Based Approach to Compact Multi-Granularity Topic Spaces Representation of Textual Documents

Mohamed Morchid[†], Mohamed Bouallegue[†], Richard Dufour[†], Georges Linarès[†], Driss Matrouf[†] and Renato de Mori[†‡]
[†]LIA, University of Avignon, France
[‡]McGill University, School of Computer Science, Montreal, Quebec, Canada
{firstname.lastname}@univ-avignon.fr
rdemori@cs.mcgill.ca

Abstract

Various studies highlighted that topic-based approaches give a powerful spoken content representation of documents. Nonetheless, these documents may contain more than one main theme, and their automatic transcription inevitably contains errors. In this study, we propose an original and promising framework based on a compact representation of a textual document, to solve issues related to topic space granularity. Firstly, various topic spaces are estimated with different numbers of classes from a Latent Dirichlet Allocation. Then, this multiple topic space representation is compacted into an elementary segment, called c-vector, originally developed in the context of speaker recognition. Experiments are conducted on the DECODA corpus of conversations. Results show the effectiveness of the proposed multi-view compact representation paradigm. Our identification system reaches an accuracy of 85%, with a significant gain of 9 points compared to the baseline (best single topic space configuration).

1 Introduction

Automatic Speech Recognition (ASR) systems frequently fail on noisy conditions and high Word Error Rates (WER) make the analysis of the automatic transcriptions difficult. Speech analytics suffer from these transcription issues that may be overcome by improving the ASR robustness and/or the tolerance of speech analytic systems to ASR errors. This paper proposes a new method to improve the robustness of speech analytics by combining a semantic multi-model approach and a noise reduction technique based on the i-vector paradigm.

This method is evaluated in the application framework of the RATP call centre (Paris Public Transportation Authority), focusing on the theme identification task (Bechet et al., 2012).

Telephone conversations are a particular case of human-human interaction whose automatic processing raises problems, especially due to the speech recognition step required to obtain the transcription of the speech contents. First, the speaker's behavior may be unexpected and the training/test mismatch may be very large. Second, the speech signal may be strongly impacted by various sources of variability: environment and channel noises, acquisition devices, etc.

Telephone conversation issues

Topics are related to the reason why the customer called. Various classes corresponding to the main customer's requests are considered (*lost and founds, traffic state, timelines*, etc). In addition to classical issues in such adverse conditions, the topic-identification system should deal with problems related to class proximity. For example, a *lost & found* request is related to itinerary (*where was the object lost?*) or timeline (*when?*), that could appear in most of the classes. In fact, these conversations involve a relatively small set of basic concepts related to transportation issues. Figure 1 shows an example of a dialogue which is manually labeled by the agent as an issue related to an *infraction*. However, words in bold suggest that this conversation could be related to a *transportation card*. Thus, we assume that a dialogue representation should be seen as a multi-view problem to substantiate the claims regarding the multi-theme representation of a given dialogue.

On the other hand, multi-view approaches introduce additional variability due to the diversity of the views. This variability is also due to the vocabulary used by both agent and customer

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 443–454,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

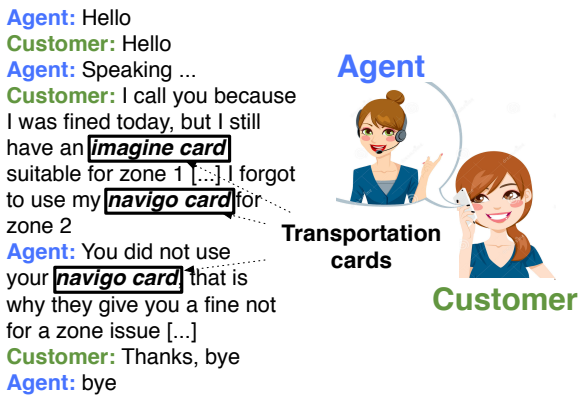


Figure 1: Example of a dialogue from the DECODA corpus labeled by the agent as an *infraction* issue which contains more than one theme (*infraction* + *transportation cards*).

during a telephone conversation. Indeed, an agent have to follow an predefined scenario of conversation. Thus, the agent can find the main reason for the call which corresponds to the theme.

Proposed solutions

An efficient way to tackle both ASR robustness and class ambiguity could be to map dialogues into a topic space abstracting the ASR outputs. Then, dialogue categorization is achieved in this topic space. Numerous unsupervised methods for topic-space estimation were proposed in the past. Latent Dirichlet Allocation (LDA) (Blei et al., 2003) has been largely used for speech analytics; one of its main drawbacks is the tuning of the model, that involves various meta-parameters such as the number of classes (that determines the model granularity), word distribution methods, temporal spans... If the decision process is highly dependent on these features, the system's performance could be quite unstable.

Classically, this abstract representation involves selecting the right number of classes composing the topic space. This decision is crucial since topic model perplexity, which expresses its quality, is highly dependent on this feature. Furthermore, the multi-theme context of the study (see Figure 1) involves a more complex dialogue representation. In this paper, we propose to deal with these two drawbacks by using a compact representation from multiple topic spaces. This model is based on a robust multi-view representation of the textual documents.

A multi-view representation of a dialogue introduces both a *relevant* variability needed to represent different contexts of the dialogue, and a *noisy* variability related to topic space processing. Thus, a topic-based representation of a dialogue is built from the dialogue content itself. For this reason, the mapping process of a dialogue into several topic spaces generates a noisy variability related to the difference between the dialogue and the content of each class. In the same way, the relevant variability comes from the common content between the dialogue and the classes composing the topic space.

We propose to reduce the noisy variability by using a factor analysis technique, which was initially developed in the domain of speaker identification. In this field, the factor analysis paradigm is used as a decomposition model that enables to separate the representation space into two subspaces containing respectively useful and useless information. The general Joint Factor Analysis (JFA) paradigm (Kenny et al., 2008) considers multiple variabilities that may be cross-dependent. Therefore, JFA representation allows us to compensate the variability within sessions of a same speaker. This representation is an extension of the GMM-UBM (Gaussian Mixture Model-Universal Background Model) models (Reynolds and Rose, 1995). (Dehak et al., 2011) extract a compact super-vector (called an i-vector) from the GMM super-vector. The aim of the compression process (i-vector extraction) is to represent the super-vector variability in a low dimensional space. Although this compact representation is widely used in speaker recognition systems, this method has not been used yet in the field of text classification.

In this paper, we propose to apply factor analysis to compensate noisy variabilities due to the multiplication of LDA models. Furthermore, a normalization approach to condition dialogue representations (multi-model and i-vector) is presented. The two methods showed improvements for speaker verification: within Class Covariance Normalization (WCCN) (Dehak et al., 2011) and Eigen Factor Radial (EFR) (Bousquet et al., 2011). The latter includes length normalization (Garcia-Romero and Espy-Wilson, 2011). Both methods dilate the total variability space as a means of reducing the within-class variability. In our multi-model representation, the within class variability is redefined according to both dialogue content

(vocabulary) and topic space characteristics (word distributions among the topics). Thus, the speaker is represented by a theme, and the speaker session is a set of topic-based representations (frames) of a dialogue (session).

The paper is organized as follows. Section 2 presents previous related works. The dialogue representation is described in Section 3. Section 4 introduces the i-vector compact representation and presents its application to text documents. Sections 5 and 6 report experiments and results. The last section concludes and proposes some perspectives.

2 Related work

In the past, several approaches considered a text document as a mixture of latent topics. These methods, such as Latent Semantic Analysis (LSA) (Deerwester et al., 1990; Bellegarda, 1997), Probabilistic LSA (PLSA) (Hofmann, 1999) or Latent Dirichlet Allocation (LDA) (Blei et al., 2003), build a higher-level representation of the document in a topic space. ¿ Document is then considered as a bag-of-words (Salton, 1989) where the word order is not taken into account. These methods have demonstrated their performance on various tasks, such as sentence (Bellegarda, 2000) or keyword (Suzuki et al., 1998) extraction.

In opposition to a multinomial mixture model, LDA considers that a theme is associated to each occurrence of a word composing the document, rather than associate a topic to the complete document. Therefore, a document can change topics from a word to another one. However, word occurrences are connected by a latent variable which controls the global match of the distribution of the topics in the document. These latent topics are characterized by a distribution of associated word probabilities. PLSA and LDA models have been shown to generally outperform LSA on IR tasks (Hofmann, 2001). Moreover, LDA provides a direct estimate of the relevance of a topic given a word set. In this paper, probabilities of hidden topic features, estimated with LDA, are considered for possibly capturing word dependencies expressing the semantic contents of a given conversation.

Topic-based approaches involve defining a number of topics composing the topic space. The choice of the "right" number of topics is a crucial step, especially when the documents may contain multiple themes. Many studies have tried to find a relevant method to deal with this issue. (Arun et al., 2010) proposed to use a Singular Value Decomposition (SVD) to represent the separability between the words contained in the vocabulary. Then, if the singular values of the topic-word matrix $\mathbf{M}$ equal the norm of the rows of $\mathbf{M}$, this means that the vocabulary is well separated among the topics. This method has to be evaluated with the Kullback-Liebler divergence metric for each topic space. However, this process would be time consuming for thousands of representations of a dialogue.

(Teh et al., 2004) proposed the Hierarchical Dirichlet Process (HDP) method to find the "right" number of topics by assuming that the data has a hierarchical structure. The HDP models were then compared to the LDA ones on the same dataset. (Zavitsanos et al., 2008) presented a method to *learn* the right depth of an ontology depending of the number of topics of LDA models. The study presented by (Cao et al., 2009) is quite similar to (Teh et al., 2004). The authors consider the average correlation between pairs of topics at each stage as the right number of topics.

All these methods assume that a document can have only one representation since they consider that finding the optimal topic model is the best solution. Another solution would be to consider a set of topic models to represent a document. Nonetheless, a multi-topic-based representation of a dialogue can involve a noisy variability due to the mapping of a dialogue in each topic space. Indeed, a dialogue does not share its content (*i.e.* words) with each class composing the topic space. Thus, a variability is added during the mapping process. Another weakness of the multi-view representation is the relation between classes in a topic space. (Blei and Lafferty, 2006) show that classes into a LDA topic space are correlated. Moreover, (Li and McCallum, 2006) consider a class as a node of an acyclic graph and as a distribution over other classes contained in the same topic space.

3 Multi-view representation of automatic dialogue transcriptions in a homogeneous space

The purpose of the considered application is the identification of the major theme of a human-human telephone conversation in the customer

care service (CCS) of the RATP Paris transportation system. The approach considered in this paper focuses on modeling the variability between different dialogues expressing the same theme t. For this purpose, it is important to select relevant features that represent semantic contents for the theme of a dialogue. An attractive set of features for capturing possible semantically relevant word dependencies is obtained with Latent Dirichlet Allocation (LDA) (Blei et al., 2003), as described in section 2.

Given a training set of conversations D, a hidden topic space is derived and a conversation d is represented by its probability in each topic of the hidden space. Estimation of these probabilities is affected by a variability inherent to the estimation of the model parameters. If many hidden spaces are considered and features are computed for each hidden space, it is possible to model the estimation variability together with the variability of the linguistic expression of a theme by different speakers in different real-life situations. Even if the purpose of the application is theme identification and a training corpus annotated with themes is available, supervised LDA (Griffiths and Steyvers, 2004) is not suitable for the proposed approach. LDA is used only for producing different feature sets used involved in statistical variability models.

In order to estimate the parameters of different hidden spaces, a set of discriminative words V is constructed as described in (Morchid et al., 2014a). Each theme t contains a set of specific words. Note that the same word may appear in several discriminative word sets. All the selected words are then merged without repetition to form V.

Several techniques, such as Variational Methods (Blei et al., 2003), Expectation-propagation (Minka and Lafferty, 2002) or Gibbs Sampling (Griffiths and Steyvers, 2004), have been proposed for estimating the parameters describing a LDA hidden space. Gibbs Sampling is a special case of Markov-chain Monte Carlo (MCMC) (Geman and Geman, 1984) and gives a simple algorithm for approximate inference in high-dimensional models such as LDA (Heinrich, 2005). This overcomes the difficulty to directly and exactly estimate parameters that maximize the likelihood of the whole data collection defined as: $p(W|\overrightarrow{\alpha}, \overrightarrow{\beta}) = \prod_{w \in W} p(\overrightarrow{w}|\overrightarrow{\alpha}, \overrightarrow{\beta})$ for the whole data collection W knowing the Dirichlet parameters $\overrightarrow{\alpha}$ and $\overrightarrow{\beta}$.

Gibbs Sampling allows us both to estimate the LDA parameters in order to represent a new dialogue d with the r^{th} topic space of size n, and to obtain a feature vector $V_d^{z^r}$ of the topic representation of d. The j^{th} feature $V_d^{z_j^r} = P(z_j^r|d)$ (where $1 \leq j \leq n$) is the probability of topic z_j^r to be generated by the unseen dialogue d in the r^{th} topic space of size n (see Figure 2) and $V_{z_j^r}^w = P(w|z_j^r)$ is the vector representation of a word into r.

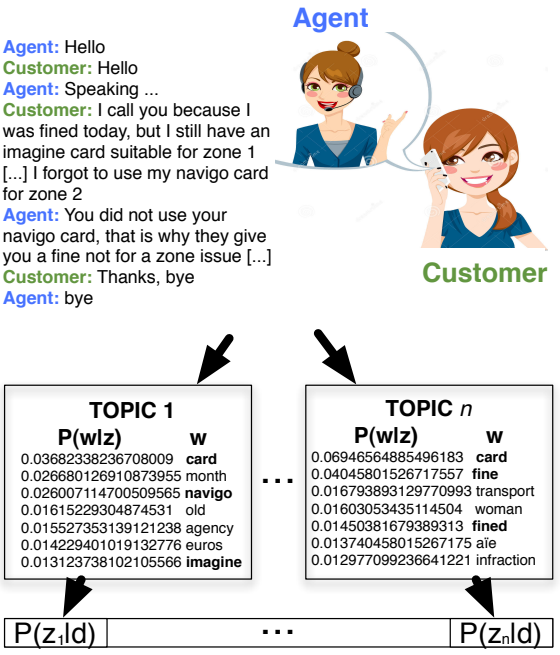


Figure 2: Example of a dialogue d mapped into a topic space of size n.

In the LDA technique, topic z_j, j is drawn from a multinomial over θ which is drawn from a Dirichlet distribution over $\overrightarrow{\alpha}$. Thus, a set of p topic spaces are learned using LDA by varying the number of topics n to obtain p topic spaces of size n. The number of topics n varies from 10 to $3,010$. Thus, a set of $3,000$ topic spaces is estimated. This is high enough to generate, for each dialogue, many feature sets for estimating the parameters of a variability model.

The next process allows us to obtain a homogeneous representation of transcription d for the r^{th} topic space r. The feature vector $V_d^{z^m}$ of d is mapped to the common vocabulary space V composed with a set of $|V|$ discriminative words (Morchid et al., 2014a) of size 166, to obtain a new feature vector $V_{d,r}^w = \{P(w|d)_r\}_{w \in V}$

of size $|V|$ for the r^{th} topic space r of size n where the i^{th} ($0 \leq i \leq |V|$) feature is:

$$\begin{aligned} V_{d,r}^{w_i} &= P(w_i|d) \\ &= \sum_{j=1}^{n} P(w_i|z_j^r)P(z_j^r|d) \\ &= \sum_{j=1}^{n} V_{z_j^r}^{w_i} \times V_d^{z_j^r} \\ &= \left\langle \overrightarrow{V_{z^r}^{w_i}}, \overrightarrow{V_d^{z^r}} \right\rangle \end{aligned}$$

where $\langle \cdot, \cdot \rangle$ is the inner product, δ being the frequency of the term w_i in d, $V_{z_j^r}^{w_i} = P(w_i|z_j)$ and $V_d^{z_j^r} = P(z_j|d)$ evaluated using Gibbs Sampling in the topic space r.

4 Compact multi-view representation

In this section, an i-vector-based method to represent automatic transcriptions is presented. Initially introduced for speaker recognition, i-vectors (Kenny et al., 2008) have become very popular in the field of speech processing and recent publications show that they are also reliable for language recognition (Martınez et al., 2011) and speaker diarization (Franco-Pedroso et al., 2010). I-vectors are an elegant way of reducing the imput space dimensionality while retaining most of the relevant information. The technique was originally inspired by the Joint Factor Analysis framework (Kenny et al., 2007). Hence, i-vectors convey the speaker characteristics among other information such as transmission channel, acoustic environment or phonetic content of speech segments. The next sections describe the i-vector extraction process, the application of this compact representation to textual documents (called c-vector), and the vector transformation with the EFR method and the Mahalanobis metric.

4.1 Total variability space definition

I-vector extraction could be seen as a probabilistic compression process that reduces the dimensionality of speech super-vectors according to a linear-Gaussian model. The speech (of a given speech recording) super-vector $\mathbf{m}_s$ of concatenated GMM means is projected in a low dimensionality space, named Total Variability space, with:

$$\mathbf{m}_{(h,s)} = m + \mathbf{T}\mathbf{x}_{(h,s)} \ , \qquad (1)$$

where m is the mean super-vector of the UBM[1]. $\mathbf{T}$ is a low rank matrix ($MD \times R$), where M is the number of Gaussians in the UBM and D is the cepstral feature size, which represents a basis of the reduced total variability space. $\mathbf{T}$ is named *Total Variability matrix*; the components of $\mathbf{x}_{(h,s)}$ are the total factors which represent the coordinates of the speech recording in the reduced total variability space called i-vector (i for $i$dentification).

4.2 From i-vector speaker identification to c-vector textual document classification

The proposed approach uses i-vectors to model transcription representation through each topic space in a homogeneous vocabulary space. These short segments are considered as basic semantic-based representation units. Indeed, vector V_d^w represents a segment or a session of a transcription d. In the following, (d, r) will indicate the dialogue representation d in the topic space r. In our model, the segment super-vector $\mathbf{m}_{(d,r)}$ of a transcription d knowing a topic space r is modeled:

$$\mathbf{m}_{(d,r)} = m + \mathbf{T}\mathbf{x}_{(d,r)} \qquad (2)$$

where $\mathbf{x}_{(d,r)}$ contains the coordinates of the topic-based representation of the dialogue in the reduced total variability space called c-vector (c for $c$lassification).

Let $\mathbf{N}_{(d,r)}$ and $\mathbf{X}_{(d,r)}$ be two vectors containing the zero order and first order dialogue statistics respectively. The statistics are estimated against the UBM:

$$\mathbf{N}_r[g] = \sum_{t \in r} \gamma_g(t); \ \{\mathbf{X}_{(d,r)}\}_{[g]} = \sum_{t \in (d,r)} \gamma_g(t) \cdot t \qquad (3)$$

where $\gamma_g(t)$ is the *a posteriori* probability of Gaussian g for the observation t. In the equation, $\sum_{t \in (d,r)}$ represents the sum over all the frames belonging to the dialogue d.

Let $\overline{\mathbf{X}}_{(d,r)}$ be the state dependent statistics defined as follows:

$$\{\overline{\mathbf{X}}_{(d,r)}\}_{[g]} = \{\mathbf{X}_{(d,r)}\}_{[g]} - \mathbf{m}_{[g]} \cdot \sum_{(d,r)} \mathbf{N}_{(d,r)}[g] \qquad (4)$$

Let $\mathbf{L}_{(d,r)}$ be a $R \times R$ matrix, and $\mathbf{B}_{(d,r)}$ a vector

[1]The UBM is a GMM that represents all the possible observations.

Algorithm 1: Estimation algorithm of **T** and latent variable **x**.

For each dialogue d mapped into the topic space r: $x_{(d,r)} \leftarrow 0$, $\mathbf{T} \leftarrow random$;
Estimate statistics: $\mathbf{N}_{(d,r)}$, $\mathbf{X}_{(d,r)}$ (eq.3);
for $i = 1$ *to* $nb\_iterations$ **do**
  for *all* d *and* r **do**
    Center statistics: $\overline{\mathbf{X}}_{(d,r)}$ (eq.4);
    Estimate $\mathbf{L}_{(d,r)}$ and $\mathbf{B}_{(d,r)}$ (eq.5);
    Estimate $\mathbf{x}_{(d,r)}$ (eq.6);
  end
  Estimate matrix **T** (eq. 7 and 8) ;
end

of dimension R, both defined as:

$$\mathbf{L}_{(d,r)} = \mathbf{I} + \sum_{g \in \text{UBM}} \mathbf{N}_{(d,r)}[g] \cdot \{\mathbf{T}\}^t_{[g]} \cdot \mathbf{\Sigma}^{-1}_{[g]} \cdot \{\mathbf{T}\}_{[g]}$$
$$\mathbf{B}_{(d,r)} = \sum_{g \in \text{UBM}} \{\mathbf{T}\}^t_{[g]} \cdot \mathbf{\Sigma}^{-1}_{g} \cdot \{\overline{\mathbf{X}}_{(d,r)}\}_{[g]}, \tag{5}$$

By using $\mathbf{L}_{(d,r)}$ and $\mathbf{B}_{(d,r)}$, $\mathbf{x}_{(d,r)}$ can be obtained using the following equation:

$$\mathbf{x}_{(d,r)} = \mathbf{L}^{-1}_{(d,r)} \cdot \mathbf{B}_{(d,r)} \tag{6}$$

The matrix **T** can be estimated line by line, with $\{\mathbf{T}\}^i_{[g]}$ being the i^{th} line of $\{\mathbf{T}\}_{[g]}$ then:

$$\mathbf{T}^i_{[g]} = \mathbf{L}\mathbf{U}^{-1}_g \cdot \mathbf{R}\mathbf{U}^i_g, \tag{7}$$

where $\mathbf{R}\mathbf{U}^i_g$ and $\mathbf{L}\mathbf{U}_g$ are given by:

$$\mathbf{L}\mathbf{U}_g = \sum_{(d,r)} \mathbf{L}^{-1}_{(d,r)} + \mathbf{x}_{(d,r)}\mathbf{x}^t_{(d,r)} \cdot \mathbf{N}_{(d,r)}[g]$$
$$\mathbf{R}\mathbf{U}^i_g = \sum_{(d,r)} \{\overline{\mathbf{X}}_{(d,r)}\}^{[i]}_{[g]} \cdot \mathbf{x}_{(d,r)} \tag{8}$$

Algorithm 1 presents the method adopted to estimate the multi-view variability dialogue matrix with the above developments where the standard likelihood function can be used to assess the convergence. One can refer to (Matrouf et al., 2007) to find out more about the implementation of the factor analysis.

C-vector representation suffers from 3 raised c-vector issues: (i) the c-vectors x of equation 2 have to be theoretically distributed among the normal distribution $\mathcal{N}(0, I)$, (ii) the "radial" effect should be removed, and (iii) the full rank total factor space should be used to apply discriminant transformations. The next section presents a solution to these 3 problems.

4.3 C-vector standardization

A solution to standardize c-vectors has been developed in (Bousquet et al., 2011). The authors proposed to apply transformations for training and test transcription representations. The first step is to evaluate the empirical mean $\overline{\mathbf{x}}$ and covariance matrix **V** of the training c-vector. Covariance matrix **V** is decomposed by diagonalization into:

$$\mathbf{P}\mathbf{D}\mathbf{P}^T \tag{9}$$

where **P** is the eigenvector matrix of **V** and **D** is the diagonal version of **V**. A training i-vector $\mathbf{x}_{(d,r)}$ is transformed in $\mathbf{x}'_{(d,r)}$ as follows:

$$\mathbf{x}'_{(d,r)} = \frac{\mathbf{D}^{-\frac{1}{2}}\mathbf{P}^T(\mathbf{x}_{(d,r)} - \overline{\mathbf{x}})}{\sqrt{(\mathbf{x}_{(d,r)} - \overline{\mathbf{x}})^T\mathbf{V}^{-1}(\mathbf{x}_{(d,r)} - \overline{\mathbf{x}})}} \tag{10}$$

The numerator is equivalent by rotation to $\mathbf{V}^{-\frac{1}{2}}(\mathbf{x}_{(d,r)} - \overline{\mathbf{x}})$ and the Euclidean norm of $\mathbf{x}'_{(d,r)}$ is equal to 1. The same transformation is applied to the test c-vectors, using the training set parameters $\overline{\mathbf{x}}$ and mean covariance **V**as estimations of the test set of parameters.

Figure 3 shows the transformation steps: Figure 3-(a) is the original training set; Figure 3-(b) shows the rotation applied to the initial training set around the principal axes of the total variability when $\mathbf{P}^T$ is applied; Figure 3-(c) shows the standardization of c-vectors when $\mathbf{D}^{-\frac{1}{2}}$ is applied; and finally, Figure 3-(d) shows the c-vector $\mathbf{x}'_{(d,r)}$ on the surface area of the unit hypersphere after a length normalization by a division of $\sqrt{(\mathbf{x}_{(d,r)} - \overline{\mathbf{x}})^T\mathbf{V}^{-1}(\mathbf{x}_{(d,r)} - \overline{\mathbf{x}})}$.

5 Experimental Protocol

The proposed c-vector representation of automatic transcriptions is evaluated in the context of the theme identification of a human-human telephone conversation in the customer care service (CCS) of the RATP Paris transportation system. The metric used to identify of the best theme is the Mahalanobis metric.

5.1 Theme identification task

The DECODA project corpus (Bechet et al., 2012) was designed to perform experiments on the identification of conversation themes. It is composed of 1,514 telephone conversations, corresponding to about 74 hours of signal, split into a training

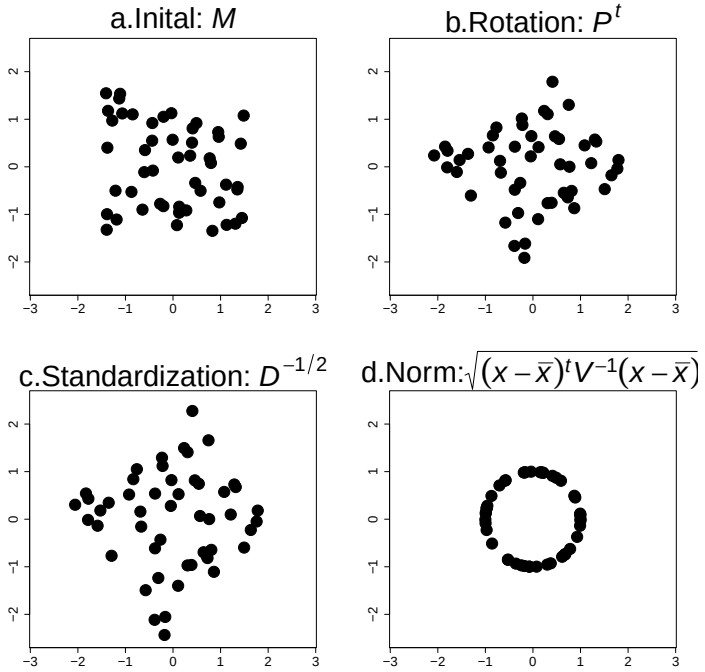


Figure 3: Effect of the standardization with the EFR algorithm.

set (740 dialogues), a development set (175 dialogues) and a test set (327 dialogues), and manually annotated with 8 conversation themes: *problems of itinerary*, *lost and found*, *time schedules*, *transportation cards*, *state of the traffic*, *fares*, *infractions* and *special offers*.

An LDA model allowed us to elaborate 3,000 topics spaces by varying the number of topics from 10 to 3,010. A topic space having less than 10 topics is not suitable for a corpus of more than 700 dialogues (training set). For each theme $\{C_i\}_{i=1}^{8}$, a set of 50 specific words is identified. All the selected words are then merged without repetition to compose V, which is made of 166 words. The topic spaces are made with the LDA Mallet Java implementation[2].

The LIA-Speeral ASR system (Linarès et al., 2007) is used for the experiments. Acoustic model parameters were estimated from 150 hours of speech in telephone conditions. The vocabulary contains 5,782 words. A 3-gram language model (LM) was obtained by adapting a basic LM with the training set transcriptions. A "stop list" of 126 words[3] was used to remove unnecessary words (mainly function words), which results in a Word Error Rate (WER) of 33.8% on the training, 45.2% on the development, and 49.5% on the test. These high WER are mainly due to speech disfluencies and to adverse acoustic environments (for example, calls from noisy streets with mobile phones).

[2]http://mallet.cs.umass.edu/

[3]http://code.google.com/p/stop-words/

5.2 Mahalanobis metric

Given a new observation x, the goal of the task is to identify the theme belonging to x. Probabilistic approaches ignore the process by which c-vectors were extracted and they pretend instead they were generated by a prescribed generative model. Once a c-vector is obtained from a dialogue, its representation mechanism is ignored and it is regarded as an observation from a probabilistic generative model. The Mahalanobis scoring metric assigns a dialogue d with the most likely theme C. Given a training dataset of dialogues, let $\mathbf{W}$ denote the within dialogue covariance matrix defined by:

$$\begin{aligned}\mathbf{W} &= \sum_{k=1}^{K} \frac{n_t}{n} \mathbf{W_k} \\ &= \frac{1}{n} \sum_{k=1}^{K} \sum_{i=0}^{n_t} \left(x_i^k - \overline{x_k}\right) \left(x_i^k - \overline{x_k}\right)^t \end{aligned} \quad (11)$$

where $\mathbf{W_k}$ is the covariance matrix of the k^{th} theme C_k, n_t is the number of utterances for the theme C_k, n is the total number of dialogues, and $\overline{x_k}$ is the centroid (mean) of all dialogues x_i^k of C_k.

Each dialogue does not contribute to the covariance in an equivalent way. For this reason, the term $\frac{n_t}{n}$ is introduced in equation 11. If homoscedasticity (equality of the class covariances) and Gaussian conditional density models are assumed, a new observation x from the test dataset can be assigned to the most likely theme $C_{k_{\text{Bayes}}}$ using the classifier based on the Bayes decision rule:

$$C_{k_{\text{Bayes}}} = \arg\max_k \{\mathcal{N}(x \mid \overline{x_k}, \mathbf{W})\}$$
$$= \arg\max_k \left\{ -\frac{1}{2} (x - \overline{x_k})^t \mathbf{W}^{-1} (x - \overline{x_k}) + a_k \right\}$$

where $\mathbf{W}$ is the within theme covariance matrix defined in eq. 11; $\mathcal{N}$ denotes the normal distribution and $a_k = \log(P(C_k))$. It is noted that, with these assumptions, the Bayesian approach is similar to Fisher's geometric approach: x is assigned to the class of the nearest centroid, according to the Mahalanobis metric (Xing et al., 2002) of $\mathbf{W}^{-1}$:

$$C_{k_{\text{Bayes}}} = \arg\max_k \left\{ -\frac{1}{2} ||x - \overline{x_k}||^2_{\mathbf{W}^{-1}} + a_k \right\}$$

6 Experiments and results

The proposed c-vector approach is applied to the same classification task and corpus proposed in (Morchid et al., 2014a; Morchid et al., 2014b; Morchid et al., 2013) (state-of-the-art in text classification in (Morchid et al., 2014a)). Experiments are conducted using the multiple topic spaces estimated with an LDA approach. From these multiple topic spaces, a classical way is to find the one that reaches the best performance. Figure 4 presents the theme classification performance obtained on the development and test sets using various topic-based representation configurations with the EFR normalization algorithm (*baseline*).

For sake of comparison, experiments are conducted using the automatic transcriptions only (ASR) only. The conditions indicated by the abbreviations between parentheses are considered for the development (Dev) and the test (Test) sets.

Only homogenous conditions (ASR for both training and validations sets) are considered in this study. Authors in (Morchid et al., 2014a) notice that results collapse dramatically when heterogenous conditions are employed (TRS or TRS+ASR for training set and ASR for validation set).

First of all, we can see that this baseline approach reached a classification accuracy of 83% and 76%, respectively on the development and the test sets. However, we note that the classification performance is rather unstable, and may completely change from a topic space configuration to another. The gap between the lower and the higher classification results is also important, with a difference of 25 points on the development set (the same trend is observed on the test set). As a result, finding the best topic space size seems crucial for this classification task, particularly in the context of highly imperfect automatic dialogue transcriptions containing more than one theme.

The topic space that yields the best accuracy with the baseline method ($n = 15$ topics) is presented in Figure 5. This figure presents each of the 15 topics and their 10 most representative words (highest $P(w|z)$). Several topics contain more or less the same representative words, such as topics 3, 6 and 9. This figure points out some interesting topics that allow us to distinguish a theme from the others. For example:

- topics 2, 10 and 15 represent some words related to *itinerary problems*,
- the *transportation cards* theme is mostly represented in topic 4 and 15 (*Imagine* and *Navigo* are names of transportation cards),
- the words which represent the *time schedules* theme are contained in topic 5,6,7 and less in topic 9,
- *state of the traffic* could be discussed with words such as: *departure, line, service, day*. These words and others are contained in topic 13,
- topics 4 and 12 are related to the *infractions* theme with to words *fine*, *pass*, *zone* or *ticket*,
- but topic 12 could be related to theme *fares* or *special offers* as well .

Table 1 presents results obtained with the proposed c-vector approach coupled with the EFR algorithm. We can firstly note that this compact representation allows it to outperform the best topic space configuration (*baseline*), with a gain of 9.4 points on the development data and of 9 points on the test data. Moreover, if we consider the different c-vector configurations with the development

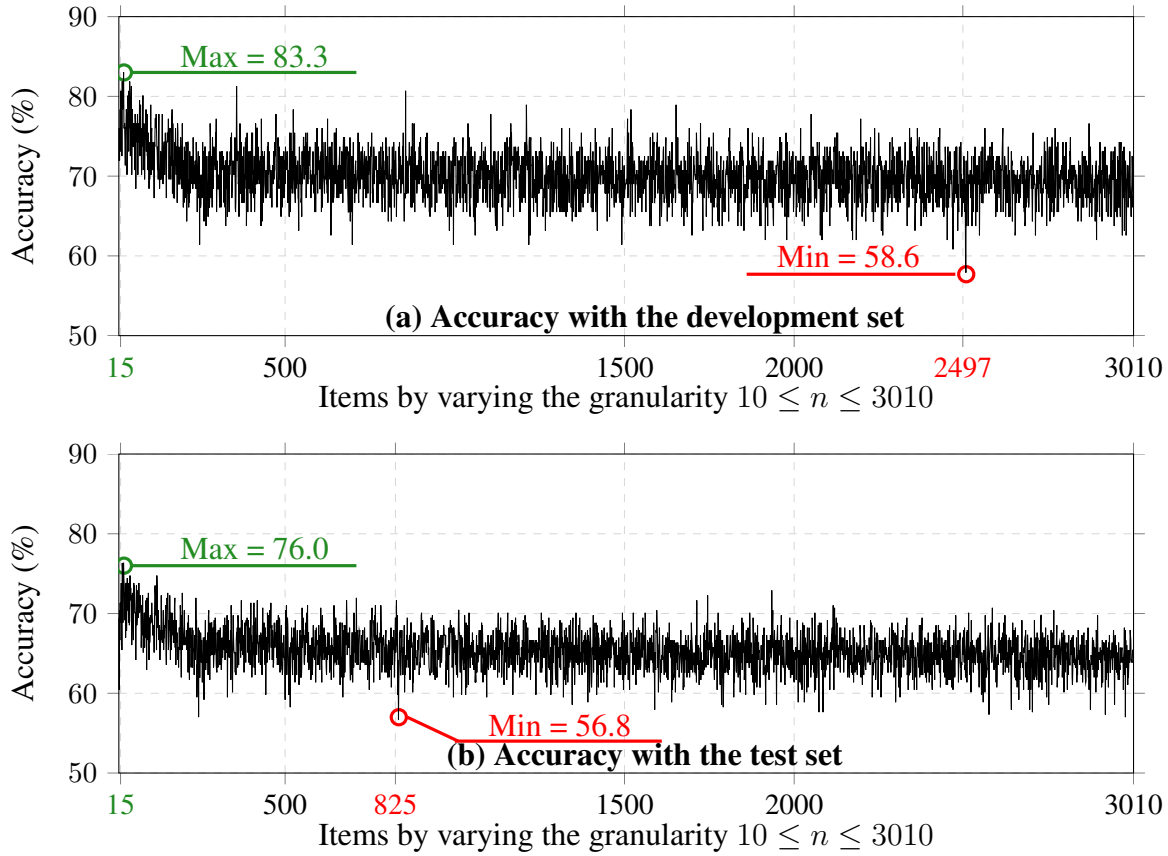


Figure 4: Theme classification accuracies using various topic-based representations with EFR normalization (baseline) on the development and test sets (X-coordinates start at 10 indeed, but to show the best configuration point (15), the origine (10) has been removed).

Table 1: Theme classification accuracy (%) with different c-vectors and GMM-UBM sizes.

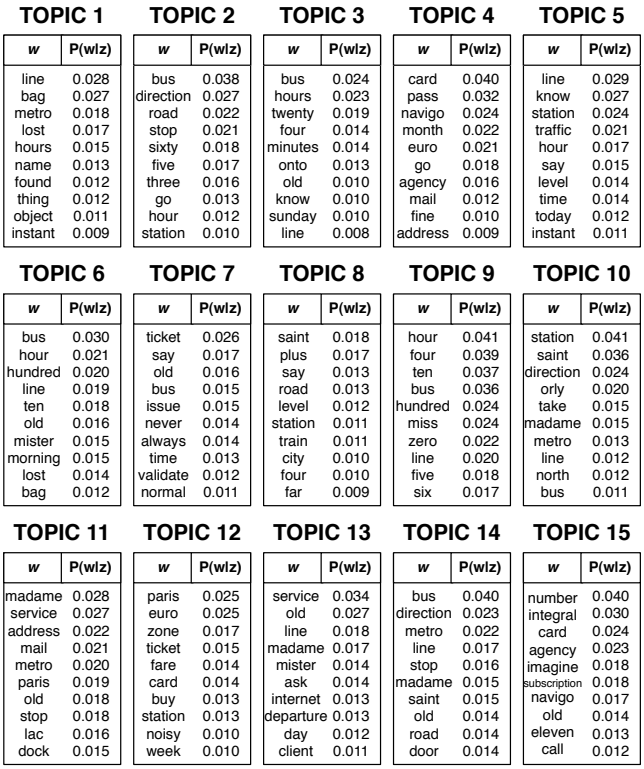
	DEV				*TEST*			
c-vector size	**Number of Gaussians in GMM-UBM**							
	32	*64*	*128*	*256*	*32*	*64*	*128*	*256*
60	88.8	*86.5*	91.2	90.6	85.0	*82.6*	83.5	84.7
100	91.2	**92.4**	92.4	87.7	86.0	**85.0**	83.5	84.7
120	89.5	92.2	89.5	87.7	85.0	83.5	85.4	84.1

Table 2: Maximum (Max), minimum (Min) and Difference ($Max - Min$) theme classification accuracies (%) using the baseline and the proposed c-vector approaches.

	Max		**Min**		**Difference**	
Method	*DEV*	*TEST*	*DEV*	*TEST*	*DEV*	*TEST*
baseline	83.3	76.0	58.6	56.8	14.7	20.8
c-vector	**92.4**	**85.0**	86.5	82.6	5.9	2.4

and test sets, the gap between accuracies is much smaller: classification accuracy does not go below 82.6%, while it reached 56% for the worst topic-based configuration. Indeed, as shown in Table 2, the difference between the maximum and the minimum theme classification accuracies is of 20% using the baseline approach while it is only of 2.4% using the c-vector method.

We can conclude that this original c-vector approach allows one to better handle the variabilities



TOPIC 1		TOPIC 2		TOPIC 3		TOPIC 4		TOPIC 5	
w	P(w|z)	*w*	P(w|z)	*w*	P(w|z)	*w*	P(w|z)	*w*	P(w|z)
line	0.028	bus	0.038	bus	0.024	card	0.040	line	0.029
bag	0.027	direction	0.027	hours	0.023	pass	0.032	know	0.027
metro	0.018	road	0.022	twenty	0.019	navigo	0.024	station	0.024
lost	0.017	stop	0.021	four	0.014	month	0.022	traffic	0.021
hours	0.015	sixty	0.018	minutes	0.014	euro	0.021	hour	0.017
name	0.013	five	0.017	onto	0.013	go	0.018	say	0.015
found	0.012	three	0.016	old	0.010	agency	0.016	level	0.014
thing	0.012	go	0.013	know	0.010	mail	0.012	time	0.014
object	0.011	hour	0.012	sunday	0.010	fine	0.010	today	0.012
instant	0.009	station	0.010	line	0.008	address	0.009	instant	0.011

TOPIC 6		TOPIC 7		TOPIC 8		TOPIC 9		TOPIC 10	
w	P(w|z)	*w*	P(w|z)	*w*	P(w|z)	*w*	P(w|z)	*w*	P(w|z)
bus	0.030	ticket	0.026	saint	0.018	hour	0.041	station	0.041
hour	0.021	say	0.017	plus	0.017	four	0.039	saint	0.036
hundred	0.020	old	0.016	say	0.013	ten	0.037	direction	0.024
line	0.019	bus	0.015	road	0.013	bus	0.036	orly	0.020
ten	0.018	issue	0.015	level	0.012	hundred	0.024	take	0.015
old	0.016	never	0.014	station	0.011	miss	0.024	madame	0.015
mister	0.015	always	0.014	train	0.011	zero	0.022	metro	0.013
morning	0.015	time	0.013	city	0.010	line	0.020	line	0.012
lost	0.014	validate	0.012	four	0.010	five	0.018	north	0.012
bag	0.012	normal	0.011	far	0.009	six	0.017	bus	0.011

TOPIC 11		TOPIC 12		TOPIC 13		TOPIC 14		TOPIC 15	
w	P(w|z)	*w*	P(w|z)	*w*	P(w|z)	*w*	P(w|z)	*w*	P(w|z)
madame	0.028	paris	0.025	service	0.034	bus	0.040	number	0.040
service	0.027	euro	0.025	old	0.027	direction	0.023	integral	0.030
address	0.022	zone	0.017	line	0.018	metro	0.022	card	0.024
mail	0.021	ticket	0.015	madame	0.017	line	0.017	agency	0.023
metro	0.020	fare	0.014	mister	0.014	stop	0.016	imagine	0.018
paris	0.019	card	0.014	ask	0.014	madame	0.015	subscription	0.018
old	0.018	buy	0.013	internet	0.013	saint	0.015	navigo	0.017
stop	0.018	station	0.013	departure	0.013	old	0.014	old	0.014
lac	0.016	noisy	0.010	day	0.012	road	0.014	eleven	0.013
dock	0.015	week	0.010	client	0.011	door	0.014	call	0.012

Figure 5: Topic space (15 topics) that obtains the best accuracy with the baseline system (see Fig. 4).

contained in dialogue conversations: in a classification context, better accuracy can be obtained and the results can be more consistent when varying the c-vector size and the number of Gaussians.

7 Conclusions

This paper presents an original multi-view representation of automatic speech dialogue transcriptions, and a fusion process with the use of a factor analysis method called i-vector. The first step of the proposed method is to represent a dialogue in multiple topic spaces of different sizes (*i.e.* number of topics). Then, a compact representation of the dialogue from the multiple views is processed to compensate the vocabulary and the variability of the topic-based representations. The effectiveness of the proposed approach is evaluated in a classification task of theme dialogue identification. Thus, the architecture of the system identifies conversation themes using the i-vector approach. This compact representation was initially developed for speaker recognition and we showed that it can be successfully applied to a text classification task. Indeed, this solution allowed the system to obtain better classification accuracy than with the use of the classical best topic space configuration. In fact, we highlighted that this original compact version of all topic-based representations of dialogues, called c-vector in this work, coupled with the EFR normalization algorithm, is a better solution to deal with dialogue variabilities (high word error rates, bad acoustic conditions, unusual word vocabulary, etc). This promising compact representation allows us to effectively solve both the difficult choice of the right number of topics and the multi-theme representation issue of particular textual documents. Finally, the classification accuracy reached 85% with a gain of 9 points compared to usual baseline (best topic space configuration). In a future work, we plan to evaluate this new representation of textual documents in other information retrieval tasks, such as keyword extraction or automatic summarization systems.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. This work was funded by the SUMACC and ContNomina projects supported by the French National Research Agency (ANR) under contracts ANR-10-CORD-007 and ANR-12-BS02-0009.

References

R. Arun, Venkatasubramaniyan Suresh, C.E. Veni Madhavan, and Musti Narasimha Murty. 2010. On finding the natural number of topics with latent dirichlet allocation: Some observations. In *Advances in Knowledge Discovery and Data Mining*, pages 391–402. Springer.

Frederic Bechet, Benjamin Maza, Nicolas Bigouroux, Thierry Bazillon, Marc El-Beze, Renato De Mori, and Eric Arbillot. 2012. Decoda: a call-centre human-human spoken conversation corpus. LREC'12.

Jerome R. Bellegarda. 1997. A latent semantic analysis framework for large-span language modeling. In *Fifth European Conference on Speech Communication and Technology*.

Jerome R. Bellegarda. 2000. Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE*, 88(8):1279–1296.

David M. Blei and John Lafferty. 2006. Correlated topic models. *Advances in neural information processing systems*, 18:147.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Pierre-Michel Bousquet, Driss Matrouf, and Jean-François Bonastre. 2011. Intersession compensation and scoring methods in the i-vectors space for speaker recognition. In *Interspeech*, pages 485–488.

Juan Cao, Tian Xia, Jintao Li, Yongdong Zhang, and Sheng Tang. 2009. A density-based method for adaptive lda model selection. *Neurocomputing*, 72(7):1775–1781.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Najim Dehak, Patrick J. Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. 2011. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798.

Javier Franco-Pedroso, Ignacio Lopez-Moreno, Doroteo T Toledano, and Joaquin Gonzalez-Rodriguez. 2010. Atvs-uam system description for the audio segmentation and speaker diarization albayzin 2010 evaluation. In *FALA VI Jornadas en Tecnologa del Habla and II Iberian SLTech Workshop*, pages 415–418.

Daniel Garcia-Romero and Carol Y Espy-Wilson. 2011. Analysis of i-vector length normalization in speaker recognition systems. In *Interspeech*, pages 249–252.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):721–741.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235.

Gregor Heinrich. 2005. Parameter estimation for text analysis. *Web: http://www. arbylon. net/publications/text-est. pdf*.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI ' 99*, page 21. Citeseer.

Thomas Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177–196.

Patrick Kenny, Gilles Boulianne, Pierre Ouellet, and Pierre Dumouchel. 2007. Joint factor analysis versus eigenchannels in speaker recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4):1435–1447.

Patrick Kenny, Pierre Ouellet, Najim Dehak, Vishwa Gupta, and Pierre Dumouchel. 2008. A study of interspeaker variability in speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(5):980–988.

Wei Li and Andrew McCallum. 2006. Pachinko allocation: Dag-structured mixture models of topic correlations.

Georges Linarès, Pascal Nocéra, Dominique Massonie, and Driss Matrouf. 2007. The lia speech recognition system: from 10xrt to 1xrt. In *Text, Speech and Dialogue*, pages 302–308. Springer.

David Martınez, Oldrich Plchot, Lukás Burget, Ondrej Glembek, and Pavel Matejka. 2011. Language recognition in ivectors space. *Interspeech*, pages 861–864.

Driss Matrouf, Nicolas Scheffer, Benoit G.B. Fauve, and Jean-Francois Bonastre. 2007. A straightforward and efficient implementation of the factor analysis model for speaker verification. In *Interspeech*, pages 1242–1245.

Thomas Minka and John Lafferty. 2002. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 352–359. Morgan Kaufmann Publishers Inc.

Mohamed Morchid, Georges Linarès, Marc El-Beze, and Renato De Mori. 2013. Theme identification in telephone service conversations using quaternions of speech features. In *Interspeech*. ISCA.

Mohamed Morchid, Richard Dufour, Pierre-Michel Bousquet, Mohamed Bouallegue, Georges Linarès, and Renato De Mori. 2014a. Improving dialogue classification using a topic space representation and a gaussian classifier based on the decision rule. In *ICASSP*. IEEE.

Mohamed Morchid, Richard Dufour, and Georges Linarès. 2014b. A LDA-Based Topic Classification Approach from Highly Imperfect Automatic Transcriptions. In *LREC*.

Douglas A. Reynolds and Richard C. Rose. 1995. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, 3(1):72–83.

Gerard Salton. 1989. Automatic text processing: the transformation. *Analysis and Retrieval of Information by Computer*.

Yoshimi Suzuki, Fumiyo Fukumoto, and Yoshihiro Sekiguchi. 1998. Keyword extraction using term-domain interdependence for dictation of radio news. In *17th international conference on Computational linguistics*, volume 2, pages 1272–1276. ACL.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2004. Sharing clusters among related groups: Hierarchical dirichlet processes. In *NIPS*.

Eric P. Xing, Michael I. Jordan, Stuart Russell, and Andrew Ng. 2002. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512.

Elias Zavitsanos, Sergios Petridis, Georgios Paliouras, and George A. Vouros. 2008. Determining automatically the size of learned ontologies. In *ECAI*, volume 178, pages 775–776.

Explaining the Stars: Weighted Multiple-Instance Learning for Aspect-Based Sentiment Analysis

Nikolaos Pappas
EPFL and Idiap Research Institute
Rue Marconi 19
CH-1920 Martigny, Switzerland
nikolaos.pappas@idiap.ch

Andrei Popescu-Belis
Idiap Research Institute
Rue Marconi 19
CH-1920 Martigny, Switzerland
andrei.popescu-belis@idiap.ch

Abstract

This paper introduces a model of multiple-instance learning applied to the prediction of aspect ratings or judgments of specific properties of an item from user-contributed texts such as product reviews. Each variable-length text is represented by several independent feature vectors; one word vector per sentence or paragraph. For learning from texts with known aspect ratings, the model performs multiple-instance regression (MIR) and assigns importance weights to each of the sentences or paragraphs of a text, uncovering their contribution to the aspect ratings. Next, the model is used to predict aspect ratings in previously unseen texts, demonstrating interpretability and explanatory power for its predictions. We evaluate the model on seven multi-aspect sentiment analysis data sets, improving over four MIR baselines and two strong bag-of-words linear models, namely SVR and Lasso, by more than 10% relative in terms of MSE.

1 Introduction

Sentiment analysis of texts provides a coarse-grained view of their overall attitude towards an item, either positive or negative. The recent abundance of user texts accompanied by real-valued labels e.g. on a 5-star scale has contributed to the development of automatic sentiment analysis of reviews of items such as movies, books, music or other products, with applications in social computing, user modeling, and recommender systems. The overall sentiment of a text towards an item often results from the ratings of several specific aspects of the item. For instance, the author of a review might have a rather positive sentiment about a movie, having particularly liked the plot and the music, but not too much the actors. Determining the ratings of each aspect automatically is a challenging task, which may seem to require the engineering of a large number of features designed to capture each aspect. Our goal is to put forward a new feature-agnostic solution for analyzing aspect-related ratings expressed in a text, thus aiming for a finer-grained, deeper analysis of text meaning than overall sentiment analysis.

Current state-of-the-art approaches to sentiment analysis and aspect-based sentiment analysis, attempt to go beyond word-level features either by using higher-level linguistic features such as POS tagging, parsing, and knowledge infusion, or by learning features that capture syntactic and semantic dependencies between words. Once an appropriate feature space is found, the ratings are typically modeled using a linear model, such as Support Vector Regression (SVR) with ℓ_2 norm for regularization or Lasso Regression with ℓ_1 norm. By treating a text globally, these models ignore the fact that the sentences of a text have diverse contributions to the overall sentiment or to the attitude towards a specific aspect of an item.

In this paper, we propose a new learning model which answers the following question: "To what extent does each part of a text contribute to the prediction of its overall sentiment or the rating of a particular aspect?" The model uses multiple-instance regression (MIR), based on the assumption that not all the parts of a text have the same contribution to the prediction of the rating. Specifically, a text is seen as a bag of sentences (instances), each of them modeled as a word vector. The overall challenge is to learn which sentences refer to a given aspect, and how they contribute to the text's attitude towards it, but the model applies to overall sentiment analysis as well. For instance, Figure 1 displays a positive global comment on a TED talk and the weights assigned to two of its sentences by MIR.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 455–466,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

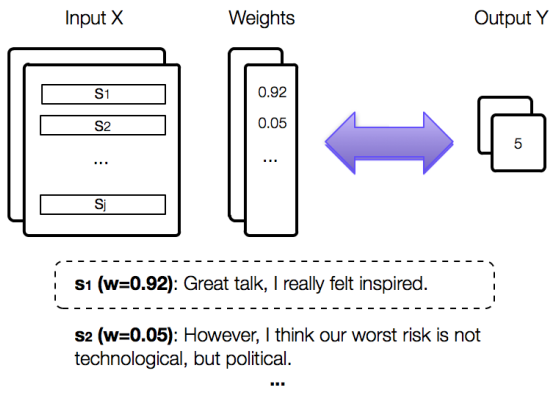


Figure 1: Analysis of a comment (bag of sentences $\{s_1, ..., s_j\}$) annotated by humans with the maximal positive sentiment score (5 stars). The weights assigned by MIR reveal that s_1 has the greatest relevance to the overall sentiment.

Using regularized least squares, we formulate an optimization objective to jointly assign instance weights and regression hyperplane weights. Then, an instance relevance estimation method is used to predict aspect ratings, or global ones, in previously unseen texts. The parameters of the model are learned using an alternating optimization procedure inspired by Wagstaff and Lane (2007). Our model requires only text with ratings for training, with no particular assumption on the word features to be extracted, and provides interpretable explanations of the predicted ratings through the relevance weights assigned to sentences. We also show that the model has reasonable computational demands. The model is evaluated on aspect and sentiment rating prediction over seven datasets: five of them contain reviews with aspect labels about beers, audiobooks and toys (McAuley et al., 2012), and two contain TED talks with emotion labels, and comments on them with sentiment labels (Pappas and Popescu-Belis, 2013). Our model outperforms previous MIR models and two strong linear models for rating prediction, namely SVR and Lasso by more than 10% relative in terms of MSE. The improvement is observed even when the sophistication of the feature space increases.

The paper is organized as follows. Section 2 shows how our model innovates with respect to previous work on MIR and rating prediction. Section 3 formulates the problem while Section 4 describes previous MIR models. Section 5 presents our MIR model and learning procedure. Section 6 presents the datasets and evaluation methods. Section 7 reports our results on rating prediction tasks, and provides examples of rating explanation.

2 Related Work

2.1 Multiple-Instance Regression

Multiple-instance regression (MIR) belongs to the class of multiple-instance learning (MIL) problems for real-valued output, and it is a variant of multiple regression where each data point may be described by more than one vectors of values. Many MIL studies focused on classification (Andrews et al., 2003; Bunescu and Mooney, 2007; Settles et al., 2008; Foulds and Frank, 2010; Wang et al., 2011) while fewer focused on regression (Ray and Page, 2001; Davis and others, 2007; Wagstaff et al., 2008; Wagstaff and Lane, 2007). Related to document analysis, several MIR studies have focused on news categorization (Zhang and Zhou, 2008; Zhou et al., 2009) or web-index recommendation (Zhou et al., 2005) but, to our knowledge, no study has attempted to use MIR for aspect rating prediction or sentiment analysis with real-valued labels.

MIR was firstly introduced by Ray et al. (2001), proposing an EM algorithm which assumes that one primary instance per bag is responsible for its label. Wagstaff and Lane (2007) proposed to simultaneously learn a regression model and estimate instance weights per bag for crop yield modeling (not applicable to prediction). A similar method which learns the internal structure of bags using clustering was proposed by Wagstaff et al. (2008) for crop yield prediction, and we will use it for comparison in the present study. Later, the method was adapted to map bags into a single-instance feature space by Zhang et al. (2009). Wang et al. (2008) assumed that each bag is generated by random noise around a primary instance, while Wang et al. (2012) represented bag labels with a probabilistic mixture model. Foulds et al. (2010) concluded that various assumptions are differently suited to different tasks, and should be stated clearly when describing an MIR model.

2.2 Rating Prediction from Text

Sentiment analysis aims at analyzing the polarity of a given text, either with classification (for discrete labels) or regression (for real-valued labels). Early studies introduced machine learning techniques for sentiment classification, e.g. Pang et al. (2002), including unsupervised techniques based on the notion of semantic orientation of phrases, e.g. Turney et al. (2002). Other studies focused on subjectivity detection, i.e. whether a

text span expresses opinions or not (Wiebe et al., 2004). Rating inference was defined by Pang et al. (2005) as multi-class classification or regression with respect to rating scales. Pang and Lee (2008) discusses the large range of features engineered for this task, though several recent studies focus on feature learning (Maas et al., 2011; Socher et al., 2011), including the use of a deep neural network (Socher et al., 2013). In contrast, we do not make any assumption about the nature or dimensionality of the feature space.

The fine-grained analysis of opinions regarding specific aspects or features of items is known as *multi-aspect sentiment analysis*. This task usually requires aspect-related text segmentation, followed by prediction or summarization (Hu and Liu, 2004; Zhuang et al., 2006). Most attempts to perform this task have engineered various feature sets, augmenting words with topic or content models (Mei et al., 2007; Titov and McDonald, 2008; Sauper et al., 2010; Lu et al., 2011), or with linguistic features (Pang and Lee, 2005; Baccianella et al., 2009; Qu et al., 2010; Zhu et al., 2012). Other studies have advocated joint modeling of multiple aspects (Snyder and Barzilay, 2007) or multiple reviews for the same product (Li et al., 2011). McAuley et al. (2012) introduced new corpora of multi-aspect reviews, which we also partly use here, and proposed models for aspect detection, sentiment summarization and rating prediction. Lastly, joint aspect identification and sentiment classification have been used for aggregating product review snippets by Sauper at al. (2013). None of the above studies considers the multiple-instance property of text in their modeling.

3 MIR Definition

Let us consider a set B of m bags with numerical labels Y as input data $D = \{(\{b_{1j}\}_{n_1}^d, y_1), ..., (\{b_{mj}\}_{n_m}^d, y_m)\}$, where $b_{ij} \in \mathbb{R}^d$ (for $1 \leq j \leq n_i$) and $y_i \in \mathbb{R}$. Each bag B_i consists of n_i data points (called 'instances'), hence it is a matrix of n_i d-dimensional vectors, e.g. word vectors. The challenge is to infer the label of the bag given a variable number of instances n_i. This requires finding a set of bag representations $X = \{x_1, \ldots, x_m\}$ of size m where $x_i \in \mathbb{R}^d$, from which the class labels can be computed. The goal is then to find a mapping from this representation, noted $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}$, which is able to predict the label of a given bag. Ideally, assuming that X is the best bag representation for our task, we look for the optimal regression hyperplane Φ which minimizes a loss function $\mathcal{L}$ plus a regularization term Ω as follows:

$$\Phi = \underset{\Phi}{arg\,min} \left(\underbrace{\mathcal{L}(Y, X, \Phi)}_{\text{loss}} + \underbrace{\Omega(\Phi)}_{\text{reg.}} \right) \quad (1)$$

Since the best set of representations X for a task is generally unknown, one has to make assumptions to define it or compute it jointly with the regression hyperplane Φ. Thus, the main difficulty lies in finding a good assumption for X, as we will now discuss.

4 Previous MIR Assumptions

We describe here three assumptions frequently made in past MIR studies, to which we will later compare our model: aggregating all instances, keeping them as separate examples, or choosing the most representative one (Wang et al., 2012). For each assumption, we will experiment with two state-of-the-art regression models (noted abstractly as f), namely SVR (Drucker et al., 1996) and Lasso (Tibshirani, 1996) with respectively the ℓ_2 and ℓ_1 norms for regularization.

The *Aggregated* algorithm assumes that each bag is represented as a single d-dimensional vector, which is the average of its instances (hence $x_i \in \mathbb{R}^d$). Then, a regression model f is trained on pairs of vectors and class labels, $D_{agg} = \{(x_i, y_i) \mid i = 1, \ldots, m\}$, and the predicted class of an unlabeled bag $B_i = \{b_{ij} \mid j = 1, \ldots, n_i\}$ is computed as follows:

$$\hat{y}(B_i) = f(mean(\{b_{ij} \mid j = 1, \ldots, n_i\})) \quad (2)$$

In fact, a simple sum can also be used instead of the mean, and we observed in practice that with an appropriate regularization there is no difference on the prediction performance between these options. This baseline corresponds to the typical approach for text regression tasks, where each text sample is represented by a single vector in the feature space (e.g. BOW with counts or TF-IDF weights).

The *Instance* algorithm considers each of the instances in a bag as separate examples, by labeling each of them with the bag's label. A regression model f is learned over the training set made of all vectors of all bags, $D_{ins} = \{(b_{ij}, y_i) \mid j = 1, \ldots, n_i; i = 1, \ldots, m\}$, assuming that there are m labeled bags. To label a new bag B_i, given that

there is no representation x_i, the method simply averages the predicted labels of its instances:

$$\hat{y}(B_i) = mean(\{f(b_{ij}) \mid j = 1, \ldots, n_i\}) \quad (3)$$

Instead of the average, the median value can also be used, which is more appropriate when the bags contain outlying instances.

The *Prime* algorithm assumes that a single instance in each bag is responsible for its label (Ray and Page, 2001). This instance is called the primary or prime one. The method is similar to the previous one, except that only one instance per bag is used as training data: $D_{pri} = \{(b_i^p, y_i) \mid i = 1, \ldots, m\}$, where b_i^p is the prime instance of the i^{th} bag B_i and m is the number of bags. The prime instances are discovered through an iterative algorithm which refines the regression model f. Given an initial model f, in each iteration the algorithm selects from each bag a prime candidate which is the instance with the lowest prediction error. Then, a new model is trained over the selected prime candidates, until convergence. For a new bag, the target class is computed as in Eq. 3.

5 Proposed MIR Model

We propose a new MIR model which assigns individual relevance values (weights) to each instance of a bag, thus making fewer simplifying assumptions than previous models. We extend instance-relevance algorithms such as (Wagstaff and Lane, 2007) by supporting high-dimensional feature spaces, as required for text regression, and by predicting both the class label and the content structure of previously unseen (hence unlabeled) bags. The former is achieved by minimizing a regularized least squares loss (RLS) instead of solving normal equations, which is prohibitive in large spaces. The latter represents a significant improvement over *Aggregated* and *Instance* algorithms, which are unable to pinpoint the most relevant instances with respect to the label of each bag, being thus applicable only to bag label prediction. Similarly, *Prime* only identifies the prime instance when the bag is already labeled. Instead, our model learns an optimal method to aggregate instances, rather than a pre-defined one, and allows more degrees of freedom in the regression model than previous ones. Moreover, the weight of an instance is interpreted as its relevance both in training and prediction.

5.1 Instance Relevance Assumption

Each bag defines a bounded region of a hyperplane orthogonal to the y-axis (the envelope of all its points). The goal is to find a regression hyperplane that passes through each bag B_i and to predict its label by using at least one data point x_i within that bounded region. Thus, the point x_i is a convex combination of the points in the bag, in other words B_i is represented by the weighted average of its instances b_{ij}:

$$x_i = \sum_{j=1}^{n_i} \psi_{ij} b_{ij}, \psi_{ij} \geq 0 \text{ and } \sum_{j=1}^{n_i} \psi_{ij} = 1 \quad (4)$$

where ψ_{ij} is the weight of the j^{th} instance of the i^{th} bag. Each weight ψ_{ij} indicates the relevance of an instance j to the prediction of the class y_i of the i^{th} bag. The constraint forces x_i to fall within the bounded region of the points in bag i and guarantees that the i^{th} bag will influence the regressor.

5.2 Modeling Bag Structure and Labels

Let us consider a set of m bags, where each bag B_i is represented by its n_i d-dimensional instances, i.e. $B_i = \{b_{ij}\}_{n_i}^d$ along with the set of target class labels for each bag, $Y = \{y_i\}_N, y_i \in \mathbb{R}$. The representation set of all B_i in the feature space, $X = \{x_1, \ldots, x_m\}, x_i \in \mathbb{R}^d$, is obtained using the n_i instance weights associated to each bag B_i, $\psi_i = \{\psi_{ij}\}_{n_i}$, $\psi_{ij} \in [0, 1]$ which are initially unknown. Thus, we look for a linear regression model f that is able to model the target values using the regression coefficients $\Phi \in \mathbb{R}^d$, where X and Y are respectively the sets of training bags and their labels: $Y = f(X) = \Phi^T X$. We define a loss function according to the least squares objective dependent on X, Y, Φ and the set of weight vectors $\Psi = \{\psi_1, \ldots, \psi_m\}$ using Eq. 4 as follows:

$$\begin{aligned}\mathcal{L}(Y, X, \Psi, \Phi) &= ||Y - \Phi^T X||_2^2 \\ &\overset{(4)}{=} \sum_{i=1}^{N} \Big(y_i - \Phi^T \big(\sum_{j=1}^{n_i} \psi_{ij} b_{ij}\big)\Big)^2 \\ &= \sum_{i=1}^{N} \Big(y_i - \Phi^T (B_i \psi_i)\Big)^2 \end{aligned} \quad (5)$$

Using the above loss function, accounting for the constraints of our assumption in Eq. 4 and assuming ℓ_2-norm for regularization with ϵ_1 and ϵ_2 terms for each $\psi_i \in \Psi$ and Φ respectively, we obtain the

following least squares objective from Eq. 1:

$$\underset{\psi_1,\ldots,\psi_m,\Phi}{arg\,min} \underbrace{\sum_{i=1}^{m}\Big(\underbrace{\Delta_i^2}_{f_1 \text{ loss}} + \underbrace{\epsilon_1||\psi_i||}_{f_1 \text{ reg.}}\Big)}_{f_2 \text{ loss}} + \underbrace{\epsilon_2||\Phi||^2}_{f_2 \text{ reg.}}$$

$$\text{where } \Delta_i^2 = \Big(y_i - \Phi^T(B_i\psi_i)\Big)^2, \qquad (6)$$

subject to $\psi_{ij} \geq 0\ \forall i,j$ and $\sum_{j=1}^{n_i}\psi_{ij} = 1\ \forall i$. The selection of the ℓ_2-norm was based on preliminary results showing that it outperforms ℓ_1-norm. Other combinations of p-norm regularization can be explored for f_1 and f_2, e.g. to learn sparser instance weights and denser regression coefficients or vice versa.

The above objective is non-convex and difficult to optimize because the minimization is with respect to all $\psi_1,\ldots,\psi_m$ and Φ at the same time. As indicated in Eq. 6 above, we will note f_1 a model that is learned from the minimization only with respect to $\psi_1,\ldots,\psi_m$ and f_2 a model obtained from the minimization with respect to Φ only. In Eq. 6, we can observe that if one of the two is known or held fixed, then the other one is convex and can be learned with the well-known least squares solving techniques. In Section 5.3, we will describe an algorithm that is able to exploit this observation.

Having computed $\psi_1,\ldots,\psi_m$ and Φ, we could predict a label for an unlabeled bag using Eq. 3, but would not be able to compute the weights of the instances. Moreover, information that has been learned about the instances during the training phase would not be used during prediction. For these reasons, we introduce a third regression model f_3 with regression coefficients $O \in \mathbb{R}^d$ assuming a ℓ_2-norm for the regularization with ϵ_3 term, which is trained on the relevance weights obtained from the Eq. 6, $D_w = \{(b_{ij},\psi_{ij}) \mid i = 1,...,m; j = 1,...,n_i\}$. The optimization objective for the f_3 model is the following:

$$\underset{O}{arg\,min} \underbrace{\sum_{i=1}^{N}\sum_{j=1}^{n_i}\big(\psi_{ij} - O^T b_{ij}\big)^2}_{f_3 \text{ loss function}} + \underbrace{\epsilon_3||O||^2}_{f_3 \text{ reg.}} \qquad (7)$$

This minimization can be easily performed with the well-known least squares solving techniques. The learned model is able to estimate the weights of the instances of an unlabeled bag during prediction time as: $\hat{\psi}_i = f_3(B_i) = \Omega^T B_i$. The $\hat{\psi}_i$ weights are estimations which are influenced by the relevance weights learned in our minimization objective of Eq. 6 but they are not constrained at prediction time. To obtain interpretable weights, we can convert the estimated scores to the $[0,1]$ interval as follows: $\hat{\psi}_i = \hat{\psi}_i / sum(\hat{\psi}_i)$. Finally, the prediction of the label for the i^{th} bag using the estimated instance weights $\hat{\psi}_i$ is done as follows:

$$\hat{y} = f_2(B_i) = \Phi^T B_i \hat{\psi}_i \qquad (8)$$

5.3 Learning with Alternating Projections

Algorithm 1 solves the non-convex optimization problem of Eq. 6 by using a powerful class of methods for finding the intersection of convex sets, namely alternating projections (AP). The problem is firstly divided into two convex problems, namely f_1 loss function and f_2 loss function, which are then solved in an alternating fashion. Like EM algorithms, AP algorithms do not have general guarantees on their convergence rate, although, in practice, we found it acceptable at generally fewer than 20 iterations.

Algorithm 1 APWeights(B, Y, ϵ_1, ϵ_2, ϵ_3)

```
Initialize(ψ_1, ..., ψ_N, Φ, X)
while not converged do
    for B_i in B do
        ψ_i = cRLS(Φ^T Bi, Y_i, ε_1) # f_1 model
        x_i = B_i ψ_i^T
    end for
    Φ = RLS(X, Y, ε_2) # f_2 model
end while
Ω = RLS({b_ij ∀i,j}, {ψ_ij ∀i,j}, ε_3) # f_3 model
```

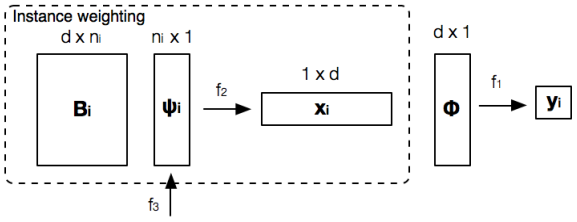


Figure 2: Visual representation for the training and testing procedure of Algorithm 1.

The algorithm takes as input the bags B_i, their target class labels Y and the regularization terms $\epsilon_1, \epsilon_2, \epsilon_3$ and proceeds as follows. First, under a fixed regression model (f_2), it proceeds with f_1 to the optimal assignment of weights to the instances of each bag (projection of Φ vectors on the ψ_i space which is a n_i-simplex) and computes its new representation set X. Second, given the fixed instance weights, it trains a new regression model (f_2) using X (projection back to the Φ

	Bags		Instances		Dimension	Aspect ratings
Dataset	**Type**	**Count**	**Type**	**Count**	**Count**	**Classes**
BeerAdvocate	review	1,200	sentence	12,189	19,418	feel, look, smell, taste, overall
RateBeer (ES)		1,200		3,269	2,120	appearance, aroma, overall, palate, taste
RateBeer (FR)		1,200		4,472	903	appearance, aroma, overall, palate, taste
Audiobooks		1,200		4,886	3,971	performance, story, overall
Toys & Games		1,200		6,463	31,984	educational, durability, fun, overall
TED comments	comment	1,200	sentence	3,814	957	sentiment (polarity)
TED talks	comments per talk	1,200	comment	11,993	5,000	unconvincing, fascinating, persuasive, ingenious, longwinded, funny, inspiring, jaw-dropping, courageous, beautiful, confusing, obnoxious

Table 1: Description of the seven datasets used for aspect, sentiment and emotion rating prediction.

space). This procedure repeats until convergence, i.e. when there is no more decrease on the training error, or until a maximum number of iterations has been reached. The regression model f_3 is trained on the weights learned from the previous steps.

5.4 Complexity Analysis

The overall time complexity T of Algorithm 1 in terms of the input variables, noted $h = \{m, \hat{n}, d\}$, with m being the number of bags, $\hat{n}$ the average size of the bags, and d the dimensionality of the feature space (here, the size of word vectors), is derived as follows:

$$\begin{aligned} T(h) &= T_{ap}(h) + T_{f_3}(h) \\ &= \mathrm{O}\big(m(\hat{n}^2 + d^2)\big) + \mathrm{O}\big(m\hat{n}d^2\big) \\ &= \mathrm{O}\big(m(\hat{n}^2 + d^2 + \hat{n}d^2)\big), \end{aligned} \tag{9}$$

where T_{ap} and T_{f_3} are respectively the time complexity of the AP procedure and of training the f_3 model. Eq. 9 shows that when $\hat{n} \ll m$, the model complexity is linear with the input bags m and always quadratic with the number of features d.

Previous works on relevance assignment for MIR have prohibitive complexity for high-dimensional feature spaces or numerous bags and hence they are not most appropriate for text regression tasks. Wagstaff and Lane (2007) have cubic time complexity with the average bag size $\hat{n}$ and number of features d; Zhou et al. (2009) use kernels, thus their complexity is quadratic with the number of bags m; and Wang et al. (2011) have cubic time wrt. d. Our formulation is thus competitive in terms of complexity.

6 Data, Protocol and Metrics

6.1 Aspect Rating Datasets

We use seven datasets summarized in Table 1. Five publicly available datasets were built for aspect prediction by McAuley et al. (2012) – BeerAdvocate, Ratebeer (ES), RateBeer (FR), Audiobooks and Toys & Games – and have aspect ratings assigned by their creators on the respective websites. On the set of comments on TED talks from Pappas and Popescu-Belis (2013), we aim to predict two things: talk-level emotion dimensions assigned by viewers through voting, and comment polarity scores assigned by crowdsourcing. The distributions of aspect ratings per dataset are shown in Figure 3. Five datasets are in English, one in Spanish (Ratebeer) and one in French (RateBeer), so our results will also demonstrate the language-independence of our method.

From every dataset we kept 1,200 texts as bags of sentences, but we also used three *full-size* datasets, namely Ratebeer ES (1,259 labeled reviews), Ratebeer FR (17,998) and Audiobooks (10,989). The features for each of them are word vectors with binary attributes signaling word presence or absence, in a traditional bag-of-words model (BOW). The word vectors are provided with the first five datasets and we generated them for the latter two, after lowercasing and stopword removal. Moreover, for TED comments, we computed TF-IDF scores using the same dimensionality as with BOW to experiment with a different feature space. The target class labels were normalized by the maximum rating in their scale, except for TED talks where the votes were normalized by the maximum number of votes over all the emotion classes for each talk, and two emotions, 'informative' and 'ok', were excluded as they are neutral ones.

6.2 Evaluation Protocol

We compare the proposed model, noted *AP-Weights*, with four baseline ones – *Aggregated*, *Instance*, *Prime* (Section 4) and *Clus-*

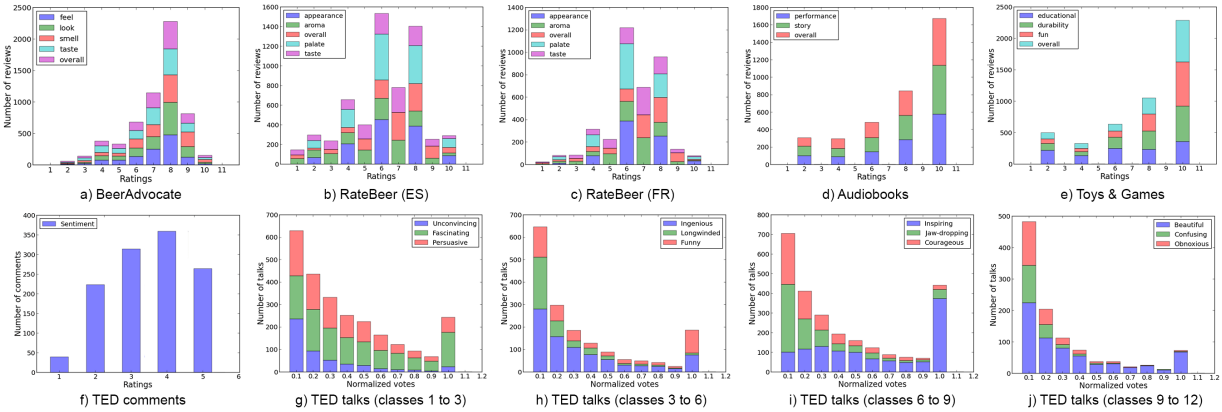


Figure 3: Distributions of rating values per aspect rating class for the seven datasets.

tering (from `github.com/garydoranjr/mcr`), which is an instance relevance method proposed by Wagstaff et al. (2008) for aspect rating prediction. First, for each aspect class, we optimize all methods on a development set of 25% of the data (300 randomly selected bags). Then, we perform 5-fold cross-validation for every aspect on each entire data set and report the average error scores using the optimal hyper-parameters per method. In addition, we report for comparison the scores of *AverageRating*, which always predicts the average rating over the training set.

We report standard error metrics for regression, namely the Mean Absolute Error (MAE) and the Mean Squared Error (MSE). The former measures the average magnitude of errors in a set of predictions while the latter measures the average of their squares, which are defined over the test set of bags B_i respectively as $\text{MAE} = (\sum_{i=1}^{k} |f(B_i) - y_i|)/k$ and $\text{MSE} = (\sum_{i=1}^{k} (f(B_i) - y_i)^2)/k$. The cross-validation scores are obtained by averaging the MAE and MSE scores on each fold.

To find the optimal hyper-parameters for each model, we perform 3-fold cross-validation on the development set using exhaustive grid-search over a fine-grained range of possible values and select the ones that perform best in terms of MAE. The hyper-parameters to be optimized for the baselines (except AverageRating) are the regularization terms λ_2, λ_1 of their possible regression model f, namely SVR which uses the ℓ_2 norm and Lasso which uses the ℓ_1 norm. As for APWeights, it relies on three regularization terms, namely $\epsilon_1, \epsilon_2, \epsilon_3$ of the ℓ_2-norm for f_1, f_2 and f_3 regression models. Lastly, for the Clustering baseline, we use the f_2 regression model, which relies on ϵ_2 and the number of clusters k, optimized over $\{5, ..., 50\}$ with step 5, for its clustering algorithm, here k-Means. All the regularization terms are optimized over the same range of possible values, noted $a \cdot 10^b$ with $a \in \{1, \ldots, 9\}$ and $b \in \{-4, \ldots, +4\}$, hence 81 values per term. For the regression models and evaluation protocol, we use the *scikit-learn* machine learning library (Pedregosa et al., 2012). Our code and data are available in the first author's website.

7 Experimental Results

7.1 Aspect Rating Prediction

The results for aspect rating prediction are given in Table 2. The proposed APWeights method outperforms Aggregated (ℓ_2) and Aggregated (ℓ_1) i.e. SVR and Lasso along with all other baselines on each case. The SVR baseline has on average 11% lower performance than APWeights in terms of MSE and about 6% in terms of MAE. Similarly, the Lasso baseline has on average 13% lower MSE and 8% MAE than APWeights. As shown in Figure 4, APWeights also outperforms them for each aspect in the five review datasets. The Instance method with ℓ_1 performed well on BeerAdvocate and Toys & Games (for MSE), and with ℓ_2 performed well on Ratebeer (ES), RateBeer (FR) and Toys & Games (for MAE). Therefore, the instance-as-example assumption is quite appropriate for this task, however both options score below APWeights – by about 5% MAE, and 8%/9% MSE, respectively. The Prime method with ℓ_1 performed well only on the BeerAdvocate dataset and Prime with ℓ_2 only on the Toys & Games dataset, always with lower scores than APWeights, namely about 9% MAE for both and 15%/18% MSE respectively. This suggests that the primary-instance

	REVIEW LABELS									
	BeerAdvocate		**RateBeer (ES)**		**RateBeer (FR)**		**Audiobooks**		**Toys & Games**	
Model \ Error	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**
AverageRating	14.20	3.32	16.59	4.31	12.67	2.69	21.07	6.75	20.96	6.75
Aggregated (ℓ_1)	13.62	3.13	15.94	4.02	12.21	2.58	20.10	6.14	20.15	6.33
Aggregated (ℓ_2)	14.58	3.68	14.47	3.41	12.32	2.70	19.08	5.99	18.99	5.93
Instance (ℓ_1)	12.67	2.89	14.91	3.54	11.89	2.48	20.13	6.17	20.33	6.34
Instance (ℓ_2)	13.74	3.28	14.40	3.39	11.82	2.40	19.26	6.04	19.70	6.59
Prime (ℓ_1)	12.90	2.97	15.78	3.97	12.70	2.76	20.65	6.46	21.09	6.79
Prime (ℓ_2)	14.60	3.64	15.05	3.68	12.92	2.98	20.12	6.59	20.11	6.92
Clustering (ℓ_2)	13.95	3.26	15.06	3.64	12.23	2.60	20.50	6.48	20.59	6.52
APWeights (ℓ_2)	**12.24**	**2.66**	**14.18**	**3.28**	**11.37**	**2.27**	**18.89**	**5.71**	**18.50**	**5.57**
APW vs. SVR (%)	*+16.0*	*+27.7*	*+2.0*	*+3.8*	*+7.6*	*+15.6*	+1.0	*+4.5*	*+2.6*	*+6.0*
APW vs. Lasso (%)	*+10.1*	*+15.1*	*+11.0*	*+18.4*	*+6.8*	*+11.8*	*+6.0*	*+6.9*	*+8.1*	*+11.9*
APW vs. 2^{nd} best (%)	+3.3	+7.8	+1.5	*+3.3*	*+3.7*	*+4.9*	+1.0	*+4.5*	*+2.6*	*+6.0*

Table 2: Performance of aspect rating prediction (the lower the better) in terms of MAE and MSE ($\times$ 100) with 5-fold cross-validation. All scores are averaged over all aspects in each dataset. The scores of the best method are in **bold** and the second best ones are underlined. Significant improvements (paired t-test, $p < 0.05$) are in *italics*. Fig. 4 shows MSE scores per aspect for three methods on five datasets.

assumption is not the most appropriate for this task. Lastly, even though Clustering is an instance relevance method, it has similar scores to Prime, presumably because the relevances are assigned according to the computed clusters and they are not directly influenced by the task's objective.

To compare with the state-of-the-art results obtained by McAuley et al. (2012), we experimented with three of their *full-size* datasets. Splitting each dataset in half for training vs. testing, and using the optimal settings from our experiments above, we measured the average MSE over all aspects. APWeights improved over Lasso by 10%, 26% and 17% MSE respectively on each dataset – the absolute MSE scores are .038 for Lasso vs. .034 for APWeights on Ratebeer SP; .023 vs. .017 on Ratebeer FR; .063 vs. .052 on Audiobooks. Similarly, when compared to the best SVM baseline provided by the McAuley et al., our method improved by 32%, 43% and 35% respectively on each dataset, though it did not use their rating model. Moreover, the best model proposed by McAuley et al., which uses a joint rating model and an aspect-specific text segmenter trained on hand-labeled data, reaches MSE scores of .03, .02 and .03, which is comparable to our model that does not use these features (.034, .017, .052), though it could benefit from them in the future. Lastly, as mentioned by the same authors, predictors which use segmented text, for example with topic models as in (Lu et al., 2011), do not necessarly outperform SVR baselines; instead they have marginal or even no improvements, therefore, we did not further experiment with them. Interestingly, multiple-instance learning algorithms under several assumptions go beyond SVR baselines with BOW and even more sophisticated features such as TF-IDF (see below).

	SENT. LABELS		EMO. LABELS	
	TED comm.		**TED talks**	
Model \ Error	**MAE**	**MSE**	**MAE**	**MSE**
AverageRating	19.47	5.05	17.86	6.06
Aggregated (ℓ_1)	17.08	4.17	15.98	5.03
Aggregated (ℓ_2)	16.88	4.47	15.24	4.97
Instance (ℓ_1)	17.69	4.37	16.48	5.30
Instance (ℓ_2)	16.93	4.24	16.10	5.57
Prime (ℓ_1)	17.39	4.37	15.98	5.78
Prime (ℓ_2)	18.03	4.91	16.74	5.94
Clustering (ℓ_2)	17.64	4.34	17.71	6.02
APWeights (ℓ_2)	**15.91**	**3.95**	**15.02**	**4.89**
APW vs SVR (%)	*+5.7*	*+11.5*	+1.5	+1.6
APW vs Lasso (%)	*+6.8*	+5.3	*+6.0*	*+2.9*
APW vs 2^{nd} (%)	*+5.7*	+5.3	+1.5	+1.6

Table 3: MAE and MSE ($\times$ 100) on sentiment and emotion prediction with 5-fold c.-v. Scores on TED talks are averaged over the 12 emotions. The scores of the best method are in **bold** and the second best ones are underlined. Significant improvements (paired t-test, $p < 0.05$) are in *italics*.

7.2 Sentiment and Emotion Prediction

Our method is also competitive for sentiment prediction over comments on TED talks, as well as for talk-level emotion prediction with 12 dimensions from subsets of 10 comments on each talk (see Table 3). APWeights outperforms SVR and Lasso, as well as all other methods for each task. For sentiment prediction, SVR is outperformed by 11% MSE and Lasso by 5%. For emotion pre-

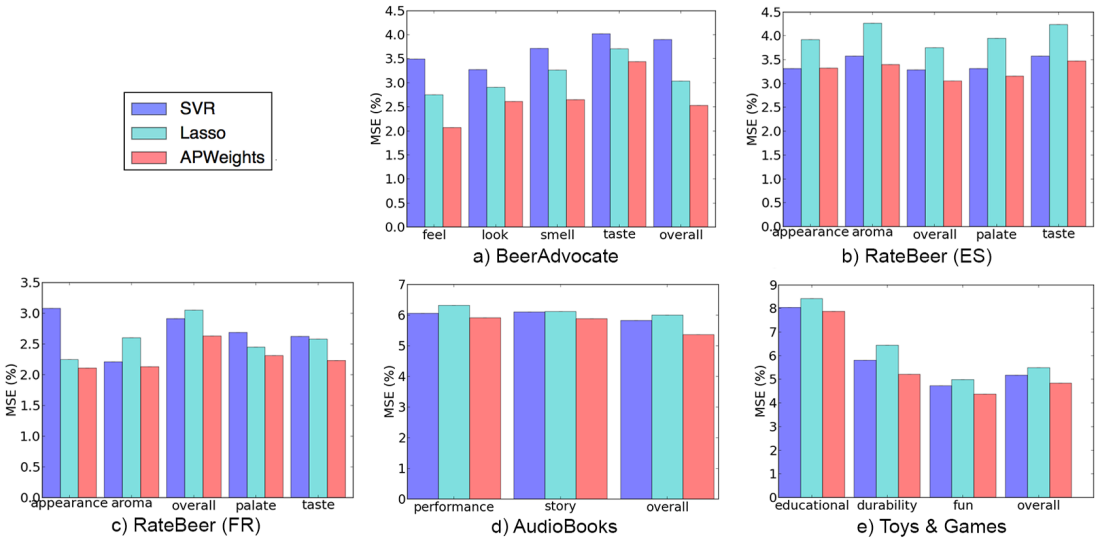


Figure 4: MSE scores of SVR, Lasso and APWeights for each aspect over the five review datasets.

diction (averaged over all 12 aspects), differences are smaller, at 1.6% and 2.9% respectively. These smaller differences could be explained by the fact that among the 10 most recent comments for each talk, many are not related to the emotion that the system tries to predict.

As mentioned earlier, the proposed model does not make any assumption about the feature space. Thus, we examined whether the improvements it brings remain present even with a different feature space, for instance based on TF-IDF instead of BOW with counts. For sentiment prediction on TED comments, we found that by changing the feature space to TF-IDF, strong baselines such as Aggregated (ℓ_1) and (ℓ_2), i.e. SVR and Lasso, improve their performance (16.25 and 16.59 MAE; 4.16 and 3.97 MSE respectively). However, APWeights still outperforms them on both MAE and MSE scores (15.35 and 3.63), improving over SVR by 5.5% on MAE and 12.5% on MSE, and over Lasso by 7.4% on MAE and 8.5% on MSE. These promising results suggest that improvements with APWeights could be observed also on more sophisticated feature spaces.

7.3 Interpreting the Relevance Weights

Apart from predicting ratings, the MIR scores assigned by our model reflect the contribution of each sentence to these predictions.

To illustrate the explanatory power of our model (until a dataset for quantitative analysis becomes available), we provide examples of predictions on test data taken from the cross-validation folds above. Table 5 displays the most relevant comment for two correctly predicted emotions on two TED talks, based on the $\hat{\psi}_i$ relevance scores, along with the $\hat{\psi}_i$ scores of the other comments, for two emotion classes: 'beautiful' and 'courageous'. These comments appear to reflect correctly the fact that the respective emotion is the majority one in each of the comments. As noted earlier, this task is quite challenging since we use only the ten most recent comments for each talk.

Table 4 displays four TED comments selected

Sentences per comment	$\hat{\psi}_i$	$\hat{y}_i$	y_i
"Very brilliant and witty, as well as great improvisation."	0.64	5.0	5.0
"I enjoyed this one a lot."	0.36		
"That's great idea, I really like it!"	0.56	4.2	4.0
"I can't wait to try it, but first thing, I need a house with big windows, next year, maybe I can do that."	0.44		
"Unfortunately countries are not led by gifted children."	0.48	2.4	2.0
"They are either dictated by the most extreme personalities who crave nothing but power or managed by politicians who are voted in by a far from gifted population."	0.52		
"I am very disappointed by this, smug, cliched and missing so much information as to be almost (...)"'	0.43	1.8	1.0
"No mention of ship transport lets say 50% of all material transport, no mention of rail transport, (...)"	0.29		
"I am sorry to be so negative, this just sounds like a sales pitch that he has given too many times (...)."	0.28		

Table 4: Predicted sentiment for TED comments: y_i is the actual sentiment, $\hat{y}_i$ the predicted one, and $\hat{\psi}_i$ the estimated relevance of each sentence.

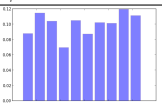
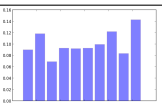
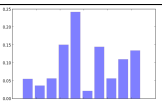
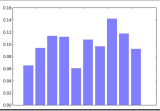
Class	Top comment per talk (according to weights ψ_i)	$\hat{\psi}_i$ distribution
inspiring	"It seems to me that the idea worth spreading of this TED Talk is inspiring and key for a full life. 'No-one else is the authority on your potential. You're the only person that decides how far you go and what you're capable of.' It seems to me that teens actually think that. As a child one is all knowing and all capable. How did we get to the (...)"	
beautiful	"The beauty of the nature. It would be more interesting just integrates his thought and idea into a mobile device, like a mobile, so we can just turn on the nature gallery in any time. The paintings don't look incidental but genuinely thought out, random perhaps, but with a clear grand design behind the randomness. Drawing is an art where it doesn't (...)"	
funny	"Funny story, but not as funny as a good 'knock, knock' joke. My favorite knock-knock joke of all time is Cheech & Chong's 'Dave's Not Here' gag from the early 1970s. I'm still waiting for someone to top it after all these years. [Knock, knock] 'Who is it?' the voice of an obviously stoned male answers from the other side of a door, (...)"	
courageous	"I was a soldier in Iraq and part of the unit represented in this documentary. I would question anyone that told you we went over there to kill Iraqi people. I spent the better part of my time in Iraq protecting the Iraqi people from insurgents who came from countries outside of Iraq to kill Iraqi people. We protected families men, women, and (...)"	

Table 5: Two examples of top comments (according to weights ψ_i) for correctly predicted emotions in four TED talks (score 1.0) and the distribution of weights over the 10 most recent comments in each talk.

Figure 5: Top words based on Φ for predicting four emotions from comments on TED talks.

from the test set of a given fold, for the comment-level sentiment prediction task. The table also shows the $\hat{\psi}_i$ relevance scores assigned to each of the composing sentences, the predicted polarity scores $\hat{y}_i$ and the actual ones y_i. We observe that the sentences that convey the most sentiment are assigned higher scores than sentences with less sentiment, always with respect to the global polarity level. These examples suggest that, given that APWeights has more degrees of freedom for interpretation, it is able to assign relevance to parts of a text (here, sentences) and even to words, while other models can only consider words. Hence, the assigned weights might be useful for other NLP tasks mentioned below.

8 Conclusion and Future Work

This paper introduced a novel MIR model for aspect rating prediction from text, which learns instance relevance together with target labels. To the best of our knowledge, this has not been considered before. Compared to previous work on MIR, the proposed model is competitive and more efficient in terms of complexity. Moreover, it is not only able to assign instance relevances on labeled bags, but also to predict them on unseen bags.

Compared to previous work on aspect rating prediction, our model performs significantly better than BOW regression baselines (SVR, Lasso) without using additional knowledge or features. The improvements persist even when the sophistication of the features increases, suggesting that our contribution may be orthogonal to feature engineering or learning. Lastly, the qualitative evaluation on test examples demonstrates that the parameters learned by the model are not only useful for prediction, but they are also interpretable.

In the future, we intend to test our model on sentiment classification at the sentence-level, based only on document-level supervision (Täckström and McDonald, 2011). Moreover, we will experiment with other model settings, such as regularization norms other than ℓ_2 and feature spaces other than BOW or TF-IDF. In the longer term, we plan to investigate new methods to estimate instance weights at prediction time, and to evaluate the impact of assigned weights on sentence ranking, segmentation or summarization.

Acknowledgments

The work described in this article was supported by the European Union through the inEvent project FP7-ICT n. 287872 (see `http://www.inevent-project.eu`).

References

Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. 2003. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems*, pages 561–568, Vancouver, British Columbia, Canada.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2009. Multi-facet rating of product reviews. In Mohand Boughanem, Catherine Berrut, Josiane Mothe, and Chantal Soule-Dupuy, editors, *Advances in Information Retrieval*, volume 5478 of *Lecture Notes in Computer Science*, pages 461–472. Springer Berlin Heidelberg.

Razvan C. Bunescu and Raymond J. Mooney. 2007. Multiple instance learning for sparse positive bags. In *Proceedings of the 24th Annual International Conference on Machine Learning*, ICML '07, Corvallis, OR, USA.

Jesse Davis et al. 2007. Tightly integrating relational learning and multiple-instance regression for real-valued drug activity prediction. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 425–432, Corvallis, OR, USA.

Harris Drucker, Chris J.C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. 1996. Support vector regression machines. In *Advances in Neural Information Processing systems*, pages 155–161, Denver, CO, USA.

James Foulds and Eibe Frank. 2010. A review of multi-instance learning assumptions. *The Knowledge Engineering Review*, 25:1:1–25.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, KDD '04, pages 168–177, Seattle, WA.

Fangtao Li, Nathan Liu, Hongwei Jin, Kai Zhao, Qiang Yang, and Xiaoyan Zhu. 2011. Incorporating reviewer and product information for review rating prediction. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence - Volume 3*, IJCAI '11, pages 1820–1825, Barcelona, Catalonia, Spain.

Bin Lu, Myle Ott, Claire Cardie, and Benjamin K. Tsou. 2011. Multi-aspect sentiment analysis with topic models. In *Proceedings of the 11th IEEE International Conference on Data Mining Workshops*, ICDMW '11, pages 81–88, Washington, DC.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, Portland, OR.

J. McAuley, J. Leskovec, and D. Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *Proceedings of the 12th IEEE International Conference on Data Mining*, ICDM '12, pages 1020–1025, Brussels, Belgium.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th Int. Conf. on the World Wide Web*, WWW '07, pages 171–180, Banff, AB.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 115–124, Ann Arbor, MI.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL Conf. on Empirical Methods in Natural Language Processing*, EMNLP '02, pages 79–86, Philadelphia, PA.

Nikolaos Pappas and Andrei Popescu-Belis. 2013. Sentiment analysis of user comments for one-class collaborative filtering over TED talks. In *36th ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, Dublin, Ireland.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2012. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490.

Lizhen Qu, Georgiana Ifrim, and Gerhard Weikum. 2010. The bag-of-opinions method for review rating prediction from sparse text patterns. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 913–921, Beijing, China.

Soumya Ray and David Page. 2001. Multiple instance regression. In *Proceedings of the 18th International Conference on Machine Learning*, ICML '01, pages 425–432.

Christina Sauper and Regina Barzilay. 2013. Automatic aggregation by joint modeling of aspects and values. *Journal of Artificial Intelligence Research*, 46(1):89–127.

Christina Sauper, Aria Haghighi, and Regina Barzilay. 2010. Incorporating content structure into text analysis applications. In *Proceedings of the 2010 Conference on Empirical Methods in Natural*

Language Processing, EMNLP '10, pages 377–387, Cambridge, MA.

Burr Settles, Mark Craven, and Soumya Ray. 2008. Multiple-instance active learning. In *Advances in Neural Information Processing Systems*, NIPS '08, pages 1289–1296, Vancouver, BC.

Benjamin Snyder and Regina Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '07, pages 300–307, Rochester, NY, USA.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 151–161, Edinburgh, UK.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '13, pages 1631–1642, Portland, OR.

Oscar Täckström and Ryan McDonald. 2011. Discovering fine-grained sentiment with latent variable structured prediction models. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval*, ECIR'11, pages 368–374, Berlin, Heidelberg. Springer-Verlag.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 111–120, Beijing, China.

Peter D. Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 417–424, Philadelphia, PA.

Kiri L. Wagstaff and Terran Lane. 2007. Salience assignment for multiple-instance regression. In *ICML 2007 Workshop on Constrained Optimization and Structured Output Spaces*, Corvallis, Oregon, USA.

Kiri L. Wagstaff, Terran Lane, and Alex Roper. 2008. Multiple-instance regression with structured data. In *Proceedings of the IEEE International Conference on Data Mining Workshops*, ICDMW '08, pages 291–300.

Zhuang Wang, Vladan Radosavljevic, Bo Han, Zoran Obradovic, and Slobodan Vucetic. 2008. Aerosol optical depth prediction from satellite observations by multiple instance regression. In *Proceedings of the SIAM Int. Conf. on Data Mining*, SDM '08, pages 165–176, Atlanta, GA.

Hua Wang, Feiping Nie, and Heng Huang. 2011. Learning instance specific distance for multi-instance classification. In *AAAI Conference on Artificial Intelligence*.

Zhuang Wang, Liang Lan, and S. Vucetic. 2012. Mixture model for multiple instance regression and applications in remote sensing. *IEEE Transactions on Geoscience and Remote Sensing*, 50(6):2226–2237.

Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Computational Linguistics*, 30(3):277–308, September.

Min-Ling Zhang and Zhi-Hua Zhou. 2008. M3MIML: A maximum margin method for multi-instance multi-label learning. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, pages 688–697, Dec.

Min-Ling Zhang and Zhi-Hua Zhou. 2009. Multi-instance clustering with applications to multi-instance prediction. *Applied Intelligence*, 31(1):47–68.

Zhi-Hua Zhou, Kai Jiang, and Ming Li. 2005. Multi-instance learning based web mining. *Applied Intelligence*, 22(2):135–147.

Zhi-Hua Zhou, Yu-Yin Sun, and Yu-Feng Li. 2009. Multi-instance learning by treating instances as non-i.i.d. samples. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1249–1256, Montreal, Quebec, Canada.

Jingbo Zhu, Chunliang Zhang, and Matthew Y. Ma. 2012. Multi-aspect rating inference with aspect-based segmentation. *IEEE Trans. on Affective Computing*, 3(4):469–481.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 43–50, Arlington, VA.

Sentiment Analysis on the People's Daily

Jiwei Li[1] **and Eduard Hovy**[2]
[1]Computer Science Department, Stanford University, Stanford, CA 94305, USA
[2]Language Technology Institute, Carnegie Mellon University, PA 15213, USA
jiweil@stanford.edu ehovy@andrew.cmu.edu

Abstract

We propose a semi-supervised bootstrapping algorithm for analyzing China's foreign relations from the People's Daily. Our approach addresses sentiment target clustering, subjective lexicons extraction and sentiment prediction in a unified framework. Different from existing algorithms in the literature, time information is considered in our algorithm through a hierarchical bayesian model to guide the bootstrapping approach. We are hopeful that our approach can facilitate quantitative political analysis conducted by social scientists and politicians.

1 Introduction

"We have no permanent allies, no permanent friends, but only permanent interests."

-Lord Palmerston

Newspapers, especially those owned by official governments, e.g., *Pravda* from Soviet Union, or *People's Daily* from P.R. China, usually provide direct information about policies and viewpoints of government. As national policies change over time, the tone that newspapers adopt, especially sentiment, changes along with the policies. For example, there is a stark contrast between the American newspapers' attitudes towards Afghanistan before and after 911. Similarly, consider the following examples extracted from the People's Daily[1]:

- People' Daily, Aug 29th, 1963
  All those who are being oppressed and exploited, Unite !! Beat US Imperialism and its lackeys.
- People's Daily, Oct, 20th, 2002
  A healthy, steady and developmental relationship between China and US, conforms to the fundamental interests of people in both countries, and the trend of historical development.

[1]Due to the space constraints, we only show the translated version in most of this paper.

Automatic opinion extraction from newspapers such as people's daily can facilitate sociologists 'or political scientists' research or help political pundits in their decision making process. While our approach applies to any newspaper in principle, we focus here on the People's Daily[2] (Renmin Ribao), a daily official newspaper in the People's Republic of China.

While massive number of works have been introduced in sentiment analysis or opinion target extraction literature (for details, see Section 2), a few challenges limit previous efforts in this specific task: First, the heavy use of linguistic phenomenon in the People's Daily including rhetoric, metaphor, proverb, or even nicknames, makes existing approaches less effective for sentiment inference as identifying these expressions is a hard NLP problem in nature.

Second, as we are more interested in the degree of sentiment rather than binary classification (i.e., positive versus negative) towards an entity (e.g. country or individual) in the news article, straightforward algorithms to apply would be document-level sentiment analysis approaches such as vector machine/regression (Pang et al., 2002) or supervised LDA (Blei and McAuliffe, 2010). A single news article, usually contains different attitudes towards multiple countries or individuals simultaneously (say praising "friends" and criticizing "enemies"), as shown in the following example from the People's Daily of Mar. 17th, 1966:

US imperialism set up a puppet regime in Vietnam and sent expeditionary force... People of Vietnam prevailed over the modern-equipped US troops with a vengeance... The result of Johnson Government's intensifying invasion is that.... There will be the day, when people from all over the world execute the heinous US imperialism by hanging on a gibbet.... The heroic people of Vietnam, obtained great victory in the struggle against the USA imperialism...

The switching of praising of Vietnam and criticizing of the USA would make aforemen-

[2]paper.people.com.cn/rmrb/

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 467–476,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

tioned document-level machine learning algorithms based on bags of words significantly less effective if not separating attitudes towards Vietnam from toward the USA in the first place. Meanwhile, the separating task is by no means trivial in news articles. While *US imperialism, US troops, Johnson Government, invaders, Ngo Dinh Diem*[3] all point to the USA or its allies, *People of Vietnam, the Workers' party*[4]*, Ho Chi Minh*[5]*, Vietnam People's Army* point to North Vietnam side. Clustering entities according to sentiment, especially in Chinese, is fundamentally a difficult task. And our goal, trying to identify entities towards whom an article holds the same attitudes, is different from standard coreference resolution, since for us the co-referent group may include several distinct entities.

To address the aforementioned problems, in this paper, we propose a sentiment analysis approach based on the following assumptions:

1. *In a single news article, sentiment towards an entity is consistent.*
2. *Over a certain period of time, sentiments towards an entity are inter-related.*

The assumptions will facilitate opinion analysis: (1) if we can identify the attitude towards an entity (e.g., Vietnam) in a news article as positive, then negative attitudes expressed in the article are about other entities. (2) The assumption enables sentiment inference for unseen words in a bootstrapping way without having to employ sophisticated NLP algorithms. For example, from 1950s to 1960s, USA is usually referred to as "a tiger made of paper" in translated version. It is a metaphor indicating things that appear powerful (tiger) but weak in nature (made of paper). If it is first identified that during the designated time period, China held a pretty negative attitude towards the USA based on clues such as common negative expressions (e.g., "evil" or "reactionary"), we can easily induce that "a tiger made of paper", is a negative word.

Based on aforementioned two assumptions, we formulate our approach as a semi-supervised model, which simultaneously bootstrap sentiment target lists, extracts subjective vocabularies and performs sentiment analysis. Time information is considered through a hierarchical bayesian model to guide time-, document-, sentence- and term-level sentiment inference. A small seed set of subjective words constitutes our only source of supervision.

The main contributions of this paper can be summarized as follows:

1. We propose a semi-supervised bootstrapping algorithm tailored for sentiment analysis in the People's daily where time information is incorporated. We are hopeful that sentiment cues can shed insights on other NLP tasks such as coreference or metaphor recognition.

2. In Analytical Political Science, the quantitative evaluation of diplomatic relations is usually a manual task (Robinson and Shambaugh, 1995). We are hopeful that our algorithm can enable automated political analysis and facilitate political scientists' and historians' work.

2 Related Works

Significant research efforts have been invested into sentiment analysis and opinion extraction. In one direction, researchers look into predicting overall sentiment polarity at document-level (Pang and Lee, 2008), aspect-level (Wang et al., 2010; Jo and Oh, 2011), sentence-level (Yang and Cardie, 2014) or tweet-level (Agarwal et al., 2011; Go et al., 2009), which can be treated as a classification/regression problem by employing standard machine-learning techniques, such as Naive Bayesian, SVM (Pang et al., 2002) or supervised-LDA (Blei and McAuliffe, 2010) with different types of features (i.e., unigram, bigram, POS).

Other efforts are focused on targeted sentiment extraction (Choi et al., 2006; Kim and Hovy, 2006; Jin et al., 2009; Kim and Hovy, 2006). Usually, sequence labeling models such as CRF (Lafferty et al., 2001) or HMM (LIU et al., 2004) are employed for identifying opinion holders (Choi et al., 2005), topics of opinions (Stoyanov and Cardie, 2008) or opinion expressions (e.g. (Breck et al., 2007; Johansson and Moschitti, 2010; Yang and Cardie, 2012)). Kim and Hovy (2004; 2006) identified opinion holders and targets by exploring their semantics rules related to the opinion words. Choi et al. (2006) jointly extracted opinion expressions, holders and their *is-from* relations using an ILP approach. Yang and Cardie (2013) introduced a sequence tagging model based on CRF to jointly identify opinion holders, opinion targets, and expressions.

Methods that relate to our approach include semi-supervised approaches such as pipeline or

[3]Leader of South Vietnam

[4]Ruling political party of Vietnam.

[5]One of Founders of Democratic Republic of Vietnam (North Vietnam) and Vietnam Workers' party.

propagation algorithms (Qiu et al., 2011; Qiu et al., 2009; Zhang et al., 2010; Duyu et al., 2013). Concretely, Qiu et al. (2011) proposed a rule-based semi-supervised framework called *double propagation* for jointly extracting opinion words and targets. Compared to existing bootstrapping approaches, our framework is more general one with less restrictions[6]. In addition, our approach harness global information (e.g. document-level, time-level) to guide the bootstrapping algorithm. Another related work is the approach introduced by O'Connor et al. (O'Connor et al., 2013) that extracts international relations from political contexts.

3 the People's Daily

The People's Daily[7] (Renmin Ribao), established on 15 June 1946, is a daily official newspaper in the People's Republic of China, with a approximate circulation of 2.5 million worldwide. It is widely recognized as the mouthpiece of the Central Committee of the Communist Party of China (CPC) (Wu, 1994). Editorials and commentaries are usually regarded both by foreign observers and Chinese readers as authoritative statements of government policy[8]. According to incomplete statistics, there have benn at least 13 major redesigns (face-liftings) for the People's Daily in history, the most recent in 2013.

4 Model

In this section, we present our model in detail.

4.1 Target and Expression extraction

We first extract *expressions* (attitude or sentiment related terms or phrases) and *target* (entities toward whom the opinion holder (e.g., the People's Daily) holds an attitude). See the following examples:

1. *[Albania Workers' party][T] is the [glorious][E] [party][T] of [Marxism and Leninism][E].*
2. *The [heroic][E] [people of Vietnam][T] obtained [great][E] [victory][E] against [the U.S. imperialism][T,E].*
3. *We strongly [warn][E] Soviet Revisionism][E,T].*

While the majority of subjective sentences omit the opinion holder, as in Examples 1 and 2, there are still a few circumstances where opinion holders (e.g., "we", "Chinese people", "Chinese government") are retained (Example 3). Some words (i.e. U.S. imperialism) can be both target and expression, and there can be multiple targets (Example 2) within one sentence.

We use a semi-Markov Conditional Random Fields (semi-CRFs) (Sarawagi and Cohen, 2004; Okanohara et al., 2006) algorithm for target and expression extraction. Semi-CRF are CRFs that relax the Markovian assumptions and allow for sequence labeling at the segment level. It has been demonstrated more powerful that CRFs in multiple sequence labeling applications including NER (Okanohara et al., 2006), Chinese word segmentation (Andrew, 2006) and opinion expression identification (Yang and Cardie, 2012). Our approach is an extension of Yang and Cardie (2012)'s system[9]. Features we adopted included:

- word, part of speech tag, word length.
- left and right context words within a window of 2 and the correspondent POS tags.
- NER feature.
- subjectivity lexicon features from dictionary[10]. The lexicon consists of a set of Chinese words that can act as strong or weak cues to subjectivity.
- segment-level syntactic features defined in (Yang and Cardie, 2012).

Most existing NER systems can barely recognize entities such as [Vietnamese People's Army] as a unified name entity in that Chinese parser usually divides them into a series of separate words, namely [Vietnamese/People's Army]. To handle this problem, we first employ the Stanford NER engine[11] and then iteratively 'chunk' consecutive words, at least one of which is labeled as a name entity by the NER engine, before checking whether the chunked entity matches a bag of words contained in Chinese encyclopedia, e.g., Baidu Encyclopedia[12] and Chinese Wikipedia[13].

[6] Qiu et al.'s rule base approach makes strong assumptions that consider opinion word to adjectives and targets to be nouns/noun, thus only capable of capturing sentences with simple patterns.

[7] paper.people.com.cn/rmrb/

[8] http://en.wikipedia.org/wiki/People's_Daily

[9] Yang and Cardie's system focuses on expression extraction (not target) and identifies *direct subjective expression (DSE)* and *expressive subjective expression (ESE)*.

[10] http://ir.dlut.edu.cn/NewsShow.aspx?ID=215

[11] http://nlp.stanford.edu/downloads/CRF-NER.shtml

[12] http://baike.baidu.com/

[13] http://zh.wikipedia.org/wiki/Wikipedia

4.2 Notation

Here we describe the key variables in our model. Let C_i denote the name entity of country i, G_i denote its corresponding collection of news articles. G_i is divided into 60*4=240 time spans (one for each quarter of the year, 60 years in total), $G_i = \{G_{i,t}\}$. $G_{i,t}$ is composed of a series of documents $\{d\}$, and d is composed of a series of sentences $\{S\}$, which is represented as a tuple $S = \{E_S, T_S\}$, where E_S is the expression and T_S is the target of current sentence.

Sentiment Score m**:** As we are interested in the degree of positiveness or negativeness, we divided international relations into 7 categories: *Antagonism* (score 1), *Tension* (score 2), *Disharmony* (score 3), *Neutrality* (score 4), *Goodness* (score 5), *Friendliness* (score 6), *Brotherhood (Comradeship)* (score 7) based on researches in political science literature[14]. Each of $G_{i,t}$, document d, sentence S and expression term w is associated with a sentiment score $m_{i,t}$, m_d, m_S and m_w, respectively. M denotes the list of subjective terms, $M = \{w, m_w\}$

Document Target List T_i^d**:** We use T_i^d to denote the collection of entity targets in document $d \in G_i$ which the People's daily holds similar attribute towards. For example, suppose document d belongs to Vietnam article collection ($C_i = Vietnam$), T_i^d can be {Vietnam, Workers' party, People's Army, Ho Chi Minh}. While U.S., U.S. troops and Lyndon Johnson are also entity targets found in d, they are not supposed to be included in T_i^d since the author holds opposite attributes.

Sentence List d_i**:** We further use d_i denotes the subset of sentences in d talking about entities from target list T_i^d. Similarly, in a Vietnam related article, sentences talking about the U.S. are not supposed to be included in d_i.

4.3 Hierarchical Bayesian Markov Model

In our approach, time information is incorporated through a hierarchical Bayesian Markov framework where $m_{i,t}$ is modeled as a first-order Poisson Process given the coherence assumption in time-dependent political news streams.

$$m_{i,t} \sim Poisson(m_{i,t}, m_{i,t-1}) \quad (1)$$

[14] http://www.imir.tsinghua.edu.cn/publish/iis/7522/20120522140122561915769

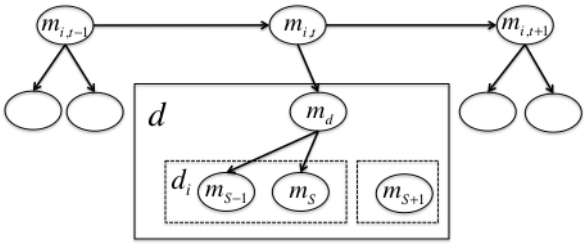


Figure 1: Hierarchical Bayesian Model for Inference

For each document $d \in G_{i,t}$, m_d is sampled from a Poisson distribution with mean value of $m_{i,t}$.

$$m_d \sim Poisson(m_d, m_{i,t}) \quad (2)$$

For sentence $S \in d_i$, m_S is sampled from m_d from a Poisson distribution based on m_d.

$$m_S \sim Poisson(m_S, m_d) \quad (3)$$

4.4 Intialization

Given a labeled subjective list M, for article $d \in G_i$, we initialize T_i^d with the name of entity C_i, d_i with sentences satisfying $T_S = C_i$ and $E_S \in M$. m_S for $S \in d_i$, is initialized as the average score of its containing expression E_s based on M. Then the MCMC algorithm is applied by iteratively updating m_d and $m_{i,t}$ according to the posterior distribution. Let $P(m|\cdot)$ denotes the probability of parameter m given all other parameters and the posterior distributions are given by:

$$\begin{aligned} &P(m_d = \lambda|\cdot) \propto Poisson(\lambda, m_{i,t}) \prod_{S \in d_i} Poisson(\lambda, m_S) \\ &P(m_{i,t} = \lambda|\cdot) \propto Poisson(\lambda, m_{i,t-1}) \\ &\times Poisson(m_{i,t+1}, \lambda) \cdot \times \prod_{d \in G_{i,t}} Poisson(m_d, \lambda) \end{aligned} \quad (4)$$

4.5 Semi-supervised Bootstrapping

Our semi-supervised learning algorithm updates M, T_i^d, d_i, S_d and S_i^d iteratively. A brief interpretation is shown in Figure 2 and the details are shown in Figure 4. Concretely, for each sentence $S \in d - d_i$, **step 1** means, if its expression E_S exists in subjective list M, we added its target T_S to T_i^d and S to d_i. **step 2** means if the target T_S exists in T_i^d, its expression, E_s, is added to subjective list M with score m_d. As M and T_i^d change in the iteration, in **step 3**, we again go over all unconsidered sentences with new M and T_i^d. m_d and $m_{i,t}$ are then updated based on new m_S using MCMC in Equ. 4. Note that sentences with pronoun target are not involved in the bootstrapping procedure.

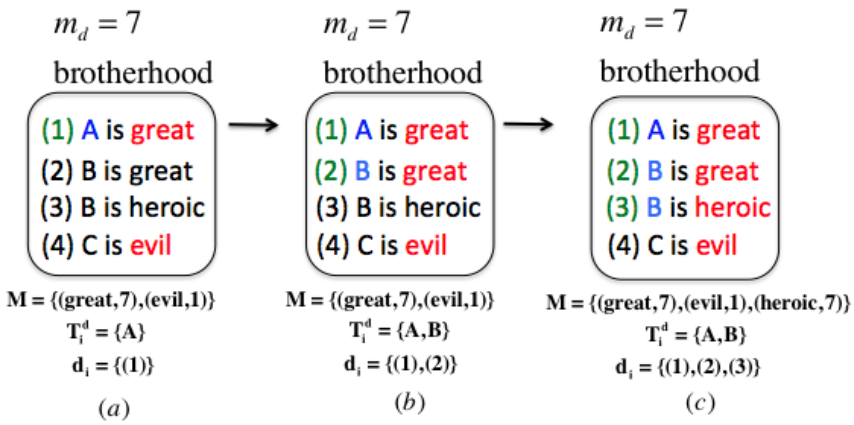


Figure 2: A brief demonstration of the adopted semi-supervised algorithm. (a)→(b): Sentence (2) is added to d_i due to the presence of already known subjective term "great" . Target B is added to target list T_i^d. (b)→(c): term "heroic" is added to subjective word list M with score 7 since it modifies target B.

Input: Entity C_i, G_i, subjective term list M
- for each entity i, each document d
  $T_i^d = \{C_i\}$, $d_i = \{S | S \in d, C_i = T_S, E_s \in M\}$
  for each sentence $S \in d_i$:
  . $m_s = \frac{1}{|E_S \in M|} \sum m_{E_s}$
- Iteratively update $m_{i,t}$, m_d using MCMC based on posterior probability shown in Equ.4.

Output: $\{d_i\}$, $\{T_i^d\}$, $\{m_{i,t}\}$ and $\{m_d\}$

Figure 3: Initialization Algorithm.

Input: Entity $\{C_i\}$, Articles Collections $\{G_i\}$, subjective term list M, sentiment score $\{m_d\}$, $\{m_{i,t}\}$, target list for each document $\{T_i^d\}$
Algorithm:
while not convergence:
• for each entity C_i, document d:
for each sentence $S \in d - d_i$
1. if $E_S \in M, T_s \notin T_i^d$
$T_i^d = T_i^d \bigcup T_s, d_i = d_i \bigcup S, m_S = m_d$
2. if $T_s \in T_i^t, E_s \notin M$
$M = M \bigcup (E_s, S_d), d_i = d_i \bigcup S, m_s = m_d$
3. if $E_S \in M, T_S \in T_i^d$
$d_i = d_i \bigcup S, m_S = m_{E_s}$
•Iteratively update $m_{i,t}$, m_d using MCMC based on posterior probability shown in Equ.4 .
end while:
Output: subjective term list M, score $\{m_{i,t}\}$

Figure 4: Semi-supervised learning algorithm.

4.6 Error Prevention in Bootstrapping

Error propagation is highly influential and damaging in bootstrapping algorithms, especially when extending very limited data to huge corpora. To avoid the collapse of the algorithm, we select candidates for opinion analysis in a extremely strict manner, at the sacrifice of many subjective sentences[15]. Concretely, we only consider sentences with exactly one target and at least one expression. Sentences with multiple targets (e.g., Example 2 in Section 4.1) or no expressions, or no targets are discarded.

In addition to the strict sentence selection approach, we adopt the following methods for self-correction in the boot-strapping procedure:

1. For $T_1, T_2 \in T_i^d$, $(E_1, T_1) \in S_1$, $(E_2, T_2) \in S_2$, $E_1, E_2 \in M$, if $|m_{E_1} - m_{E_2}| > 1$: Expel E_1 and E_2 from M, expel T_1 and T_2 from T_i^d, with the exception of original labeled data.
   Explanation: If sentiment scores for two expressions, whose correspondent targets both belong to the target list T_i^d, diverge enough, we discard both expressions and targets based according to Assumption 1: sentiments towards one entity (or its allies) in an article should be consistent.
2. $\exists S \in d$, $T_S \in T_i^d, |m_{E_S} - m_d| > 1$, T_S is expelled from T_i^d.
   Explanation: If target T_S for sentence S belongs to T_i^d, but its corresponding expression E_s is not consistent with article-level sentiment m_d, T_S is expelled from T_i^d.

[15]Negative effect of strict sentence selection can be partly compensated by the consideration of time-level information

5 Experiment

5.1 Data and Preprocessing

Our data set is composed of the People's daily from 1950 to 2010, across a 60-year time span.

antagonism (m=1)	残暴(extremely cruel), 敌人(enemy)
tension (m=2)	愤慨(indignation), 侵犯(offend)
disharmony (m=3)	失望(disappointed), 遗憾(regret)
neutrality (m=4)	关切, 关注(concern)
goodness (m=5)	发展的(developmental), 尊重(respect)
friendship (m=6)	友谊(friendship), 朋友(friend)
brotherhood (m=7)	伟大(firmly), 兄弟(brother)

Table 1: Illustration of subjective list M

News articles are first segmented using ICTCLAS Chinese segmentation word system[16] (Zhang et al., 2003). Articles with fewer than 200 Chinese words are discarded. News articles are clustered by the presence of a country's name more than 2 times based on a country name list from Wikipedia[17]. Articles mentioning more than 5 different countries are discarded since they usually talk about international conferences. Note that one article can appear in different collections (example in Section 1 will appear in both Vietnam and the U.S. collection).

Compound sentences are segmented into clauses based on dependency parse tree. Then those containing more than 50 characters or less than 4 characters are discarded. To avoid complicated inference, sentences with negation indicators are discarded.

5.2 Obtaining Subjectivity Word List

Since there are few Chinese subjectivity lexicons (with degrees) available and those exist may not serve our specific purpose, we manually label a small number of Chinese subjective terms as seed corpus. We divided the labeling process into 2 steps rather than directly labeling vocabularies[18]. We first selected 100 news articles and assigned each of them (as well as the appropriate country entity C_i) to 2 students majoring in *International Studies*, asking them to give a label sentiment score (1 to 7) according to the rules described in Section 4.2. 20 students participated in the procedure. Since annotators have plenty of background knowledge, they agreed on 98 out of 100. Second, we selected out subjectivity lexicons by matching to a comprehensive subjectivity lexicons list[19]. and ask 2 students select the candidates that signal the document-level label from the first step. According to whether a word a selected or not, the value of $Cohen's\ \kappa$ is 0.78, showing substantial agreement. For the small amount of labels on which the judges disagree, we recruited an extra judge and to serve as a tie breaker. Table 1 shows some labeled examples.

[16]http://ictclas.org/

[17]http://zh.wikipedia.org/wiki/ 国家列表-(按洲排列)

[18]We tried direct vocabulary labeling in the first place, but got low score for inter agreement, where value of $Cohen's\ \kappa$ is only 0.43.

[19]http://ir.dlut.edu.cn/NewsShow.aspx?ID=215

	P	R	F
	Total		
semi-CRF	**0.74**	**0.78**	**0.76**
CRF	0.73	0.66	0.68
	Single		
semi-CRF	**0.87**	**0.92**	**0.90**
CRF	0.80	0.87	0.83

Table 2: Results for Expressions/Targets extraction.

5.3 Targets and Expressions Extraction

As the good performance of *semi-CRF* in opinion extraction has been demonstrated in previous work (Yang and Cardie, 2012), we briefly go over model evaluation in this subsection for brevity. We manually labeled 600 sentences and performed 5-fold cross validation for evaluation. We compare *semi-CRF* to *Standard CRF*. We report performances on two settings in Table 2. The first setting, *Total*, corresponds to performance on the whole dataset, while second one *Single*, denotes the performance on the set of sentences with only one target, which we are more interested in because multiple-target sentences are discarded in our algorithm. It turned out that *semi-CRF* significantly outputs standard CRF, approaching 0.90 F-1 score on *Single* setting.

5.4 Foreign Relation Evaluation

Gold-standard foreign relations are taken from Political Science research at the Institute of Modern International Relations, Tsinghua University, extracted from monthly quantitative China foreign relations reports with 7 countries (U.S., Japan, Russia/Soviet, England, France, India, and Germany) from 1950 to 2012[20].

We consider several baselines. For fair comparison, we use identical processing techniques for each approach. Some baselines make article-level predictions, for which we obtain time-period level relation prediction by averaging the documents.

Coreference+Bootstrap (CB): We first implemented Ngai and Wang's Chinese coreference sys-

[20]Details found here http://www.imir.tsinghua.edu.cn/publish/iisen/7523/index.html.

Model	Ours	CB	No-time
Pearson	0.895	0.753	0.808
Model	SVR-d	SLDA	SVR-S
Pearson	0.482	0.427	0.688

Table 3: Pearson Correlation with Gold Standard.

tem (2007). We then bootstrap sentiment terms and score based on entity coreference.

No-time: A simplified version of our approach where each article is considered as an independent unit and no time-level information is considered. m_d is obtained by averaging its containing sentences and used for later bootstrapping.

SVR-d: Uses SVM^{light}(Joachims, 1999) to train a linear SVR (Pang and Lee, 2008) for document-level sentiment prediction using the unigram feature. The 100 labeled documents are used as training data.

SLDA: supervised-LDA (Blei and McAuliffe, 2010) for document-level label prediction. Topic number is set to 10, 20, 50, 100 respectively and we report the best result.

SVR-S: Sentence-level SVR to sentences with presence of entity C_i[21]. We obtain document-level prediction by averaging its containing sentences and then time-period level prediction by averaging its containing documents.

We report the Pearson Correlation with gold standards in table 3. As we can observe, simple document-level regression models, i.e., **SVR** and **SLDA** do not fit this task. The reason is simple: one article d can appear in different collections. Recall the Vietnam example in Section 1, it appears in both $G_{Vietnam}$ and $G_{the\ U.S.}$. Sentiment prediction for d should be totally opposite in the two document collections: very positive in $G_{Vietnam}$ and very negative in G_{USA}. But document level prediction would treat them equally. Our approach outperforms **No-Time**, illustrating the meaningfulness of exploiting time-level information in our task. Our system approaches around 0.9 correlation with the gold standards. The reason why **No-Time** is better than **CB** is also simple: **CB** includes only coreferent entities in the target list (e.g., America for the USA article collection), and therefore overlooks rich information provided by non-coreferent entities (e.g., President Nixon or Nixon Government). **No-Time** instead groups entities according to attitude, thereby enabling more information to be harnessed. For **SVR-S**, as the regression model trained from limited labeled data can hardly cover unseen terms during testing, the performance is just OK. **SVR-S** also suffers from overlooking rich sources of information since it only considers sentences with exact mention of the name entity of the corresponding country.

[21]Features we explore include word entities in current sentence, POS, a window of $k \in \{1, 2\}$ words from the target and the expression and corresponding POS, and the dependency path between target and expression.

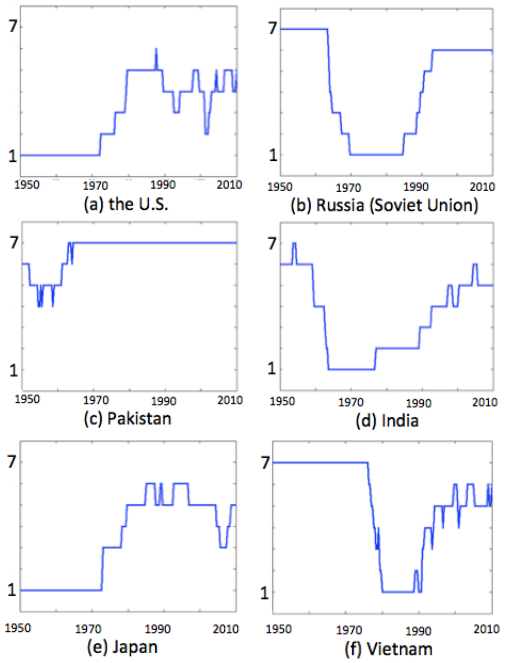


Figure 5: Examples of China's Foreign Relations.

6 Diplomatic Relations

"The enemy of my enemy is my friend"
—Arabic proverb

A central characteristic of post-World War Second international system with which China had to deal would be overwhelming preeminence of the USA and USSR as each of the superpowers stood at the center of a broad alliance system who was engaged in an intense and protracted global conflict with the other. We choose 6 countries and report results in Figure 5. One of interesting things we can observe from Figure 5 is that foreign attitudes are usually divergent towards two opposing forces: Sino-American relation (see Figure 5(a)) began to improve when the Sino-Soviet relation (see Figure 5(b)) reached its bottom at the beginning of 1970s. Similar patterns appear for Sino-Pakistan (see Figure 5(c)), Sino-India relations (see Figure 5(d))

Figure 6: Top coreference terms Towards USA and Soviet Union/Russia versus time. Blue denotes words that are both Target and positive words in M. Red denotes words that are both Target and negative words in M

in early 1960s[22], and Sino-Vietnam 5(f)), Sino-American relations in late 1970s. On the contrast, attitudes are usually consistent toward allied forces: Sino-Japan relations with Sino-USA relations before 1990s, and Sino-Vietnam relations with Sino-Soviet relations in late 1970s and 1980s.

Figure 6 presents top clustering target (T_i^d) in the USA and Soviet Union/Russia article collection. As some of vocabulary terms can be both target and expression, we use blue to label terms with positive sentiment, red to label negative ones. As we can see from Figure 6, targets(T) extracted by our model show a very clear pattern where allies and co-referent entities are grouped. Another interesting thing is, the subjectivity of target words from different times is generally in accord with the relation curves shown in Figure 5.

7 Conclusion and Discussion

In this paper, we propose a sentiment analysis algorithm to track China's foreign relations from the People's Daily. Our semi-supervised algorithm harnesses higher level information (i.e., document-level, time-level) by incorporating a hierarchical Bayesian approach into the framework, to resolve sentiment target clustering, create subjective lexicons, and perform sentiment prediction simultaneously. While we focus here on the People's Daily for diplomatic relation extraction, the idea of our approach is general and can be extended broadly. Another contribution of this work is the creation a comprehensive Chinese subjective lexicon list. We are hopeful that our approach can not only facilitate quantitative research by political scientists, but also shed light on NLP applications such as coreference and metaphor, where sentiment clues can be helpful.

It is worth noting that, while harnessing time-level information can indeed facilitate opinion analysis, especially when labeled data is limited in our specific task, it is not a permanent-perfect assumption, especially considering the diversity and treacherous currents at the international political stage.

At algorithm-level, to avoid error propagation due to limitations of current sentiment analysis tools (even though semi-CRF produces state-of-art performance in target and expression extraction task, a performance of 0.8 F-value, when applied to the whole corpus, can by no means satisfy our requirements), we discard a great number of sentences, among which is contained much useful information. How to resolve these problems and improve opinion extraction performance is our long-term goal in sentiment analysis/opinion extraction literature.

Acknowledgements

The authors want to thank Bishan Yang and Claire Cardie for useful comments and discussions. The authors are thankful for suggestions offered by EMNLP reviewers.

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment

[22] A fan of history can trace the crucial influence of the USSR in Sino-India relation in 1960s

analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*.

Galen Andrew. 2006. A hybrid markov/semi-markov conditional random field for sequence segmentation. In *EMNLP*.

David M Blei and Jon D McAuliffe. 2010. Supervised topic models. *arXiv preprint arXiv:1003.0783*.

Eric Breck, Yejin Choi, and Claire Cardie. 2007. Identifying expressions of opinion in context. In *IJCAI*.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In *EMNLP*.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *EMNLP*.

Tang Duyu, Qin Bing, Zhou LanJun, Wong KamFai, Zhao Yanyan, and Liu Ting. 2013. Domain-specific sentiment word extraction by seed expansion and pattern generation. *arXiv preprint arXiv:1309.6722*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*.

Wei Jin, Hung Hay Ho, and Rohini K Srihari. 2009. A novel lexicalized hmm-based learning framework for web opinion mining. In *ICML*.

Yohan Jo and Alice H Oh. 2011. Aspect and sentiment unification model for online review analysis. In *ICWSM*.

Thorsten Joachims. 1999. Making large scale svm learning practical.

Richard Johansson and Alessandro Moschitti. 2010. Syntactic and semantic structure for opinion expression detection. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1367. Association for Computational Linguistics.

Soo-Min Kim and Eduard Hovy. 2006. Extracting opinions, opinion holders, and topics expressed in online news media text. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Yun-zhong LIU, Ya-ping LIN, and Zhi-ping CHEN. 2004. Text information extraction based on hidden markov model [j]. *Acta Simulata Systematica Sinica*.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*.

Grace Ngai and Chi-Shing Wang. 2007. A knowledge-based approach for unsupervised chinese coreference resolution. *Computational Linguistics and Chinese Language Processing*, 12(4):459–484.

Daisuke Okanohara, Yusuke Miyao, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2006. Improving the scalability of semi-markov conditional random fields for named entity recognition. In *ACL*.

Brendan O'Connor, Brandon M Stewart, and Noah A Smith. 2013. Learning to extract international relations from political context. In *ACL*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *EMNLP*.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2009. Expanding domain sentiment lexicon through double propagation. In *IJCAI*.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational linguistics*.

Thomas W Robinson and David L Shambaugh. 1995. *Chinese foreign policy: theory and practice*. Oxford University Press.

Sunita Sarawagi and William W Cohen. 2004. Semi-markov conditional random fields for information extraction. In *NIPS*.

Veselin Stoyanov and Claire Cardie. 2008. Topic identification for fine-grained opinion analysis. In *Proceedings of the 22nd International Conference on Computational Linguistics*.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: a rating regression approach. In *SIGKDD*.

Guoguang Wu. 1994. Command communication: The politics of editorial formulation in the people's daily. *China Quarterly*, 137:194–211.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-markov conditional random fields. In *EMNLP*.

Bishan Yang and Claire Cardie. 2014. Context-aware learning for sentence-level sentiment analysis with posterior regularization. *ACL*.

Hua-Ping Zhang, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. Hhmm-based chinese lexical analyzer ictclas. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*.

Lei Zhang, Bing Liu, Suk Hwan Lim, and Eamonn O'Brien-Strain. 2010. Extracting and ranking product features in opinion documents. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*.

A Joint Segmentation and Classification Framework for Sentiment Analysis

Duyu Tang[♮*], **Furu Wei**[♯], **Bing Qin**[♮], **Li Dong**[♯*], **Ting Liu**[♮], **Ming Zhou**[♯]
[♮]Research Center for Social Computing and Information Retrieval,
Harbin Institute of Technology, China
[♯]Microsoft Research, Beijing, China
[♯]Beihang University, Beijing, China
[♮]{dytang, qinb, tliu}@ir.hit.edu.cn
[♯]{fuwei, mingzhou}@microsoft.com [♯]donglixp@gmail.com

Abstract

In this paper, we propose a joint segmentation and classification framework for sentiment analysis. Existing sentiment classification algorithms typically split a sentence as a word sequence, which does not effectively handle the inconsistent sentiment polarity between a phrase and the words it contains, such as "*not bad*" and "*a great deal of*". We address this issue by developing a joint segmentation and classification framework (**JSC**), which simultaneously conducts sentence segmentation and sentence-level sentiment classification. Specifically, we use a log-linear model to score each segmentation candidate, and exploit the phrasal information of top-ranked segmentations as features to build the sentiment classifier. A marginal log-likelihood objective function is devised for the segmentation model, which is optimized for enhancing the sentiment classification performance. The joint model is trained only based on the annotated sentiment polarity of sentences, without any segmentation annotations. Experiments on a benchmark Twitter sentiment classification dataset in SemEval 2013 show that, our joint model performs comparably with the state-of-the-art methods.

1 Introduction

Sentiment classification, which classifies the sentiment polarity of a sentence (or document) as positive or negative, is a major research direction in the field of sentiment analysis (Pang and Lee, 2008; Liu, 2012; Feldman, 2013). Majority of existing approaches follow Pang et al. (2002) and treat sentiment classification as a special case of text categorization task. Under this perspective, previous studies typically use pipelined methods with two steps. They first produce sentence segmentations with separate text analyzers (Choi and Cardie, 2008; Nakagawa et al., 2010; Socher et al., 2013b) or bag-of-words (Paltoglou and Thelwall, 2010; Maas et al., 2011). Then, feature learning and sentiment classification algorithms take the segmentation results as inputs to build the sentiment classifier (Socher et al., 2011; Kalchbrenner et al., 2014; Dong et al., 2014).

The major disadvantage of a pipelined method is the problem of error propagation, since sentence segmentation errors cannot be corrected by the sentiment classification model. A typical kind of error is caused by the *polarity inconsistency* between a phrase and the words it contains, such as ⟨*not bad*, *bad*⟩ and ⟨*a great deal of*, *great*⟩. The segmentations based on bag-of-words or syntactic chunkers are not effective enough to handle the *polarity inconsistency* phenomenons. The reason lies in that bag-of-words segmentations regard each word as a separate unit, which losses the word order and does not capture the phrasal information. The segmentations based on syntactic chunkers typically aim to identify noun groups, verb groups or named entities from a sentence. However, many sentiment indicators are phrases constituted of adjectives, negations, adverbs or idioms (Liu, 2012; Mohammad et al., 2013a), which are splitted by syntactic chunkers. Besides, a better approach would be to utilize the sentiment information to improve the segmentor. Accordingly, the sentiment-specific segmentor will enhance the performance of sentiment classification in turn.

In this paper, we propose a joint segmentation and classification framework (**JSC**) for sentiment analysis, which simultaneous conducts sentence segmentation and sentence-level sentiment classification. The framework is illustrated in Fig-

[*] This work was partly done when the first and fourth authors were visiting Microsoft Research.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 477–487,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

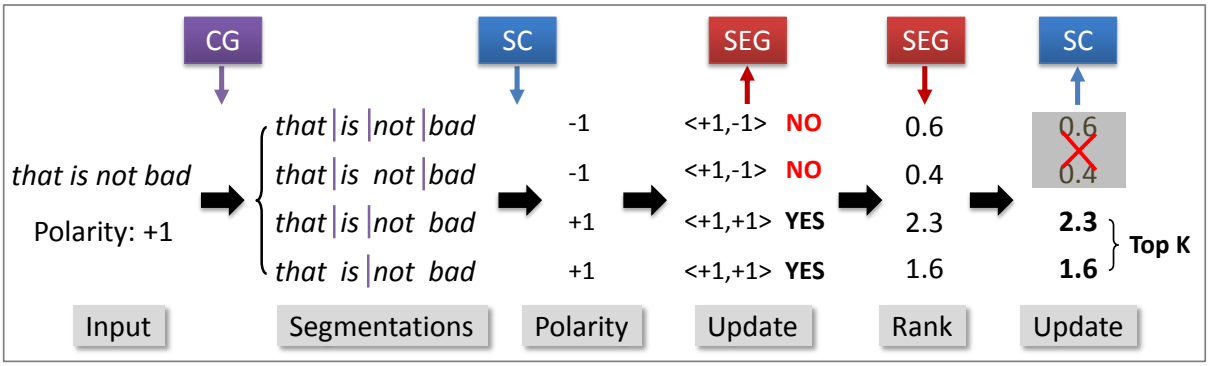


Figure 1: The joint segmentation and classification framework (**JSC**) for sentiment classification. **CG** represents the candidate generation model, **SC** means the sentiment classification model and **SEG** stands for the segmentation ranking model. **Down Arrow** means the use of a specified model, and **Up Arrow** indicates the update of a model.

ure 1. We develop (1) a candidate generation model to generate the segmentation candidates of a sentence, (2) a segmentation ranking model to score each segmentation candidate of a given sentence, and (3) a classification model to predict the sentiment polarity of each segmentation. The phrasal information of top-ranked candidates from the segmentation model are utilized as features to build the sentiment classifier. In turn, the predicted sentiment polarity of segmentation candidates from classification model are leveraged to update the segmentor. We score each segmentation candidate with a log-linear model, and optimize the segmentor with a marginal log-likelihood objective. We train the joint model from sentences annotated only with sentiment polarity, without any segmentation annotations.

We evaluate the effectiveness of our joint model on a benchmark Twitter sentiment classification dataset in SemEval 2013. Results show that the joint model performs comparably with state-of-the-art methods, and consistently outperforms pipeline methods in various experiment settings. The main contributions of the work presented in this paper are as follows.

- To our knowledge, this is the first work that automatically produces sentence segmentation for sentiment classification within a joint framework.

- We show that the joint model yields comparable performance with the state-of-the-art methods on the benchmark Twitter sentiment classification datasets in SemEval 2013.

2 Related Work

Existing approaches for sentiment classification are dominated by two mainstream directions. Lexicon-based approaches (Turney, 2002; Ding et al., 2008; Taboada et al., 2011; Thelwall et al., 2012) typically utilize a lexicon of sentiment words, each of which is annotated with the sentiment polarity or sentiment strength. Linguistic rules such as intensifications and negations are usually incorporated to aggregate the sentiment polarity of sentences (or documents). Corpus-based methods treat sentiment classification as a special case of text categorization task (Pang et al., 2002). They mostly build the sentiment classifier from sentences (or documents) with manually annotated sentiment polarity or distantly-supervised corpora collected by sentiment signals like emoticons (Go et al., 2009; Pak and Paroubek, 2010; Kouloumpis et al., 2011; Zhao et al., 2012).

Majority of existing approaches follow Pang et al. (2002) and employ corpus-based method for sentiment classification. Pang et al. (2002) pioneer to treat the sentiment classification of reviews as a special case of text categorization problem and first investigate machine learning methods. They employ Naive Bayes, Maximum Entropy and Support Vector Machines (SVM) with a diverse set of features. In their experiments, the best performance is achieved by SVM with bag-of-words feature. Under this perspective, many studies focus on designing or learning effective features to obtain better classification performance. On movie or product reviews, Wang and Manning (2012) present NBSVM, which trades-off

between Naive Bayes and NB-feature enhanced SVM. Kim and Zhai (2009) and Paltoglou and Thelwall (2010) learn the feature weights by investigating variants weighting functions from Information Retrieval. Nakagawa et al. (2010) utilize dependency trees, polarity-shifting rules and conditional random fields (Lafferty et al., 2001) with hidden variables to compute the document feature. On Twitter, Mohammad et al. (2013b) develop a state-of-the-art Twitter sentiment classifier in SemEval 2013, using a variety of sentiment lexicons and hand-crafted features.

With the revival of deep learning (representation learning (Hinton and Salakhutdinov, 2006; Bengio et al., 2013; Jones, 2014)), more recent studies focus on learning the low-dimensional, dense and real-valued vector as text features for sentiment classification. Glorot et al. (2011) investigate Stacked Denoising Autoencoders to learn document vector for domain adaptation in sentiment classification. Yessenalina and Cardie (2011) represent each word as a matrix and compose words using iterated matrix multiplication. Socher et al. propose Recursive Autoencoder (RAE) (2011), Matrix-Vector Recursive Neural Network (MV-RNN) (2012) and Recursive Neural Tensor Network (RNTN) (2013b) to learn the composition of variable-length phrases based on the representation of its children. To learn the sentence representation, Kalchbrenner et al. (2014) exploit Dynamic Convolutional Neural Network and Le and Mikolov (2014) investigate Paragraph Vector. To learn word vectors for sentiment analysis, Maas et al. (2011) propose a probabilistic document model following Blei et al. (2003), Labutov and Lipson (2013) re-embed words from existing word embeddings and Tang et al. (2014b) develop three neural networks to learn word vectors from tweets containing positive/negative emoticons.

Unlike most previous corpus-based algorithms that build sentiment classifier based on splitting a sentence as a word sequence, we produce sentence segmentations automatically within a joint framework, and conduct sentiment classification based on the segmentation results.

3 The Proposed Approach

In this section, we first give the task definition of two tasks, namely sentiment classification and sentence segmentation. Then, we present the overview of the proposed joint segmentation and classification model (**JSC**) for sentiment analysis. The segmentation candidate generation model and the segmentation ranking model are described in Section 4. The details of the sentiment classification model are presented in Section 5.

3.1 Task Definition

The task of sentiment classification has been well formalized in previous studies (Pang and Lee, 2008; Liu, 2012). The objective is to identify the sentiment polarity of a sentence (or document) as positive or negative [1].

The task of sentence segmentation aims to split a sentence into a sequence of exclusive parts, each of which is a basic computational unit of the sentence. An example is illustrated in Table 1. The original text "*that is not bad*" is segmented as "*[that] [is] [not bad]*". The segmentation result is composed of three basic computational units, namely *[that]*, *[is]* and *[not bad]*.

Type	**Sample**
Sentence	that is not bad
Segmentation	[that] [is] [not bad]
Basic units	[that], [is], [not bad]

Table 1: Example for sentence segmentation.

3.2 Joint Model (JSC)

The overview of the proposed joint segmentation and classification model (**JSC**) for sentiment analysis is illustrated in Figure 1. The intuitions of the joint model are two-folds:

- The segmentation results have a strong influence on the sentiment classification performance, since they are the inputs of the sentiment classification model.
- The usefulness of a segmentation can be judged by whether the sentiment classifier can use it to predict the correct sentence polarity.

Based on the mutual influence observation, we formalize the joint model in Algorithm 1. The inputs contain two parts, training data and feature extractors. Each sentence s_i in the training data

[1] In this paper, the sentiment polarity of a sentence is not relevant to the target (or aspect) it contains (Hu and Liu, 2004; Jiang et al., 2011; Mitchell et al., 2013).

Algorithm 1 The joint segmentation and classification framework (**JSC**) for sentiment analysis

Input:

training data: $T = [s_i, pol_i^g]$, $1 \leq i \leq |T|$
segmentation feature extractor: $sfe(\cdot)$
classification feature extractor: $cfe(\cdot)$

Output:

sentiment classifier: ***SC***
segmentation ranking model: ***SEG***

1: Generate segmentation candidates Ω_i for each sentence s_i in T, $1 \leq i \leq |T|$
2: Initialize sentiment classifier $\boldsymbol{SC}^{(0)}$ based on $cfe(\Omega_{ij})$, *randomize* $j \in [1, |\Omega_i|]$, $1 \leq i \leq |T|$
3: Randomly initialize the segmentation ranking model $\boldsymbol{SEG}^{(0)}$
4: **for** $r \leftarrow 1 \ldots R$ **do**
5: Predict the sentiment polarity pol_i for Ω_i based on $\boldsymbol{SC}^{(r-1)}$ and $cfe(\Omega_{i\cdot})$
6: Update the segmentation model $\boldsymbol{SEG}^{(r)}$ with $\boldsymbol{SEG}^{(r-1)}$ and $[\Omega_i, sfe(\Omega_{i\cdot}), \text{pol}_{i\cdot}, pol_i^g]$, $1 \leq i \leq |T|$
7: **for** $i \leftarrow 1 \ldots |T|$ **do**
8: Calculate the segmentation score for $\Omega_{i\cdot}$ based on $\boldsymbol{SEG}^{(r)}$ and $sfe(\Omega_{i\cdot})$
9: Select the top-ranked K segmentation candidates Ω_{i*} from Ω_i
10: **end for**
11: Train the sentiment classifier $\boldsymbol{SC}^{(r)}$ with $cfe(\Omega_{i*})$, $1 \leq i \leq |T|$
12: **end for**
13: $\boldsymbol{SC} \leftarrow \boldsymbol{SC}^{(R)}$
14: $\boldsymbol{SEG} \leftarrow \boldsymbol{SEG}^{(R)}$

T is annotated only with its gold sentiment polarity pol_i^g, without any segmentation annotations. There are two feature extractors for the task of sentence segmentation ($sfe(\cdot)$) and sentiment classification ($cfe(\cdot)$), respectively. The outputs of the joint model are the segmentation ranking model ***SEG*** and the sentiment classifier ***SC***.

In Algorithm 1, we first generate segmentation candidates Ω_i for each sentence s_i in the training set (line 1). Each Ω_i contains no less than one segmentation candidates. We randomly select one segmentation result from each Ω_i and utilize their classification features to initialize the sentiment classifier $\boldsymbol{SC}^{(0)}$ (line 2). We randomly initialize the segmentation model $\boldsymbol{SEG}^{(0)}$ (line 3). Subsequently, we iteratively train the segmentation model $\boldsymbol{SEG}^{(r)}$ and sentiment classifier $\boldsymbol{SC}^{(r)}$ in a joint manner (line 4-12). At each iteration, we predict the sentiment polarity of each segmentation candidate $\Omega_{i\cdot}$ with the current sentiment classifier $\boldsymbol{SC}^{(r-1)}$ (line 5), and then leverage them to update the segmentation model $\boldsymbol{SEG}^{(r)}$ (line 6). Afterwards, we utilize the recently updated segmentation ranking model $\boldsymbol{SEG}^{(r)}$ to update the sentiment classifier $\boldsymbol{SC}^{(r)}$ (line 7-11). We extract the segmentation features for each segmentation candidate $\Omega_{i\cdot}$, and employ them to calculate the segmentation score (line 8). The top-ranked K segmentation results Ω_{i*} of each sentence s_i is selected (line 9), and further used to train the sentiment classifier $\boldsymbol{SC}^{(r)}$ (line 11). Finally, after training R iterations, we dump the segmentation ranking model $\boldsymbol{SEG}^{(R)}$ and sentiment classifier $\boldsymbol{SC}^{(R)}$ in the last iteration as outputs (line 13-14).

At training time, we train the segmentation model and classification model from sentences with manually annotated sentiment polarity. At prediction time, given a test sentence, we generate its segmentation candidates, and then calculate segmentation score for each candidate. Afterwards, we select the top-ranked K candidates and vote their predicted sentiment polarity from sentiment classifier as the final result.

4 Segmentation Model

In this section, we present details of the segmentation candidate generation model (Section 4.1), the segmentation ranking model (Section 4.2) and the feature description for segmentation ranking model (Section 4.3).

4.1 Segmentation Candidate Generation

In this subsection, we describe the strategy to generate segmentation candidates for each sentence. Since the segmentation results have an exponential search space in the number of words in a sentence, we approximate the computation using beam search with constrains on a phrase table, which is induced from massive corpora.

Many studies have been previously proposed to recognize phrases in the text. However, it is out of scope of this work to compare them. We exploit a data-driven approach given by Mikolov et al. (2013), which identifies phrases based on the occurrence frequency of unigrams and bigrams,

$$freq(w_i, w_j) = \frac{freq(w_i, w_j) - \delta}{freq(w_i) \times freq(w_j)} \quad (1)$$

where δ is a discounting coefficient that prevents too many phrases consisting of very infrequent words. We run 2-4 times over the corpora to get longer phrases containing more words. We empirically set δ as 10 in our experiment. We use the default frequency threshold (value=5) in the word2vec toolkit [2] to select bi-terms.

Given a sentence, we initialize the beam of each index with the current word, and sequentially add phrases into the beam if the new phrase is contained in the phrase table. At each index of a sentence, we rank the segmentation candidates by the inverted number of items within a segmentation, and save the top-ranked N segmentation candidates into the beam. An example of the generated segmentation candidates is given in Table 2.

Type	Sample
Sentence	that is not bad
Phrase Table	[is not], [not bad], [is not bad]
Segmentations	[that] [is not bad] [that] [is not] [bad] [that] [is] [not bad] [that] [is] [not] [bad]

Table 2: Example for segmentation candidate generation.

4.2 Segmentation Ranking Model

The objective of the segmentation ranking model is to assign a scalar to each segmentation candidate, which indicates the usefulness of the segmentation result for sentiment classification. In this subsection, we describe a log-linear model to calculate the segmentation score. To effectively train the segmentation ranking model, we devise a marginal log-likelihood as the optimization objective.

Given a segmentation candidate Ω_{ij} of the sentence s_i, we calculate the segmentation score for Ω_{ij} with a log-linear model, as given in Equation 2.

$$\phi_{ij} = exp(b + \sum_{k} sfe_{ijk} \cdot w_k) \qquad (2)$$

where ϕ_{ij} is the segmentation score of Ω_{ij}; sfe_{ijk} is the k-th segmentation feature of Ω_{ij}; w and b are the parameters of the segmentation ranking model.

During training, given a sentence s_i and its gold sentiment polarity pol_i^g, the optimization objective of the segmentation ranking model is to maximize the segmentation scores of the **hit candidates**, whose predicted sentiment polarity equals to the gold polarity of sentence pol_i^p. The loss function of the segmentation model is given in Equation 3.

$$loss = -\sum_{i=1}^{|T|} log(\frac{\sum_{j \in H_i} \phi_{ij}}{\sum_{j' \in A_i} \phi_{ij'}}) + \lambda ||w||_2^2 \qquad (3)$$

where T is the training data; A_i represents all the segmentation candidates of sentence s_i; H_i means the hit candidates of s_i; λ is the weight of the L2-norm regularization factor. We train the segmentation model with L-BFGS (Liu and Nocedal, 1989), running over the complete training data.

4.3 Feature

We design two kinds of features for sentence segmentation, namely the phrase-embedding feature and the segmentation-specific feature. The final feature representation of each segmentation is the concatenation of these two features. It is worth noting that, the phrase-embedding feature is used in both sentence segmentation and sentiment classification.

Segmentation-Specific Feature We empirically design four segmentation-specific features to reflect the information of each segmentation, as listed in Table 3.

Phrase-Embedding Feature We leverage phrase embedding to generate the features of segmentation candidates for both sentence segmentation and sentiment classification. The reason is that, in both tasks, the basic computational units of each segmentation candidate might be words or phrases of variable length. Under this scenario, phrase embedding is highly suitable as it is capable to represent phrases with different length into a consistent distributed vector space (Mikolov et al., 2013). For each phrase, phrase embedding is a dense, real-valued and continuous vector. After the phrase embedding is trained, the nearest neighbors in the embedding space are favored to have similar grammatical usages and semantic meanings. The effectiveness of phrase embedding has been verified for building large-scale sentiment lexicon (Tang et al., 2014a) and machine translation (Zhang et al., 2014).

We learn phrase embedding with Skip-Gram model (Mikolov et al., 2013), which is the state-of-

[2] Available at https://code.google.com/p/word2vec/

Feature	Feature Description
#unit	the number of basic computation units in the segmentation candidate
#unit / #word	the ratio of units' number in a candidate to the length of original sentence
#word − #unit	the difference between sentence length and the number of basic computational units
#unit > 2	the number of basic component units composed of more than two words

Table 3: Segmentation-specific features for segmentation ranking.

Feature	Feature Description
All-Caps	the number of words with all characters in upper case
Emoticon	the presence of positive (or negative) emoticons, whether the last unit is emoticon
Hashtag	the number of hashtag
Elongated units	the number of basic computational containing elongated words (with one character repeated more than two times), such as *gooood*
Sentiment lexicon	the number of sentiment words, the score of last sentiment words, the total sentiment score and the maximal sentiment score for each lexicon
Negation	the number of negations as individual units in a segmentation
Bag-of-Units	an extension of bag-of-word for a segmentation
Punctuation	the number of contiguous sequences of dot, question mark and exclamation mark.
Cluster	the presence of units from each of the 1,000 clusters from Twitter NLP tool (Gimpel et al., 2011)

Table 4: Classification-specific features for sentiment classification.

the-art phrase embedding learning algorithm. We compose the representation (or feature) of a segmentation candidate from the embedding of the basic computational units (words or phrases) it contains. In this paper, we explore min, max and $average$ convolution functions, which have been used as simple and effective methods for composition learning in vector-based semantics (Mitchell and Lapata, 2010; Collobert et al., 2011; Socher et al., 2013a; Shen et al., 2014; Tang et al., 2014b), to calculate the representation of a segmentation candidate. The final phrase-embedding feature is the concatenation of vectors derived from different convolutional functions, as given in Equation 4,

$$pf(seg) = [pf_{max}(seg), pf_{min}(seg), pf_{avg}(seg)] \quad (4)$$

where $pf(seg)$ is the representation of the given segmentation; $pf_x(seg)$ is the result of the convolutional function $x \in \{min, max, avg\}$. Each convolutional function $pf_x(\cdot)$ conducts the matrix-vector operation of x on the sequence represented by columns in the lookup table of phrase embedding. The output of $pf_x(\cdot)$ is calculated as

$$pf_x(seg) = \theta_x \langle L_{ph} \rangle_{seg} \quad (5)$$

where θ_x is the convolutional function of pf_x; $\langle L_{ph} \rangle_{seg}$ is the concatenated column vectors of the basic computational units in the segmentation; L_{ph} is the lookup table of phrase embedding.

5 Classification Model

For sentiment classification, we follow the supervised learning framework (Pang et al., 2002) and build the classifier from sentences with manually labelled sentiment polarity. We extend the state-of-the-art hand-crafted features in SemEval 2013 (Mohammad et al., 2013b), and design the classification-specific features for each segmentation. The detailed feature description is given in Table 4.

6 Experiment

In this section, we conduct experiments to evaluate the effectiveness of the joint model. We describe the experiment settings and the result analysis.

6.1 Dataset and Experiment Settings

We conduct sentiment classification of tweets on a benchmark Twitter sentiment classification dataset in SemEval 2013. We run 2-class (positive *vs* negative) classification as sentence segmentation has a great influence on the positive/negative polarity of tweets due to the *polarity inconsistency* between a phrase and its constitutes, such as ⟨*not bad*, *bad*⟩.

We leave 3-class classification (positive, negative, neutral) and fine-grained classification (very negative, negative, neutral, positive, very positive) in the future work.

	Positive	Negative	Total
Train	2,642	994	3,636
Dev	408	219	627
Test	1,570	601	2,171

Table 5: Statistics of the SemEval 2013 Twitter sentiment classification dataset (positive *vs* negative).

The statistics of our dataset crawled from SemEval 2013 are given in Table 5. The evaluation metric is the macro-F1 of sentiment classification. We train the joint model on the training set, tune parameters on the dev set and evaluate on the test set. We train the sentiment classifier with LibLinear (Fan et al., 2008) and utilize existing sentiment lexicons [3] to extract classification-specific features. We randomly crawl 100M tweets from February 1st, 2013 to April 30th, 2013 with Twitter API, and use them to learn the phrase embedding with Skip-Gram [4]. The vocabulary size of the phrase embedding is 926K, from unigram to 5-gram. The parameter -c in SVM is tuned on the dev-set in both baseline and our method. We run the L-BFGS for 50 iterations, and set the regularization factor λ as 0.003. The beam size N of the candidate generation model and the top-ranked segmentation number K are tuned on the dev-set.

6.2 Baseline Methods

We compare the proposed joint model with the following sentiment classification algorithms:

• *DistSuper*: We collect 10M balanced tweets selected by positive and negative emoticons [5] as training data, and build classifier using the LibLinear and ngram features (Go et al., 2009; Zhao et al., 2012).

• *SVM*: The n-gram features and Support Vector Machine are widely-used baseline methods to build sentiment classifiers (Pang et al., 2002). We use LibLinear to train the SVM classifier.

• *NBSVM*: NBSVM (Wang and Manning, 2012) trades-off between Naive Bayes and NB-features enhanced SVM. We use NBSVM-bi because it performs best on sentiment classification of reviews.

• *RAE*: Recursive Autoencoder (Socher et al., 2011) has been proven effective for sentiment classification by learning sentence representation. We train the RAE using the pre-trained phrase embedding learned from 100M tweets.

• *SentiStrength*: Thelwall et al. (2012) build a lexicon-based classifier which uses linguistic rules to detect the sentiment strength of tweets.

• *SSWE$_u$*: Tang et al. (2014b) propose to learn sentiment-specific word embedding (SSWE) from 10M tweets collected by emoticons. They apply SSWE as features for Twitter sentiment classification.

• *NRC*: NRC builds the state-of-the-art system in SemEval 2013 Twitter Sentiment Classification Track, incorporating diverse sentiment lexicons and hand-crafted features (Mohammad et al., 2013b). We re-implement this system because the codes are not publicly available. We do not directly report their results in the evaluation task, as our training and development sets are smaller than their dataset. In *NRC + PF*, We concatenate the NRC features and the phrase embeddings feature (*PF*), and build the sentiment classifier with LibLinear.

Except for *DistSuper*, other baseline methods are conducted in a supervised manner. We do not compare with *RNTN* (Socher et al., 2013b) because the tweets in our dataset do not have accurately parsed results. Another reason is that, due to the differences between domains, the performance of *RNTN* trained on movie reviews might be decreased if directly applied on the tweets (Xiao et al., 2013).

6.3 Results and Analysis

Table 6 shows the macro-F1 of the baseline systems as well as our joint model (**JSC**) on sentiment classification of tweets (positive *vs* negative).

As is shown in Table 6, distant supervision is relatively weak because the noisy-labeled tweets are treated as the gold standard, which decreases the performance of sentiment classifier. The result of bag-of-unigram feature (74.50%) is not satisfied as it losses the word order and does not well cap-

[3] In this work, we use *HL* (Hu and Liu, 2004), *MPQA* (Wilson et al., 2005), *NRC_Emotion_Lexicon* (Mohammad and Turney, 2012), *NRC_Hashtag_Lexicon* and *Sentiment140Lexicon* (Mohammad et al., 2013b).

[4] https://code.google.com/p/word2vec/

[5] We use the emoticons selected by Hu et al. (2013). The positive emoticons are :) :) :-) :D =), and the negative emoticons are :(: (:-(.

Method	Macro-F1
DistSuper + unigram	61.74
DistSuper + 5-gram	63.92
SVM + unigram	74.50
SVM + 5-gram	74.97
Recursive Autoencoder	75.42
NBSVM	75.28
SentiStrength	73.23
$SSWE_u$	84.98
NRC (Top System in SemEval 2013)	84.73
NRC + PF	84.75
JSC	**85.51**

Table 6: Macro-F1 for positive *vs* negative classification of tweets.

ture the semantic meaning of phrases. The integration of high-order n-ngram (up to 5-gram) does not achieve significant improvement (+0.47%). The reason is that, if a sentence contains a bigram "*not bad*", they will use "*bad*" and "*not bad*" as parallel features, which confuses the sentiment classification model. *NBSVM* and *Recursive Autoencoder* perform comparatively and have a big gap in comparison with *JSC*. In *RAE*, the representation of a sentence is composed from the representation of words it contains. Accordingly, "*great*" in "*a great deal of*" also contributes to the final sentence representation via composition function. *JSC* automatically conducts sentence segmentation by considering the sentiment polarity of sentence, and utilize the phrasal information from the segmentations. Ideally, *JSC* regards phrases like "*not bad*" and "*a great deal of*" as basic computational units, and yields better classification performance. *JSC* (85.51%) performs slightly better than the state-of-the-art systems (*$SSWE_u$*, 84.98%; *NRC+PF*, 84.75%), which verifies its effectiveness.

6.4 Comparing Joint and Pipelined Models

We compare the proposed joint model with pipelined methods on Twitter sentiment classification with different feature sets. Figure 2 gives the experiment results. The tick *[A, B]* on x-axis means the use of *A* as segmentation feature and the use of *B* as classification feature. *PF* represents the phrase-embedding feature; *SF* and *CF* stand for the segmentation-specific feature and classification-specific feature, respectively. We use the bag-of-word segmentation result to build sentiment classier in *Pipeline 1*, and use the segmentation candidate with maximum phrase number in *Pipeline 2*.

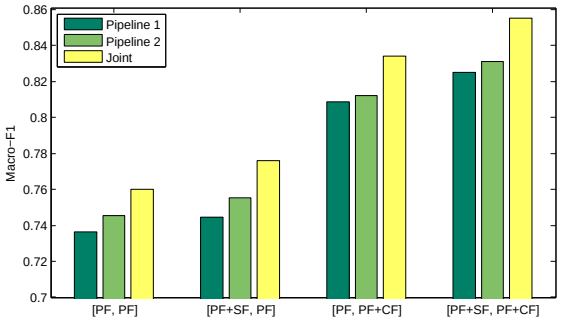


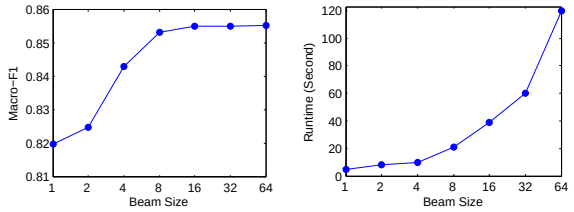
Figure 2: Macro-F1 for positive *vs* negative classification of tweets with joint and pipelined models.

From Figure 2, we find that the joint model consistently outperforms pipelined baseline methods in all feature settings. The reason is that the pipelined methods suffer from error propagation, since the errors from linguistic-driven and bag-of-word segmentations cannot be corrected by the sentiment classification model. Besides, traditional segmentors do not update the segmentation model with the sentiment information of text. Unlike pipelined methods, the joint model is capable to address these problems by optimizing the segmentation model with the classification results in a joint framework, which yields better performance on sentiment classification. We also find that *Pipeline 2* always outperforms *Pipeline 1*, which indicates the usefulness of phrase-based segmentation for sentiment classification.

6.5 Effect of the beam size N

We investigate the influence of beam size N, which is the maximum number of segmentation candidates of a sentence. In this part, we clamp the feature set as [PF+SF, PF+CF], and vary the beam size N in [1,2,4,8,16,32,64]. The experiment results of macro-F1 on the development set are illustrated in Figure 3 (a). The time cost of each training iteration is given in Figure 3 (b).

From Figure 3 (a), we can see that when larger beam size is considered, the classification performance is improved. When beam size is 1, the model stands for the greedy search with the bag-of-words segmentation. When the beam size is small, such as 2, beam search losses many phrasal information of sentences and thus the improvement is not significant. The performance remains steady when beam size is larger than 16. From



(a) Macro-F1 score for sentiment classification.

(b) Time cost (seconds) of each training iteration.

Figure 3: Sentiment classification of tweets with different beam size N.

Figure 3 (b), we can find that the runtime of each training iteration increases with larger beam size. It is intuitive as the joint model with larger beam considers more segmentation results, which increases the training time of the segmentation model. We set beam size as 16 after parameter learning.

6.6 Effect of the top-ranked segmentation number K

We investigate how the top-ranked segmentation number K affects the performance of sentiment classification. In this part, we set the feature as [PF+SF, PF+CF], and the beam size as 16. The results of macro-F1 on the development set are illustrated in Figure 4.

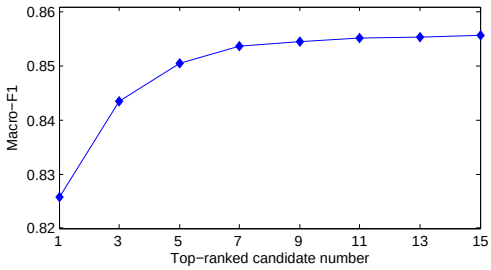


Figure 4: Sentiment classification of tweets with different top-ranked segmentation number K.

From Figure 4, we find that the classification performance increases with K being larger. The reason is that when a larger K is used, (1) at training time, the sentiment classifier is built by using more phrasal information from multiple segmentations, which benefits from the ensembles; (2) at test time, the joint model considers several top-ranked segmentations and get the final sentiment polarity through voting. The performance remains stable when K is larger than 7, as the phrasal information has been mostly covered.

7 Conclusion

In this paper, we develop a joint segmentation and classification framework (**JSC**) for sentiment analysis. Unlike existing sentiment classification algorithms that build sentiment classifier based on the segmentation results from bag-of-words or separate segmentors, the proposed joint model simultaneously conducts sentence segmentation and sentiment classification. We introduce a marginal log-likelihood function to optimize the segmentation model, and effectively train the joint model from sentences annotated only with sentiment polarity, without segmentation annotations of sentences. The effectiveness of the joint model has been verified by applying it on the benchmark dataset of Twitter sentiment classification in SemEval 2013. Results show that, the joint model performs comparably with state-of-the-art methods, and outperforms pipelined methods in various settings. In the future, we plan to apply the joint model on other domains, such as movie/product reviews.

Acknowledgements

We thank Nan Yang, Yajuan Duan, Yaming Sun and Meishan Zhang for their helpful discussions. We thank the anonymous reviewers for their insightful comments and feedbacks on this work. This research was partly supported by National Natural Science Foundation of China (No.61133012, No.61273321, No.61300113). The contact author of this paper, according to the meaning given to this role by Harbin Institute of Technology, is Bing Qin.

References

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Analysis and Machine Intelligence*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 793–801.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa.

2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Xiaowen Ding, Bing Liu, and Philip S Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 231–240.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 49–54.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Ronen Feldman. 2013. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 42–47.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. *Proceedings of International Conference on Machine Learning*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12.

G.E. Hinton and R.R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Ming Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 168–177.

Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. 2013. Unsupervised sentiment analysis with emotional signals. In *Proceedings of the International World Wide Web Conference*, pages 607–618.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. *The Proceeding of Annual Meeting of the Association for Computational Linguistics*.

Nicola Jones. 2014. Computer science: The learning machines. *Nature*, 505(7482):146.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A sentence model based on convolutional neural networks. In *Procedding of the 52th Annual Meeting of Association for Computational Linguistics.*

Hyun Duk Kim and ChengXiang Zhai. 2009. Generating comparative summaries of contradictory opinions in text. In *Proceedings of CIKM 2009*. ACM.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! In *The International AAAI Conference on Weblogs and Social Media*.

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *Annual Meeting of the Association for Computational Linguistics*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of international conference on Machine learning*. ACM.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *Proceedings of International Conference on Machine Learning*.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Conference on Neural Information Processing Systems*.

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

Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1654.

Saif M Mohammad and Peter D Turney. 2012. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*.

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

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013b. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *Proceedings of the International Workshop on Semantic Evaluation*.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of Language Resources and Evaluation Conference*, volume 2010.

Georgios Paltoglou and Mike Thelwall. 2010. A study of information retrieval weighting schemes for sentiment analysis. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1386–1395.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 79–86.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 373–374.

Richard Socher, J. Pennington, E.H. Huang, A.Y. Ng, and C.D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Y Ng. 2013a. Reasoning with neural tensor networks for knowledge base completion. *The Conference on Neural Information Processing Systems*.

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

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and Ting Liu. 2014a. Building large-scale twitter-specific sentiment lexicon : A representation learning approach. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, pages 172–182.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014b. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1555–1565.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2012. Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology*, 63(1):163–173.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 417–424.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 90–94.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 347–354.

Min Xiao, Feipeng Zhao, and Yuhong Guo. 2013. Learning latent word representations for domain adaptation using supervised word clustering. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 152–162, October.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 172–182.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014. Bilingually-constrained phrase embeddings for machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 111–121.

Jichang Zhao, Li Dong, Junjie Wu, and Ke Xu. 2012. Moodlens: an emoticon-based sentiment analysis system for chinese tweets. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Positive Unlabeled Learning for Deceptive Reviews Detection

Yafeng Ren **Donghong Ji** **Hongbin Zhang**
Computer School
Wuhan University
Wuhan 430072, China
{renyafeng,dhji,zhanghongbin}@whu.edu.cn

Abstract

Deceptive reviews detection has attracted significant attention from both business and research communities. However, due to the difficulty of human labeling needed for supervised learning, the problem remains to be highly challenging. This paper proposed a novel angle to the problem by modeling PU (positive unlabeled) learning. A semi-supervised model, called mixing population and individual property PU learning (MPIPUL), is proposed. Firstly, some reliable negative examples are identified from the unlabeled dataset. Secondly, some representative positive examples and negative examples are generated based on LDA (Latent Dirichlet Allocation). Thirdly, for the remaining unlabeled examples (we call them *spy* examples), which can not be explicitly identified as positive and negative, two similarity weights are assigned, by which the probability of a *spy* example belonging to the positive class and the negative class are displayed. Finally, *spy* examples and their similarity weights are incorporated into SVM (Support Vector Machine) to build an accurate classifier. Experiments on gold-standard dataset demonstrate the effectiveness of MPIPUL which outperforms the state-of-the-art baselines.

1 Introduction

The Web has dramatically changed the way people express themselves and interact with others, people frequently write reviews on e-commerce sites, forums and blogs to achieve these purposes. For NLP (Natural Language Processing), these user-generated contents are of great value in that they contain abundant information related to people's opinions on certain topics. Currently, online reviews on products and services are used extensively by consumers and businesses to conduct decisive purchase, product design and marketing strategies. Hence, sentiment analysis and opinion mining based on product reviews have become a popular topic of NLP (Pang and Lee, 2008; Liu, 2012). However, since reviews information can guide people's purchase behavior, positive reviews can result in huge economic benefits and fame for organizations or individuals. This leaves room for promoting the generation of review spams. Through observations and studies of the predecessors (Jindal and Liu, 2008; Ott et al., 2011), review spams are divided into the following two classes:

- **Deceptive Reviews:** Those deliberately mislead readers by giving undeserving positive reviews to some target objects in order to promote the objects, or by giving unjust negative reviews to some target objects in order to damage their reputation.
- **Disruptive Reviews:** Those are non-reviews, which mainly include advertisements and other irrelevant reviews containing no opinion.

Disruptive reviews pose little threat to people, because human can easily identify and ignore them. In this paper, we focus on the more challenging ones: deceptive reviews. Generally, deceptive reviews detection is deemed to be a classification problem (Ott et al., 2011; Li et al., 2011; Feng et al., 2012). Based on the positive and negative examples annotated by people, supervised learning is utilized to build a classifier, and then an unlabeled review can be predicted as deceptive review or truthful one. But the work from Ott et al. (2011) shows that human cannot identify deceptive reviews from their prior knowledge, which indicates that human-annotated review datasets must

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 488–498,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

include some mislabeled examples. These examples will disturb the generation ability of the classifiers. So simple supervised learning is regarded as unsuitable for this task.

It is difficult to come by human labeling needed for supervised learning and evaluation, we cannot obtain the datasets containing deceptive reviews. However, we can get some truthful reviews with high confidence by heuristic rules and prior knowledge. Meanwhile, a lot of unlabeled reviews are available. The problem thus is this: based on some truthful reviews and a lot of unlabeled reviews, can we build an accurate classifier to identify deceptive reviews.

PU (positive unlabeled) learning can be utilized to deal with the above situation (Liu et al., 2002; Liu et al., 2003). Different from traditional supervised learning, PU learning can still build an accurate classifier even without the negative training examples. Several PU learning techniques have been applied successfully in document classification with promising results (Zhang, 2005; Elkan and Noto, 2008; Li et al., 2009; Xiao et al., 2011), while they have yet to be applied in detecting deceptive reviews. Here, we will study how to design PU learning to detect deceptive reviews.

An important challenge is how to deal with *spy* examples (easily mislabeled) of unlabeled reviews, which is not easily handled by the previous PU learning techniques. In this paper, we propose a novel approach, mixing population and individual property PU learning (MPIPUL), by assigning similarity weights and incorporating weights into SVM learning phase. This paper makes the following contributions:

- For the first time, PU learning is defined in the environment of identifying deceptive reviews.
- A novel PU learning is proposed based on LDA and SVM.
- Experimental results demonstrate that our proposed method outperforms the current baselines.

2 Related Work

2.1 Deceptive Reviews Detection

Spam has historically been investigated in the contexts of e-mail (Drucker et al., 1999; Gyongyi et al., 2004) and the Web (Ntoulas et al., 2006). In recent years, researchers have started to look at deceptive reviews.

Jindal and Liu (2008) found that opinion spam was widespread and different from e-mail and Web spam in essence (Jindal and Liu, 2008). They trained models using product review data, by defining features to distinguish duplicate opinion and non-duplicate based on the review text, reviewers and product information. Wu et al. (2010) proposed an alternative strategy of popularity rankings (Wu et al., 2010).

Ott et al. (2011) developed the first dataset containing gold-standard deceptive reviews by crowdsourcing (Ott et al., 2011), and presented three supervised learning methods to detect deceptive reviews by integrating knowledge from psycholinguistics and computational linguistics. This gold-standard dataset will be used in the paper. Li et al. (2011) manually built a review dataset from their crawled reviews (Li et al., 2011), and exploited semi-supervised co-training algorithm to identify deceptive reviews.

Feng et al. (2012) verified the connection between the deceptive reviews and the abnormal distributions (Feng et al., 2012a). Later, they (Feng et al., 2012b) demonstrated that features driven from CFG (Context Free Grammar) parsing trees consistently improve the detection performance.

Mukherjee et al. (2012) proposed detecting group spammers (a group of reviewers who work collaboratively to write deceptive reviews) in product reviews (Mukherjee et al., 2012). The proposed method first used frequent itemset mining to find a set of candidate groups. Then GSRank was presented which can consider relationships among groups, individual reviewers and products they reviewed to detect spammer groups. Later, they also proposed exploiting observed reviewing behaviors to detect opinion spammers in an unsupervised Bayesian inference framework (Mukherjee et al., 2013).

Ren et al. (2014) assumed that there must be some difference on language structure and sentiment polarity between deceptive reviews and truthful ones (Ren et al., 2014a), then they defined the features related to the review text and used genetic algorithm for feature selection, finally they combined two unsupervised clustering algorithm to identify deceptive reviews. Later, they (Ren et al., 2014b) present a new approach, from the viewpoint of correcting the mislabeled

examples, to find deceptive reviews. Firstly, they partition a dataset into several subsets.Then they construct a classifier set for each subset and select the best one to evaluate the whole dataset. Meanwhile, error variables are defined to compute the probability that the examples have been mislabeled. Finally, the mislabeled examples are corrected based on two threshold schemes, majority and non-objection.

Unlike previous studies, PU learning is implemented to identify deceptive reviews.

2.2 Positive Unlabeled Learning

According to the use of the unlabeled data, PU learning can be divided into two classes.

One family of methods built the final classifier by using positive examples dataset and some examples of the unlabeled dataset (Liu et al., 2002; Liu et al., 2003). The basic idea is to find a set of reliable negative examples from the unlabeled data firstly, and then to learn a classifier using EM (Expectation Maximization) or SVM. The performance is limited for neglecting the rest examples of unlabeled dataset.

Another family of methods learned the final classifier by using positive examples dataset and all examples of the unlabeled dataset. Li et al. (Li et al., 2009) studied PU learning in the data stream environment, they proposed a PU learning LELC (PU Learning by Extracting Likely positive and negative micro-Clusters) for document classification, they assume that the examples close together shared the same labels. Xiao et al. (Xiao et al., 2011) proposed a method, called SPUL (similarity-based PU learning), the local similarity-based and global similarity-based mechanisms are proposed to generate the similarity weights for the easily mislabeled examples, respectively. Experimental results show global SPUL generally performs better than local SPUL.

In this paper, a novel PU learning (MPIPUL) is proposed to identify deceptive reviews.

3 Preliminary

Before we introduce the proposed method, we briefly review SVM, which has proven to be an effective classification algorithm (Vapnik, 1998).

Let $T = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(|T|)}, y^{(|T|)})\}$ be a training set, where $x^{(i)} \in R^d$ and $y^{(i)} \in \{+1, -1\}$. SVM aims to seek an optimal separating hyperplane $w^T x^{(i)} + b = 0$, the hyperplane can be obtained by solving the following optimization problem:

$$\begin{aligned} min \quad & F(w, b, \epsilon_i) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{|T|}\epsilon_i \\ s.t. \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \epsilon_i, i = 1, \ldots, |T| \\ & \epsilon_i \geq 0, i = 1, \ldots, |T| \end{aligned} \tag{1}$$

where w^T represents the transpose of w, C is a parameter to balance the classification errors and ϵ_i are variables to relax the margin constraints. The optimal classifier can be achieved by using the Lagrange function. For a test example x, if $w^T x + b < 0$, it is classified into the negative class; otherwise, it is positive.

In the following, SVM is extended to incorporate the *spy* examples and their weights, such that the *spy* examples can contribute differently to the classifier construction.

4 The Proposed Method

In this section, we will introduce the proposed approach in details. In our PU learning (MPIPUL), truthful reviews are named positive examples, and deceptive reviews are called negative examples. P is defined as a set which contains all positive examples. U is a set for all unlabeled examples. PU learning aims at building a classifier using P and U. MPIPUL adopts the following four steps:

- Step 1: Extract the reliable negative examples;
- Step 2: Compute the representative positive and negative examples;
- Step 3: Generate the similarity weights for the *spy* examples;
- Step 4: Build the final SVM classifier;

4.1 Extracting Reliable Negative Examples

Considering only positive and unlabeled examples are available in PU learning, some negative examples need to be extracted firstly. These examples will influence the performance of the following three steps. So high-quality negative examples must be guaranteed. Previous works solved the problem with the Spy technique (Liu et al., 2002) or the Rocchio technique (Liu et al., 2003), we integrate them in order to get reliable negative examples. Let subsets NS_1 and NS_2 contain the

corresponding reliable negative examples extracted by the two techniques, respectively. Examples are considered to be a reliable negative only if both techniques agree that they are negative. That is, $NS = NS_1 \cap NS_2$, where NS contains the reliable negative examples.

After reliable negative examples are extracted, there are still some unlabeled examples (we call *spy* examples) in set U, let subset $US = U - NS$, which stores all the *spy* examples. It is crucial to determine how to deal with these *spy* examples.

4.2 Computing Representative Positive and Negative Examples

Generally, a classifier can be constructed to predict deceptive reviews based on the positive examples set P and the reliable negative examples set NS. But the classifier is not accurate enough for lacking of making full use of unlabeled dataset U. In order to utilize *spy* examples in subset US, some representative positive and negative examples are calculated firstly. Since the examples have different styles in sentiment polarity and topic distribution, for every class, computing one representative example is not suitable. For the positive class or the negative class, to ensure there is a big difference between the different representative examples. This paper proposes clustering reliable negative examples into several groups based on LDA (Latent Dirichlet Allocation) topic model and K-means, and then multiple representative examples can be obtained.

LDA topic model is known as a parametric Bayesian clustering model (Blei et al., 2003), and assumes that each document can be represented as the distribution of several topics, each document is associated with common topics. LDA can well capture the relationship between internal documents.

In our experiments based on LDA model, we can get the topic distribution for the reliable negative examples, then some reliable negative examples which are similar in topic distribution will be clustered into a group by K-means. Finally, these reliable negative examples can be clustered into n micro-clusters $(NS_1, NS_2, \ldots, NS_n)$. Here,

$$n = 30 * |NS|/(|US| + |NS|) \quad (2)$$

Here, according to the suggestion of previous work (Xiao et al., 2011), we examine the impact of the different parameter (from 10 to 60) on overall performance, and select the best value 30.

Based on the modified Rocchio formula (Buckley et al., 1999), n representative positive examples (p_k) and n negative ones (n_k) can be obtained using the following formula:

$$p_k = \alpha \frac{1}{|P|} \sum_{i=1}^{|P|} \frac{x^{(i)}}{\| x^{(i)} \|} - \beta \frac{1}{|NS_k|} \sum_{i=1}^{|NS_k|} \frac{x^{(i)}}{\| x^{(i)} \|}$$

$$n_k = \alpha \frac{1}{|NS_k|} \sum_{i=1}^{|NS_k|} \frac{x^{(i)}}{\| x^{(i)} \|} - \beta \frac{1}{|P|} \sum_{i=1}^{|P|} \frac{x^{(i)}}{\| x^{(i)} \|}$$

$$k = 1, \ldots, n \quad (3)$$

According to previous works (Buckley et al., 1994), where the value of α and β are set to 16 and 4 respectively. The research from Buckley et al. demonstrate that this combination emphasizes occurrences in the relevant documents as opposed to non-relevant documents.

4.3 Generating Similarity Weights

For a *spy* example x, since we do not know which class it should belong to, enforcing x to the positive class or the negative class will lead to some mislabeled examples, which disturbs the performance of final classifier. We represent a *spy* example x using the following probability model:

$$\{x, (p^+(x), p^-(x))\}, \quad p^+(x) + p^-(x) = 1 \quad (4)$$

Where $p^+(x)$ and $p^-(x)$ are similarity weights which represent the probability of x belonging to the positive class and the negative class, respectively. For example, $\{x, (1, 0)\}$ means that x is positive, while $\{x, (0, 1)\}$ indicates that x is identified to be negative. For $\{x, (p^+(x), p^-(x))\}$, where $0 < p^+(x) < 1$ and $0 < p^-(x) < 1$, it implies that the probability of x belonging to the positive class and the negative class are both considered.

In this section, similarity weights are decided by mixing global information (population property) and local information (individual property). Then all *spy* examples and their similarity weights are incorporated into a SVM-based learning model.

4.3.1 Population Property

Population property means that the examples in each micro-cluster share the similarity in sentiment polarity and topic distribution, and they belong to the same category with a high possibility. In our framework, in order to compare with the representative examples, all *spy* examples are firstly clustered into n micro-clusters

$(US_1, US_2, \ldots, US_n)$ based on LDA and K-means. Then, for every *spy* example x in one micro-cluster US_i, we tags with temporary label by finding its most similar representative example. Finally, we can get the similarity weights for a *spy* example x in micro-cluster US_i, their probability pertaining to the positive class and negative class can be represented by the following formula:

$$\begin{aligned} p\_pop(x) &= \frac{|positive|}{|US_i|} \\ n\_pop(x) &= \frac{|negative|}{|US_i|} \end{aligned} \tag{5}$$

where $|US_i|$ represents the number of all examples in micro-cluster US_i, $|positive|$ means the number of the examples which is called temporary positive in US_i, and $|negative|$ means the number of the examples which is called temporary negative in US_i.

For example, Figure 1 shows the part (C1, C2, C3, C4) of the clustering results for the *spy* examples based on LDA and K-means, the examples x in C4 are assigned with weights $p\_pop(x) = \frac{4}{9}, n\_pop(x) = \frac{5}{9}$, the examples x in C1 are assigned with weights $p\_pop(x) = 1, n\_pop(x) = 0$.

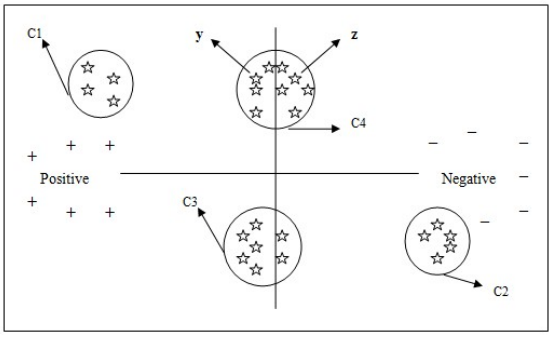


Figure 1: Illustration of population property

The advantage of population property lies in the fact that it considers the similar relationship between the examples, from which the same micro-cluster are assigned the same similarity weight. However, it cannot distinguish the difference of examples in one micro-cluster. In fact, the similarity weights of examples from the same micro-cluster can be different, since they are located physically different. For example, for the *spy* example y and z in micro-cluster C4, it is apparently unreasonable that we assign the same similarity weights to them. So we should join the local information (individual property) when we are computing the similarity weights for a *spy* example.

4.3.2 Individual Property

Individual property is taken into account to measure the relationship between every *spy* example and all representative ones. Specifically, for example x, we firstly compute its similarity to each of the representative examples, and then the probability of the example x belonging to the positive class and negative class can be calculated using the following formula:

$$\begin{aligned} p\_ind(x) &= \frac{\sum_{k=1}^{n} sim(x, p_k)}{\sum_{k=1}^{n}(sim(x, p_k) + sim(x, n_k))} \\ n\_ind(x) &= \frac{\sum_{k=1}^{n} sim(x, n_k)}{\sum_{k=1}^{n}(sim(x, p_k) + sim(x, n_k))} \end{aligned} \tag{6}$$

In the above formula,

$$sim(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||}$$

4.3.3 Similarity Weights

A scheme mixing population and individual property is designed to generate the similarity weights of *spy* examples. Specifically, for *spy* example x, their similarity weights can be obtained by the following formula:

$$\begin{aligned} p^+(x) &= \lambda \cdot p\_pop(x) + (1 - \lambda) \cdot p\_ind(x) \\ p^-(x) &= \lambda \cdot n\_pop(x) + (1 - \lambda) \cdot n\_ind(x) \end{aligned} \tag{7}$$

Where λ is a parameter to balance the information from population property and individual property. In the remaining section, we will examine the impact of the parameter λ on overall performance. Meanwhile, it can be easily proved that $p^+(x) + p^-(x) = 1$.

4.4 Constructing SVM Classifier

After performing the third step, each *spy* example x is assigned two similarity weights: $p^+(x)$ and $p^-(x)$. In this section, we will extend the formulation of SVM by incorporating the examples in positive set P, reliable negative set NS, *spy* examples set US and their similarity weights into a SVM-based learning model.

4.4.1 Primal Problem

Since the similarity weights $p^+(x)$ and $p^-(x)$ indicate the probability for a *spy* example x belonging to the positive class and the negative class, respectively. The optimization formula (1) can be

rewritten as the following optimization problem:

$$
\begin{aligned}
min \quad & F(w,b,\epsilon) = \frac{1}{2}||w||^2 + C_1 \sum_{i=1}^{|P|} \epsilon_i + C_2 \cdot \\
& \sum_{j=1}^{|US|} p^+(x^{(j)})\epsilon_j + C_3 \sum_{m=1}^{|US|} p^-(x^{(m)})\epsilon_m \\
& + C_4 \sum_{n=1}^{|NS|} \epsilon_n \\
s.t. \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \epsilon_i, \quad x^{(i)} \in P \\
& y^{(j)}(w^T x^{(j)} + b) \geq 1 - \epsilon_j, \quad x^{(j)} \in US \\
& y^{(m)}(w^T x^{(m)} + b) \geq 1 - \epsilon_m, \quad x^{(m)} \in US \\
& y^{(n)}(w^T x^{(n)} + b) \geq 1 - \epsilon_n, \quad x^{(n)} \in NS \\
& \epsilon_i \geq 0, \quad \epsilon_j \geq 0, \quad \epsilon_m \geq 0, \quad \epsilon_n \geq 0
\end{aligned} \tag{8}
$$

Where C_1, C_2, C_3 and C_4 are penalty factors controlling the tradeoff between the hyperplane margin and the errors, $\epsilon_i, \epsilon_j, \epsilon_m$ and ϵ_n are the error terms. $p^+(x^{(j)})\epsilon_j$ and $p^-(x^{(m)})\epsilon_m$ can be considered as errors with different weights. Note that, a bigger value of $p^+(x^{(j)})$ can increase the effect of parameter ϵ_j, so that the corresponding example $x^{(j)}$ becomes more significant towards the positive class. In the following, we will find the dual form to address the above optimization problem.

4.4.2 Dual Problem

Assume α_i and α_j are Lagrange multipliers. To simplify the presentation, we redefine some notations as follows:

$$
C_i^+ = \begin{cases} C_1, & x^{(i)} \in P \\ C_2 p^+(x^{(j)}), & x^{(j)} \in US \end{cases}
$$

$$
C_j^- = \begin{cases} C_3 p^-(x^{(m)}), & x^{(m)} \in US \\ C_4, & x^{(n)} \in NS \end{cases}
$$

Based on the above definitions, we let $T^+ = P \cup US$, $T^- = US \cup NS$ and $T^* = T^+ \cup T^-$. The Wolfe dual of primal formulation can be obtained as follows (Appendix A for the calculation process):

$$
\begin{aligned}
max \quad & W(\alpha) = \sum_{i=1}^{|T^*|} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{|T^*|} \alpha_i \alpha_j y^{(i)} \cdot \\
& y^{(j)} < x^{(i)}, x^{(j)} > \\
s.t. \quad & C_i^+ \geq \alpha_i \geq 0, \quad x^{(i)} \in T^+ \\
& C_j^- \geq \alpha_j \geq 0, \quad x^{(j)} \in T^- \\
& \sum_{i=1}^{|T^+|} \alpha_i - \sum_{j=1}^{|T^-|} \alpha_j = 0
\end{aligned} \tag{9}
$$

where $< x^{(i)}, x^{(j)} >$ is the inner product of $x^{(i)}$ and $x^{(j)}$. In order to get the better performance, we can replace them by using kernel function $\phi(x^{(i)})$ and $\phi(x^{(j)})$, respectively. The kernel track can convert the input space into a high-dimension feature space. It can solve the uneven distribution of dataset and complex problem from heterogeneous data sources, which allows data to get a better expression in the new space (Lanckriet et al., 2004; Lee et al., 2007).

After solving the above problem, w can be obtained, then b can also be obtained by using KKT (Karush-Kuhn-Tucker) conditions. For a test example x, if $w^T x + b > 0$, it belongs to the positive class. Otherwise, it is negative.

5 Experiments

We aim to evaluate whether our proposed PU learning can identify deceptive reviews properly. We firstly describe the gold-standard dataset, and then introduce the way to generate the positive examples P and unlabeled examples U. Finally we present human performance in gold-standard dataset.

5.1 Datasets

There is very little progress in detection of deceptive reviews, one reason is the lack of standard dataset for algorithm evaluation. The gold-standard dataset is created based on crowdsourcing platform (Ott et al., 2011), which is also adopted as the experimental dataset in this paper.

5.1.1 Deceptive Reviews

Crowdsourcing services can carry out massive data collection and annotation; it defines the task in the network platform, and paid for online anonymous workers to complete the task.

Humans cannot be precisely distinguish deceptive ones from existing reviews, but they can create deceptive reviews as one part of the dataset. Ott et al. (2011) accomplish this work by AMT (Amazon Mechanical Turk). They set 400 tasks for 20 hotels, in which each hotel gets 20 tasks. Specific task is: If you are a hotel market department employee, for each positive review you wrote for the benefit for hotel development, you may get one dollar. They collect 400 deceptive reviews.

5.1.2 Truthful Reviews

For the collection of truthful reviews, they get 6977 reviews from TripAdvisor[1] based on the same 20 Chicago hotels, and remove some reviews on the basis of the following constraints:

- Delete all non-five star reviews;
- Delete all non-English reviews;
- Delete all reviews which are less than 75 characters;
- Delete all reviews written by first-time authors;

2124 reviews are gathered after filtering. 400 of them are chosen as truthful ones for balancing the number of deceptive reviews, as well as maintaining consistent with the distribution of the length of deceptive reviews. 800 reviews constitute whole gold-standard dataset at last.

5.2 Experiment Setup

We conduct 10-fold cross-validation: the dataset is randomly split into ten folds, where nine folds are selected for training and the tenth fold for test. In training dataset, it contains 360 truthful reviews and 360 deceptive ones. This paper is intended to apply PU learning to identify deceptive reviews. We specially make the following setting: take 20% of the truthful reviews in training set as positive examples dataset P, all remaining truthful and deceptive reviews in training set as the unlabeled dataset U. Therefore, during one round of the algorithm, the training set contains 720 examples including 72 positive examples (set P) and 648 unlabeled examples (set U), and the test set contains 80 examples including 40 positive and 40 negative ones. In order to verify the stability of the proposed method, we also experiment another two different settings, which account for 30% and 40% of the truthful reviews in training set as positive examples dataset P respectively.

[1] http://www.tripadvisor.com

5.3 Human Performance

Human performance reflects the degree of difficulty to address this task. The rationality of PU learning is closely related to human performance.

We solicit the help of three volunteer students, who were asked to make judgments on test subset (corresponding to the tenth fold of our cross-validation experiments, contains 40 deceptive reviews and 40 truthful reviews). Additionally, to test the extent to which the individual human judges are biased, we evaluate the performance of two virtual meta-judges: one is the MAJORITY meta-judge when at last two out of three human judge believe the review to be deceptive, and the other is the SKEPTIC when any human judge believes the review to be deceptive. It is apparent from the results that human judges are not particularly effective at this task (Table 1). Inter-annotator agreement among the three judges, computed using Fleiss' kappa, is 0.09. Landis and Koch (Landis and Koch, 1977) suggest that scores in the range (0.00, 0.20) correspond to "slight agreemen" between annotators. The largest pairwise Cohen's kappa is 0.11 between JUDGE-1 and JUDGE-3, far below generally accepted pairwise agreement levels. We can infer that the dataset which are annotated by people will include a lot of mislabeled examples. Identifying deceptive reviews by simply using supervised learning methods is not appropriate. So we propose addressing this issue by using PU learning.

Table 1: Human performance

Methods		Accuracy (%)
Human	JUDGE-1	57.9
	JUDGE-2	55.4
	JUDGE-3	61.7
META	MAJORITY	58.3
	SKEPTIC	62.4

6 Results and Analysis

In order to verify the effectiveness of our proposed method, we perform two PU learning (LELC and SPUL) in the gold-standard dataset.

6.1 Experimental Results

Table 2 shows that the experimental results compared with different PU learning techniques. In Table 2, $P(20\%)$ means that we randomly select 20 percentages of truthful reviews to form the positive examples subset P. In our MPIPUL framework, we set $\lambda = 0.3$. We can see that our proposed method can obtain 83.91%, 85.43% and 86.69% in accuracy from different experimental settings, respectively. Compared to the current best method (SPUL-global), the accuracy can be improved 2.06% on average. MPUPUL can improve 3.21% on average than LELC. The above discussion shows our proposed methods consistently outperform the other PU baselines.

Table 2: Accuracy on the different PU learning

Baselines	P(20%)	P(30%)	P(40%)
LELC	81.12	82.08	83.21
SPUL-local	81.43	82.71	84.09
SPUL-global	81.89	83.24	84.73
MPIPUL (0.3)	83.91	85.43	86.69

PU learning framework in this paper can obtain the better performance. Two factors contribute to the improved performance. Firstly, LDA can capture the deeper information of the reviews in topic distribution. Secondly, strategies of mixing population and individual property can generate the similarity weights for *spy* examples, and these examples and their similarity weights are extended into SVM, which can build a more accurate classifier.

6.2 Parameter Sensitivity

For the *spy* examples, the similarity weights are generated by population property and individual property. Should we select the more population information or individual information? In MPIPUL, parameter λ is utilized to adjust this process. So we experiment with the different value of the parameter λ on MPUPUL performance (Figure 2).

As showed in Figure 2, for $P(20\%)$, if $\lambda < 0.3$, the performance increases linearly, if $\lambda > 0.3$, the performance will decrease linearly. Meanwhile, we can get the same trends for $P(30\%)$ and $P(40\%)$. Based on the above discussion, MPIPUL can get the best performance when $\lambda \approx 0.3$.

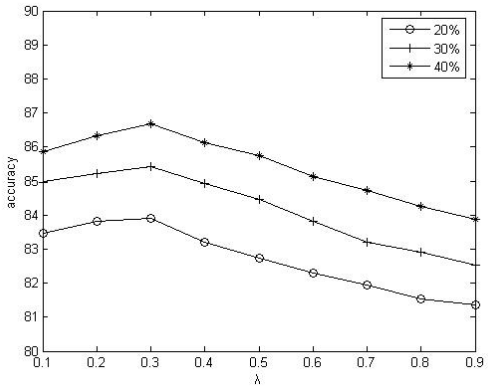


Figure 2: Algorithm performance on different parameter

7 Conclusions and Future Work

This paper proposes a novel PU learning (MPIPUL) technique to identify deceptive reviews based on LDA and SVM. Firstly, the *spy* examples are assigned similarity weights by integrating the information from the population property and individual property. Then the *spy* examples and their similarity weights are incorporated into SVM learning phase to build an accurate classifier. Experimental results on gold-standard dataset show the effectiveness of our method.

In future work, we will discuss the application of our proposed method in the massive dataset.

Acknowledgments

We are grateful to the anonymous reviewers for their thoughtful comments. This work is supported by the State Key Program of National Natural Science Foundation of China (Grant No.61133012), the National Natural Science Foundation of China (Grant No.61173062, 61373108) and the National Philosophy Social Science Major Bidding Project of China (Grant No. 11&ZD189).

References

Alexandros Ntoulas, Marc Najork, Mark Manasse, and Dennis Fetterly. 2006. Detecting spam web pages through content analysis. In *Proceedings of the 15th International Conference on World Wide Web*, page 83-92, Edinburgh, Scotland.

Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Spotting opinion spammers using behavioral footprints. In *Proceeding of the 19th*

ACM SIGKDD International Conference on Knowledge Discovery and Data Ming, page 632-640, Lyon, France.

Arjun Mukherjee, Bing Liu, and Natalie Glance. 2012. Spotting fake reviewer groups in consumer reviews. In *Proceeding of the 21st International Conference on World Wide Web*, page 191-200, New York, USA.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Morgan & Claypool Publishers*. San Rafael, USA.

Bing Liu, Wee Sun Lee, Philip S. Yu, and Xiaoli Li. 2002. Partially supervised classification of text documents. In *Proceedings of the 19th International Conference on Machine Learning*, page 387-394, San Francisco, USA.

Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S. Yu. 2003. Building text classifiers using positive and unlabeled examples. In *Proceedings of the 3rd IEEE International Conference on Data Ming*, page 179-182, Washington, USA.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1-135.

Charles Elkan and Keith Noto. 2008. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Ming*, page 213-220, Las Vegas, USA.

Chirs Buckley, Bgrard Salton, and James Allan. 1994. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the 17th Annual International SIGIR Conference on Research and Development Retrieval*, page 292-300, Dublin, Ireland.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993-1022.

Dell Zhang. 2005. A simple probabilistic approach to learning from positive and unlabeled examples. In *Proceedings of the 5th Annual UK Workshop on Computational Intelligence*, page 83-87.

Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. 2011. Learning to identify review spam. In *Proceeding of the 22nd International Joint Conference on Artificial Intelligence*, page 2488-2493, Barcelona, Spain.

Fang Wu and Bernardo A. Huberman. 2010. Opinion formation under costly express. *ACM Transactions on Intelligence System Technology*, 1(5):1-13.

Gert R. G. Lanckeriet, Nello Cristianini, Peter Bartlett, Laurent EI Ghaoui, and Michael I.Jordan. 2004. Learning the kernel matrix with seim-difinit programming. *Journal of Machine Learning Research*, 5:27-72.

Harris Drucker, Donghui Wu, and Vladimir N. Vapnik. 1999. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048-1054.

Kumar Ankita and Sminchisescu Cristian. 2006. Support kernel machines for object recognition. In *Proceedings of the IEEE 11th International Conference on Computer Vision*, page 1-8, Rio de Janeiro, Brzail.

Myle Ott, Yelin Choi, Claire Caridie, and Jeffrey T. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceeding of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technoloies*, page 309-319, Portland, USA.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceeding of the 1st ACM International Conference on Web Search and Data Mining*, page 137-142, California, USA.

Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159-174.

Song Feng, Longfei Xing, Anupam Gogar, and Yejin Choi. 2012. Distributional footprints of deceptive product reviews. In *Proceeding of the 6th International AAAI Conference on WebBlogs and Social Media*, page 98-105, Dublin, Ireland.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceeding of the 50th Annual Meeting of the Association for Computational Linguistics*, page 171-175, Jeju Island, Korea.

Vladimir N. Vapnik. 1998. Statistical learning theory. *Springer*. New York, USA.

Wanjui Lee, Sergey Verzakov, and Robert P. Duin. 2007. Kernel combination versus classifier combination. In *Proceedings of the 7th International Workshop on Multiple Classifier Systems*, page 22-31, Rrague, Czech Republic.

Xiaoli Li, Philip S. Yu, Bing Liu, and See Kiong Ng. 2009. Positive unlabeled learning for data stream classification. In *Proceedings of the SIAM International Conference on Data Ming*, page 257-268, Nevada, USA.

Yafeng Ren, Donghong Ji, Lan Yin, and Hongbin Zhang. 2014. Finding deceptive opinion spam by correcting the mislabled instances. *Chinese Journal of Electronics*, 23(4):702-707.

Yafeng Ren, Lan Yin, and Donghong Ji. 2014. Deceptive reviews detection based on language structure and sentiment polarity. *Journal of Frontiers of Computer Science and Technology*, 8(3):313-320.

Yanshan Xiao, Bing Liu, Jie Yin, Longbing Cao, Chengqi Zhang, and Zhifeng Hao. 2011. Similarity-based approach for positive and unlabeled learning. In *Proceeding of the 22nd International Joint Conference on Artifical Intelligence*, page 1577-1582, Barcelona, Spain.

Zoltan Gyongyi, Hector Garcia-Molina, and Jan Pedesen. 2004. Combating web spam web with trustrank. In *Proceedings of the 30th International Conference on Very Large Data Bases*, page 576-587, Toronto, Canada.

Appendix A

The optimization problem is as follows:

$$\begin{aligned} min \qquad & F(w,b,\epsilon) = \frac{1}{2}||w||^2 + C_1\sum_{i=1}^{|P|}\epsilon_i + C_2\cdot \\ & \sum_{j=1}^{|US|}p^+(x^{(j)})\epsilon_j + C_3\sum_{m=1}^{|US|}p^-(x^{(m)})\epsilon_m + \\ & C_4\sum_{n=1}^{|NS|}\epsilon_n \\ s.t. \quad & y^{(i)}(w^Tx^{(i)}+b) \geq 1-\epsilon_i, \quad x^{(i)} \in P \\ & y^{(j)}(w^Tx^{(j)}+b) \geq 1-\epsilon_j, \quad x^{(j)} \in US \\ & y^{(m)}(w^Tx^{(m)}+b) \geq 1-\epsilon_m, \quad x^{(m)} \in US \\ & y^{(n)}(w^Tx^{(n)}+b) \geq 1-\epsilon_n, \quad x^{(n)} \in NS \\ & \epsilon_i \geq 0, \quad \epsilon_j \geq 0, \quad \epsilon_m \geq 0, \quad \epsilon_n \geq 0 \end{aligned} \tag{10}$$

We construct the Lagrangian function for the above optimization problem, we have:

$$\begin{aligned} L(w,b,\epsilon,\alpha,\gamma) = & F(w,b,\epsilon) + \sum_{i=1}^{|P|}\alpha_i[-y^{(i)}\cdot \\ & (w^Tx^{(i)}+b)+1-\epsilon_i] + \sum_{j=1}^{|US|}\alpha_j[-y^{(j)}(w^Tx^{(j)}+ \\ & b)+1-\epsilon_j] + \sum_{m=1}^{|US|}\alpha_m[-y^{(m)}(w^Tx^{(m)}+b)+1 \\ & -\epsilon_m] + \sum_{n=1}^{|NS|}\alpha_n[-y^{(n)}(w^Tx^{(n)}+b)+1-\epsilon_n] - \\ & \sum_{i=1}^{|P|}\gamma_i\epsilon_i - \sum_{j=1}^{|US|}\gamma_j\epsilon_j - \sum_{m=1}^{|US|}\gamma_m\epsilon_m - \sum_{n=1}^{|NS|}\gamma_n\epsilon_n \end{aligned} \tag{11}$$

Here, the α and γ are Lagrange multipliers. To find the dual form of the problem, we need to first minimize $L(w,b,\epsilon,\alpha,\gamma)$ with respect to w and b, we will do by setting the derivatives of L with respect to w and b to zero, we have:

$$\begin{aligned} \frac{\partial L(w,b,\epsilon,\alpha,\gamma)}{\partial w} = & w - \sum_{i=1}^{|P|}\alpha_i y^{(i)}x^{(i)} - \sum_{j=1}^{|US|} \\ & \alpha_j y^{(j)}x^{(j)} - \sum_{m=1}^{|US|}\alpha_m y^{(m)}x^{(m)} - \sum_{n=1}^{|NS|}\alpha_n y^{(n)}\cdot \\ & x^{(n)} = 0 \end{aligned} \tag{12}$$

This implies that

$$\begin{aligned} w = & \sum_{i=1}^{|P|}\alpha_i y^{(i)}x^{(i)} + \sum_{j=1}^{|US|}\alpha_j y^{(j)}x^{(j)} + \sum_{m=1}^{|US|}\alpha_m\cdot \\ & y^{(m)}x^{(m)} + \sum_{n=1}^{|NS|}\alpha_n y^{(n)}x^{(n)} \end{aligned} \tag{13}$$

Here, to simplify the presentation, we redefine some notations in the following:

$$T^+ = P \cup US, T^- = US \cup NS, T^* = T^+ \cup T^-$$

$$C_i^+ = \begin{cases} C_1, & x^{(i)} \in P \\ C_2 p^+ x^{(j)}, & x^{(j)} \in US \end{cases}$$

$$C_j^- = \begin{cases} C_3 p^- x^{(m)}, & x^{(m)} \in US \\ C_4, & x^{(n)} \in NS \end{cases}$$

so we obtain

$$w = \sum_{i=1}^{|T^*|}\alpha_i y^{(i)}x^{(i)} \tag{14}$$

As for the derivative with respect to b, we obtain

$$\begin{aligned} \frac{\partial L(w,b,\epsilon,\alpha,\gamma)}{\partial b} = & -\sum_{i=1}^{|P|}\alpha_i y^{(i)} - \sum_{j=1}^{|US|}\alpha_j y^{(j)} \\ & - \sum_{m=1}^{|US|}\alpha_m y^{(m)} - \sum_{n=1}^{|NS|}\alpha_n y^{(n)} = 0 \end{aligned} \tag{15}$$

We get:

$$\sum_{i=1}^{|T^*|}\alpha_i y^{(i)} = 0 \tag{16}$$

If we take Equation (14) and (16) back into the Lagrangian function (Equation 11), and simplify, we get

$$L(w,b,\epsilon,\alpha,\gamma) = \sum_{i=1}^{|T^*|}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{|T^*|} y^{(i)}y^{(j)}\alpha_i \cdot \alpha_j < x^{(i)}, x^{(j)} > \tag{17}$$

To the primal optimization formula (10), we can obtain the following dual optimization problem:

$$\begin{aligned} max \qquad W(\alpha) &= \sum_{i=1}^{|T^*|}\alpha_i - \frac{1}{2}\sum_{i=1,j=1}^{|T^*|} \alpha_i\alpha_j y^{(i)} \cdot y^{(j)} < x^{(i)}, x^{(j)} > \\ s.t. \quad & C_i^+ \geq \alpha_i \geq 0, \quad x^{(i)} \in T^+ \\ & C_j^- \geq \alpha_j \geq 0, \quad x^{(j)} \in T^- \\ & \sum_{i=1}^{|T^+|}\alpha_i - \sum_{j=1}^{|T^-|}\alpha_j = 0 \end{aligned} \tag{18}$$

where $< x^{(i)}, x^{(j)} >$ is the inner product of $x^{(i)}$ and $x^{(j)}$, we can replace them by using kernel function $\phi(x^{(i)})$ and $\phi(x^{(j)})$, respectively.

Resolving Shell Nouns

Varada Kolhatkar
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3G4, Canada
varada@cs.toronto.edu

Graeme Hirst
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3G4, Canada
gh@cs.toronto.edu

Abstract

Shell nouns, such as *fact* and *problem*, occur frequently in all kinds of texts. These nouns themselves are unspecific, and can only be interpreted together with the *shell content*. We propose a general approach to automatically identify shell content of shell nouns. Our approach exploits lexico-syntactic knowledge derived from the linguistics literature. We evaluate the approach on a variety of shell nouns with a variety of syntactic expectations, achieving accuracies in the range of 62% (baseline = 33%) to 83% (baseline = 74%) on crowd-annotated data.

1 Introduction

Shell nouns are abstract nouns, such as *fact*, *issue*, *idea*, and *problem*, which facilitate efficiency by avoiding repetition of long stretches of text. The *shell* metaphor comes from Schmid (2000), and it captures the various functions of these nouns in a discourse: containment, signalling, pointing, and encapsulating. Shell nouns themselves are unspecific, and can only be interpreted together with their *shell content*, i.e., the propositional content they encapsulate in the given context. The process of identifying this content in the given context is referred to as *shell noun resolution* or *interpretation*. Examples (1), (2), and (3) show usages of the shell nouns *fact* and *issue*. The shell noun phrases are resolved to the postnominal *that* clause, the complement *wh* clause, and the immediately preceding clause, respectively.[1,2]

(1) **The fact that a major label hadn't been at liberty to exploit and repackage the material on CD** meant that prices on the vintage LP market were soaring.

(2) **The issue** that this country and Congress must address is **how to provide optimal care for all without limiting access for the many**.

(3) **Living expenses are much lower in rural India than in New York**, but **this fact** is not fully captured if prices are converted with currency exchange rates.

Observe that the relation between shell noun phrases and their shell content is similar to the relation of abstract anaphora (or cataphora) (Asher, 1993) with backward- or forward-looking abstract-object antecedents. For anaphoric shell noun examples, the shell content precedes the shell noun phrase, and for cataphoric shell noun examples the shell content follows the shell noun phrase.[3]

Shell nouns as a group occur frequently in argumentative texts (Schmid, 2000; Flowerdew, 2003; Botley, 2006). They play an important role in organizing a discourse and maintaining its coherence (Schmid, 2000; Flowerdew, 2003), and resolving them is an important component of various computational linguistics tasks that rely on

[1] Note that the postnominal *that*-clause in (1) is not a relative clause: the fact in question is not an argument of *exploit and repackage*.

[2] All examples in this paper are from the New York Times corpus (`https://catalog.ldc.upenn.edu/LDC2008T19`)

[3] We use the terms cataphoric shell noun and anaphoric shell noun for lack of better alternatives.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 499–510,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

discourse structure. Accordingly, identifying shell content can be helpful in summarization, information retrieval, and ESL learning (Flowerdew, 2003; Hinkel, 2004).

Despite their importance in discourse, understanding of shell nouns from a computational linguistics perspective is only in the preliminary stage. Recently, we proposed an approach to annotate and resolve anaphoric cases of six typical shell nouns: *fact*, *reason*, *issue*, *decision*, *question*, and *possibility* (Kolhatkar et al., 2013b). This work drew on the observation that shell nouns following cataphoric constructions are easy to resolve. We manually developed rules to identify shell content for such cases. Later, we used these cataphoric examples and their shell content as training data to resolve harder anaphoric examples.

In this paper, we propose a general algorithm to resolve cataphoric shell noun examples. Our long-term goal is to build an end-to-end shell-noun resolution system. If we want to go beyond the six shell nouns from our previous work, and generalize our approach to other shell nouns, first we need to develop an approach to resolve cataphoric shell noun examples. A number of challenges are associated with this seemingly easy task. The primary challenges is that this resolution is in many crucial respects a semantic phenomenon. To obtain the required semantic knowledge, we exploit the properties of shell nouns and their categorization described in the linguistics literature. We evaluate our method using crowdsourcing, and demonstrate how far one can get with simple, deterministic shell content extraction.

2 Related work

Shell-nounhood is a well-established concept in linguistics (Vendler, 1968; Ivanic, 1991; Asher, 1993; Francis, 1994; Schmid, 2000, inter alia). However, understanding of shell nouns from a computational linguistics perspective is only in the preliminary stage.

Shell nouns take a number of semantic arguments. In this respect, they are similar to the general class of argument-taking nominals as given in the NomBank (Meyers et al., 2004). Similarly, there is a small body of literature that addresses nominal semantic role labelling (Gerber et al., 2009) and nominal subcategorization frames (Preiss et al., 2007). That said, the distinguishing property of shell nouns is that one of their semantic arguments is the shell content, but the literature in computational linguistics does not provide any method that is able to identify the shell content. The focus of our work is to rectify this.

Shell content represents complex and abstract objects. So traditional linguistic and psycholinguistic principles used in pronominal anaphora resolution (see the survey by Poesio et al. (2011)), such as gender and number agreement, are not applicable in resolving shell nouns. That said, there is a line of literature on annotating and resolving personal and demonstrative pronouns, which typically refer to similar kinds of non-nominal abstract entities (Passonneau, 1989; Eckert and Strube, 2000; Byron, 2003; Müller, 2008; Hedberg et al., 2007; Poesio and Artstein, 2008; Navarretta, 2011, inter alia). Also, there have been attempts at annotating the shell content of anaphoric occurrences of shell nouns (e.g., Botley (2006), Kolhatkar et al. (2013a)). However, none of these approaches attempt to annotate and resolve cataphoric examples such (1) and (2).

3 Challenges

A number of challenges are associated with the task of resolving cataphoric shell noun examples, especially when it comes to developing a holistic approach for a variety of shell nouns.

First, each shell noun has idiosyncrasies. Different shell nouns have different semantic and syntactic expectations, and hence they take different types of one or more semantic arguments: one introducing the shell content, and others expressing circumstantial information about the shell noun. For instance, *fact* typically takes a single factual clause as an argument, which is its shell content, as we saw in example (1), whereas *reason* expects two arguments: the cause and the effect, with the content introduced in the cause, as shown in example (4).[4] Similarly, *decision* takes an agent making the decision and the shell content is represented as an action or a proposition, as shown in (5).[5]

(4) **<u>One reason</u>** [that 60 percent of New York City public-school children read below grade level]$^{\text{effect}}$ is [**that many elementary schools don't have libraries**]$^{\text{cause}}$.

[4]Observe that the postnominal *that* clause in (4) is not a relative clause, and still it is not the shell content because it is not the cause argument of the shell noun *reason*.

[5]Observe that this aspect of shell nouns of taking different numbers and kinds of complement clauses is similar to verbs having different subcategorization frames.

(5) I applaud loudly the decision of [Greenburgh]^{agent} **to ban animal performances**.

Second, the relation between a shell noun and its content is in many crucial respects a semantic phenomenon. For instance, resolving the shell noun *reason* to its shell content involves identifying a) that *reason* generally expects two semantic arguments: cause and effect, b) that the cause argument (and not the effect argument) represents the shell content, and c) that a particular constituent in the given context represents the cause argument.

Third, at the conceptual level, once we know which semantic argument represents shell content, resolving examples such as (4) seems straightforward using syntactic structure, i.e., by extracting the complement clause. But at the implementation level, this is a non-trivial problem for two reasons. The first reason is that examples containing shell nouns often follow syntactically complex constructions, including embedded clauses, coordination, and sentential complements. An automatic parser is not always accurate for such examples. So the challenge is whether the available tools in computational linguistics such as syntactic parsers and discourse parsers are able to provide us with the information that is necessary to resolve these difficult cases. The second reason is that the shell content can occur in many different constructions, such as apposition (e.g., ***parental ownership of children***, ***a concept*** *that allows* ...), postnominal and complement clause constructions, as we saw in examples (1) and (2), and modifier constructions (e.g., ***the liberal trade policy*** *that* ...). Moreover, in some constructions, the content is indefinite (e.g., ***A bad idea*** *does not harm until someone acts upon it.*) or *None* because the example is a non-shell noun usage (e.g., ***this week's issue*** *of Sports Illustrated*), and the challenge is to identify such cases.

Finally, whether the postnominal clause introduces the shell content or not is dependent on the context of the shell noun phrase. The resolution can be complicated by complex syntactic constructions. For instance, when the shell noun follows verbs such as *expect*, it becomes difficult for an automatic system to identify whether the postnominal or the complement clause is of the verb or of the shell noun (e.g., *they did not expect* ***the decision*** *to reignite tension in Crown Heights* vs. *no one expected* ***the decision*** ***to call an election***). Similarly, shell noun phrases can be objects of prepositions, and whether the postnominal clause introduces the shell content or not is dependent on this preposition. For instance, for the pattern *reason that*, the postnominal *that* clause does not generally introduce the shell content, as we saw in (4); however, this does not hold when the shell noun phrase containing *reason* follows the preposition *for*, as shown in (6).

(6) Low tax rates give people an incentive to work, for **the simple reason that they get to keep more of what they earn**.

4 Linguistic framework

Linguists have studied a variety of shell nouns, their classification, different patterns they follow, and their semantic and syntactic properties in detail (Vendler, 1968; Ivanic, 1991; Asher, 1993; Francis, 1994; Schmid, 2000, inter alia). Schmid points out that being a shell noun is a property of a specific usage of the noun rather than an inherent property of the word. He provides a list of 670 English nouns that tend to occur as shell nouns. A few frequently occurring ones are: *problem*, *notion*, *concept*, *issue*, *fact*, *belief*, *decision*, *point*, *idea*, *event*, *possibility*, *reason*, *trouble*, *question*, *plan*, *theory*, *aim*, and *principle*.

4.1 Lexico-syntactic patterns

Precisely defining the notion of shell-nounhood is tricky. A necessary property of shell nouns is that they are capable of taking clausal arguments, primarily with two lexico-syntactic constructions: *Noun + postnominal clause* and *Noun + be + complement clause* (Vendler, 1968; Biber et al., 1999; Schmid, 2000; Huddleston and Pullum, 2002). Schmid exploits these lexico-syntactic constructions to identify shell noun usages. In particular, he provides a number of typical lexico-syntactic patterns that are indicative of either anaphoric or cataphoric shell noun occurrences. Table 1 shows these patterns with examples.

Cataphoric These patterns primarily follow two constructions.

N-be-clause In this construction, the shell noun phrase occurs as the subject in a subject-verb-clause construction, with the linking verb *be*, and the shell content embedded as a *wh* clause, *that* clause, or *to*-infinitive clause. The linking

	Cataphoric	
1	*N-be-to*	**Our plan is *to*** *hire and retain the best managers we can*.
2	*N-be-that*	**The major reason is *that*** *doctors are uncomfortable with uncertainty*.
3	*N-be-wh*	Of course, **the central, and probably insoluble, issue is *whether*** *animal testing is cruel*.
4	*N-to*	**The decision *to*** *disconnect the ventilator* came after doctors found no brain activity.
5	*N-that*	Mr. Shoval left open **the possibility *that*** *Israel would move into other West Bank cities*.
6	*N-wh*	If there ever is **any doubt *whether*** *a plant is a poppy or not*, break off a stem and squeeze it.
7	*N-of*	**The concept of** *having an outsider as Prime Minister* is outdated.
	Anaphoric	
8	*th-N*	*Living expenses are much lower in rural India than in New York*, but **this fact** is not fully captured if prices are converted with currency exchange rates.
9	*th-be-N*	*People change*. **This is a fact**.
10	*Sub-be-N*	If the money is available, however, ***cutting the sales tax* is a good idea**.

Table 1: Lexico-grammatical patterns of shell nouns (Schmid, 2000). Shell noun phrases are underlined, the pattern is marked in boldface, and the shell content is marked in italics.

	Proportion							
Noun	***N-be-to***	***N-be-that***	***N-be-wh***	***N-to***	***N-that***	***N-wh***	***N-of***	**total**
idea	7	2	-	5	23	10	53	91,277
issue	-	1	5	7	14	2	71	55,088
concept	1	-	-	6	12	-	79	14,301
decision	-	-	-	80	12	1	5	55,088
plan	5	-	-	72	17	-	4	67,344
policy	4	1	-	16	25	2	51	24,025

Table 2: Distribution of cataphoric patterns for six shell nouns in the New York Times corpus. Each column shows the percentage of instances following that pattern. The last column shows the total number of cataphoric instances of each noun in the corpus.

verb *be* indicates the semantic identity between the shell noun and its content in the given context. The construction follows the patterns in rows 1, 2, and 3 of Table 1.

N-clause This construction includes the cataphoric patterns 4–7 in Table 1. For these patterns the link between the shell noun and the content is much less straightforward: whether the postnominal clause expresses the shell content or not is dependent on the shell noun and the syntactic structure under consideration. For instance, for the shell noun *fact*, the shell content is embedded in the postnominal *that* clause, as shown in (1), but this does not hold for the shell noun *reason* in example (4). The *N-of* pattern is different from other patterns: it follows the construction *N-prepositional phrase* rather than *N-clause*, and since a prepositional phrase can take different kinds of embedded constituents such as a noun phrase, a sentential complement, and a verb phrase, the pattern offers flexibility in the syntactic type of the shell content.

Anaphoric For these patterns, the link between the shell noun and the content is created using linguistic elements such as *the*, *this*, *that*, *other*, *same*, and *such*. For the patterns 8 and 9 the shell content does not typically occur in the sentence containing the shell noun phrase. For the pattern 10, the shell content is the subject in a subject-verb-N construction.

Pattern preferences Different shell nouns have different pattern preferences. Table 2 shows the distribution of cataphoric patterns for six shell nouns in the New York Times corpus. The shell nouns *idea*, *issue*, and *concept* prefer *N-of* pattern, whereas *plan* and *decision* prefer the pattern *N-to*. Among all instances of the shell noun *decision* fol-

***Idea* family**
Semantic features: [mental], [conceptual]
Frame: mental; focus on propositional content of IDEA
Nouns: ***idea***, ***issue***, ***concept***, *point*, *notion*, *theory*, ...
Patterns: *N-be-that/of*, *N-that/of*

***Plan* family**
Semantic features: [mental], [volitional], [manner]
Frame: mental; focus on IDEA
Nouns: ***decision***, ***plan***, ***policy***, *idea*, ...
Patterns: *N-be-to/that*, *N-to/that*

***Trouble* family**
Semantic features: [eventive], [attitudinal], [manner], [deontic]
Frame: general eventive
Nouns: ***problem***, ***trouble***, ***difficulty***, *dilemma*, *snag*
Patterns: *N-be-to*

***Problem* family**
Semantic features: [factual], [attitudinal], [impeding]
Frame: general factual
Nouns: ***problem***, ***trouble***, ***difficulty***, *point*, *thing*, *snag*, *dilemma* , ...
Patterns: *N-be-that/of*

***Thing* family**
Semantic features: [factual]
Frame: general factual
Nouns: ***fact***, ***phenomenon***, *point*, *case*, *thing*, *business*
Patterns: *N-that*, *N-be-that*

***Reason* family**
Semantic features: [factual], [causal]
Frame: causal; attentional focus on CAUSE
Nouns: ***reason***, *cause*, *ground*, *thing*
Patterns: *N-be-that/why*, *N-that/why*

Table 3: Example families from Schmid (2000). The nouns in boldface are used to evaluate this work.

lowing Schmid's cataphoric patterns, 80% of the instances follow the pattern *N-to*.[6]

4.2 Categorization of shell nouns

Schmid classifies shell nouns at three levels. At the most abstract level, he classifies shell nouns into six semantic classes: *factual*, *linguistic*, *mental*, *modal*, *eventive*, and *circumstantial*. Each semantic class indicates the type of experience the shell noun is intended to describe. For instance, the *mental* class describes ideas and cognitive states, whereas the *linguistic* class describes utterances, linguistic acts, and products thereof.

The next level of classification includes more-detailed semantic features. Each broad semantic class is sub-categorized into a number of *groups*. A group of an abstract class tries to capture the semantic features associated with the fine-grained differences between different usages of shell nouns in that class. For instance, groups associated with the *mental* class are: *conceptual*, *creditive*, *dubiative*, *volitional*, and *emotive*.

The third level of classification consists of *families*. A family groups together shell nouns with similar semantic features. Schmid provides 79 distinct families of 670 shell nouns. Each family is named after the primary noun in that family. Table 3 shows six families: *Idea*, *Plan*, *Trouble*, *Problem*, *Thing*, and *Reason*. A shell noun can be a member of multiple families. The nouns subsumed in a family share semantic features. For instance, all nouns in the *Idea* family are *mental* and *conceptual*. They are mental because ideas are only accessible through thoughts, and conceptual because they represent reflection or an application of a concept. Each family activates a *semantic frame*. The idea of these semantic frames is similar to that of frames in Frame semantics (Fillmore, 1985) and in semantics of grammar (Talmy, 2000). In particular, Schmid follows Talmy's conception of frames. A semantic frame describes conceptual structures, its elements, and their interrelationships. For instance, the *Reason* family invokes the causal frame, which has cause and effect as its elements with the attentional focus on the cause. According to Schmid, the nouns in a family also share a number of lexico-syntactic features. The *patterns* attribute in Table 3 shows prototypical lexico-syntactic patterns, which *attract* the members of the family. Schmid defines *attraction* as the degree to which a lexico-grammatical pattern attracts a certain noun. For instance, the patterns *N-to* and *N-that* attract the shell nouns in the *Plan* family, whereas the *N-that* pattern attracts the nouns in the *Thing* family. The pattern *N-of* is restricted to a smaller group of nouns such as *concept*, *problem*, and *issue*.[7,8]

[6] Table 2 does not include anaphoric patterns, as this paper is focused on cataphoric shell noun examples. Anaphoric patterns are common for all shell nouns: among all instances of a shell noun, approximately 50 to 80% are anaphoric.

[7] Schmid used the British section of COBUILD'S *Bank of English* for his classification.

[8] Schmid's families could help enrich resources such as FrameNet (Baker et al., 1998) with the shell content information.

5 Resolution algorithm

With this exposition, the problem of shell noun resolution is identifying the appropriate semantic argument of the shell noun representing its shell content. This section describes our algorithm to resolve shell nouns following cataphoric patterns. The algorithm addresses the primary challenge of idiosyncrasies of shell nouns by exploiting Schmid's semantic families (see Section 4.2). The input of the algorithm is a shell noun instance following a cataphoric pattern, and the output is its shell content or *None* if the shell content is not present in the given sentence. The algorithm follows three steps. First, we parse the given sentence using the Stanford parser.[9] Second, we look for the noun phrase (NP), where the head of the NP is the shell noun to be resolved.[10] Finally, we extract the appropriate shell content, if it is present in the given sentence.

5.1 Identifying potentially anaphoric shell-noun constructions

Before starting the actual resolution, first we identify whether the shell content occurs in the given sentence or not. According to Schmid, the lexico-syntactic patterns signal the position of the shell content. For instance, if the pattern is of the form *N-be-clause*, the shell content is more likely to occur in the complement clause in the same sentence. That said, although on the surface level, the shell noun seems to follow a cataphoric pattern, it is possible that the shell content is not given in a postnominal or a complement clause, as shown in (7).

(7) Just as weekend hackers flock to the golf ball most used by PGA Tour players, **recreational skiers, and a legion of youth league racers, gravitate to the skis worn by Olympic champions**. It is **the reason** that top racers are so quick flash their skis for the cameras in the finish area.

Here, the shell noun and its content are linked via the pronoun *it*. For such constructions, the shell noun phrase and shell content do not occur in the same sentence. Shell content occurs in the preceding discourse, typically in the preceding sentence.

[9] `http://nlp.stanford.edu/software/lex-parser.shtml`

[10] We extract the head of an NP following the heuristics proposed by Collins (1999, p. 238).

We identify such cases, and other cases where the shell content is not likely to occur in the postnominal or complements clauses, by looking for the patterns below in the given order, returning the shell content when it occurs in the given sentence.

Sub-be-N This pattern corresponds to the lexico-grammatical pattern in Figure 1(a). If this pattern is found, there are three main possibilities for the *subject*. First, if an existential *there* occurs at the subject position, we move to the next pattern. Second, if the subject is *it* (example (7)), *this* or *that*, we return *None*, assuming that the content is not present in the given sentence. Finally, if the first two conditions are not satisfied, i.e., if the subject is neither a pronoun not an existential *there*, we assume that subject contains a valid shell content, and return it. An example is shown in (8). Note that in such cases, unlike other patterns, the shell content is expressed as a noun phrase.

(8) **Strict liability** is **the biggest issue** when considering what athletes put in their bodies.

Apposition Another case where shell content does not typically occur in the postnominal or complement clause is the case of apposition. Indefinite shell noun phrases often occur in apposition constructions, as shown in (9).

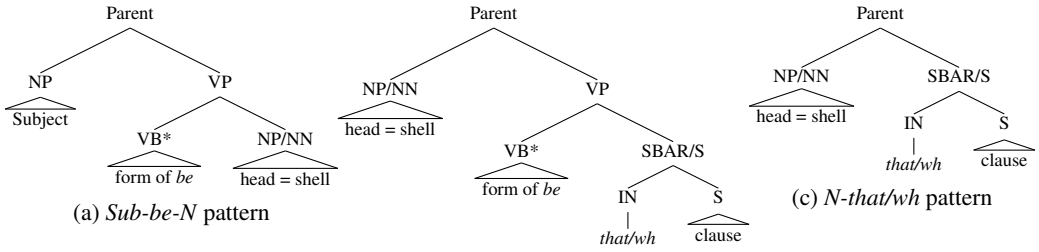
(9) The LH lineup, according to Gale, will feature **"cab-forward" design**, **a concept** that particularly pleases him.

In this step, we check for this construction and return the sentential, verbal, or nominal left sibling of the shell noun phrase.

Modifier For shell nouns such as *issue*, *phenomenon*, and *policy*, often the shell content is given in the modifier of the shell noun, as shown in (10).

(10) But in the 18th century, Leipzig's central location in German-speaking Europe and **the liberal trade policy** of the Saxon court fostered publishing.

We deal with such cases as follows. First, we extract the modifier phrases by concatenating the modifier words having noun, verb, or adjective part-of-speech tags. To exclude unlikely modifier phrases as shell content (e.g., *good idea*, *big*



(a) *Sub-be-N* pattern

(b) *N-be-clause* pattern

(c) *N-that/wh* pattern

Figure 1: Lexico-syntactic patterns for shell nouns

issue), we extract a list of modifiers for a number of shell nouns and create a stoplist of modifiers. If any of the words in the modifier phrases is a pronoun or occurs in the stoplist, we move to the next pattern. If the modifier phrase passes the stoplist test, to distinguish between non-shell content and shell content modifiers, we examine the hypernym paths of the words in the modifier phrase in WordNet (Fellbaum, 1998). If the synset *abstraction.n.06* occurs in the path, we consider the modifier phrase to be valid shell content, assuming that the shell content of shell nouns most typically represents an abstract entity.

5.2 Resolving remaining instances

At this stage we are assuming that the shell content occurs either in the postnominal clause or the complement clause. So we look for the patterns below, returning the shell content when found.

N-be-clause The lexico-grammatical pattern corresponding to the pattern *N-be-clause* is shown in Figure 1(b). This is one of the more reliable patterns for shell content extraction, as the *be* verb suggests the semantic identity between the shell noun and the complement clause. The *be*-verb does not necessarily have to immediately follow the shell noun. For instance, in example (2), the head of the NP *The issue that this country and Congress must address* is the shell noun *issue*, and hence it satisfies the construction in Figure 1(b).

N-clause Finally, we look for this pattern. An example of this pattern is shown in Figure 1(c). This is the most common (see Table 2) and trickiest pattern in terms of resolution, and whether the shell content is given in the postnominal clause or not is dependent on the properties of the shell noun under consideration and the syntactic construction of the example. For instance, for the shell noun *decision*, the postnominal *to*-infinitive clause typically represents shell content. But this did not hold for the shell noun *reason*, as shown in (11).

(11) **The reason** to resist becoming a participant is obvious.

Here, Schmid's semantic families come in the picture. We wanted to examine a) the extent to which the previous steps help in resolution, and b) whether knowledge extracted from Schmid's families add value to the resolution. So we employ two versions of this step.

Include Schmid's cues (+SC) This version exploits the knowledge encoded in Schmid's semantic families (Section 4.2), and extracts postnominal clauses only if Schmid's pattern cues are satisfied. In particular, given a shell noun, we determine the families in which it occurs and list all possible patterns of these families as shell content cues. The postnominal clause is a valid shell content only if it satisfies these cues. For instance, the shell noun *reason* occurs in only one family: *Reason*, with the allowed shell content patterns *N-that* and *N-why*. Schmid's patterns suggest that the postnominal *to*-infinitive clauses are not allowed as shell content for this shell noun, and thus this step will return *None*. This version helps correctly resolving examples such as (11) to *None*.

Exclude Schmid's cues (–SC) This version does not enforce Schmid's cues in extracting the postnominal clauses. For instance, the *Problem* family does not include *N-that/wh/to/of* patterns, but in this condition, we nonetheless allow these patterns in extracting the shell content of the nouns from this family.

6 Evaluation data

We claim that our algorithm is able to resolve a variety of shell nouns. That said, creating evaluation data for all of Schmid's 670 English shell

nouns is extremely time-consuming, and is therefore not pursued further in the current study. Instead we create a sample of representative evaluation data to examine how well the algorithm works a) on a variety of shell nouns, b) for shell nouns within a family, c) for shell nouns across families with completely different semantic and syntactic expectations, and d) for a variety of shell patterns from Table 1.

6.1 Selection of nouns

Recall that each shell noun has its idiosyncrasies. So in order to evaluate whether our algorithm is able to address these idiosyncrasies, the evaluation data must contain a variety of shell nouns with different semantic and syntactic expectations. To examine a), we consider the six families shown in Table 3. These families span three abstract categories: *mental*, *eventive*, and *factual*, and five distinct groups: *conceptual*, *volitional*, *factual*, *causal*, and *attitudinal*. Also, the families have considerably different syntactic expectations. For instance, the nouns in the *Idea* family can have their content in *that* or *of* clauses occurring in *N-clause* or *N-be-clause* constructions, whereas the *Trouble* and *Problem* families do not allow *N-clause* pattern. The shell content of the nouns in the *Plan* family is generally represented with *to*-infinitive clauses. To examine b) and c), we choose three nouns from each of the first four families from Table 3. To add diversity, we also include two shell nouns from the *Thing* family and a shell noun from the *Reason* family. So we selected a total of 12 shell nouns for evaluation: *idea*, *issue*, *concept*, *decision*, *plan*, *policy*, *problem*, *trouble*, *difficulty*, *reason*, *fact*, and *phenomenon*.

6.2 Selection of instances

Recall that the shell content varies based on the shell noun and the pattern it follows. Moreover, shell nouns have pattern preferences, as shown in Table 2. To examine d), we need shell noun examples following different patterns from Table 1. We consider the New York Times corpus as our base corpus, and from this corpus extract all sentences following the lexico-grammatical patterns in Table 1 for the twelve selected shell nouns. Then we arbitrarily pick 100 examples for each shell noun, making sure that the selection contains examples of each cataphoric pattern from Table 1. These examples consist of 70% examples of each of the seven cataphoric patterns, and the remaining 30% of the examples are picked randomly from the distribution of patterns for that shell noun.

6.3 Crowdsourcing annotation

We designed a crowdsourcing experiment to obtain the annotated data for evaluation. We parse each sentence using the Stanford parser, and extract all possible candidates, i.e., arguments of the shell noun from the parser's output. Since our examples include embedding clauses and sentential complements, the parser is often inaccurate. For instance, in example (12), the parser attaches only the first clause of the coordination (*that people were misled*) to the shell noun *fact*.

(12) **<u>The fact</u> that people were misled and information was denied**, that's the reason that you'd wind up suing.

To deal with such parsing errors, we consider the 30-best parses given by the parser. From these parses, we extract a list of eligible candidates. This list includes the arguments of the shell noun given in the appositional clauses, modifier phrases, postnominal *that*, *wh*, or *to*-infinitive clauses, complement clauses, objects of postnominal prepositions of the shell noun, and subject if the shell noun follows *subject-be-N* construction. On average, there were three candidates per instance.

After extracting the candidates, we present the annotators with the sentence, with the shell noun highlighted, and the extracted candidates. We ask the annotators to choose the option that provides the correct interpretation of the highlighted shell noun. We also provide them the option *None of the above*, and ask them to select it if the shell content is not present in the given sentence or the shell content is not listed in the list of candidates.

CrowdFlower We used CrowdFlower[11] as our crowdsourcing platform, which in turn uses various worker channels such as Amazon Mechanical Turk[12]. CrowdFlower offers a number of features. First, it provides a *quiz* mode which facilitates filtering out spammers by requiring an annotator to pass a certain number of test questions before starting the real annotation. Second, during annotation, it randomly presents test questions with known answers to the annotators to keep them on their toes. Based on annotators' responses to the test questions, each annotator is assigned a trust

[11] http://crowdflower.com/

[12] https://www.mturk.com/mturk/welcome

	≥ 5	≥ 4	≥ 3	< 3
idea	53	67	95	5
issue	44	65	95	5
concept	40	56	96	4
decision	50	72	98	2
plan	41	55	95	5
policy	42	61	94	6
problem	52	70	100	0
trouble	44	69	99	1
difficulty	45	61	96	4
reason	48	60	93	7
fact	52	68	98	2
phenomenon	39	56	95	5
all	46	63	96	4

Table 4: Annotator agreement on shell content. Each column shows the percentage of instances on which at least *n* or fewer than *n* annotators agree on a single answer.

score: an annotator performing well on the test questions gets a high trust score. Finally, CrowdFlower allows the user to select the permitted demographic areas and skills required.

Settings We asked for at least 5 annotations per instance by annotators from the English-speaking countries. The evaluation task contained a total of 1200 instances, 100 instances per shell noun. To maintain the annotation quality, we included 105 test questions, distributed among different answers. We paid 2.5 cents per instance and the annotation task was completed in less than 24 hours.

Results Table 4 shows the agreement of the crowd. In most cases, at least 3 out of 5 annotators agreed on a single answer. We took this answer as the gold standard in our evaluation, and discard the instances where fewer than three annotators agreed. The option *None of the above* was annotated for about 30% of the cases. We include these cases in the evaluation. In total we had 1,257 instances (1,152 instances where at least 3 annotators agreed + 105 test questions).

7 Evaluation results

Baseline We evaluate our algorithm against crowd-annotated data using a *lexico-syntactic clause* (LSC) baseline. Given a sentence containing a shell instance and its parse tree, this baseline extracts the postnominal or complement clause from the parse tree depending only upon the lexico-syntactic pattern of the shell noun. For instance, for the *N-that* and *N-be-to* patterns, it extracts the postnominal *that* clause and the complement *to*-infinitive clause, respectively.[13]

	Nouns	LSC	A–SC	A+SC
1	*idea*	74	82	**83**
2	*issue*	60	75	**77**
3	*concept*	51	67	**68**
4	*decision*	70	71	**73**
5	*plan*	51	**63**	62
6	*policy*	58	**70**	52
7	*problem*	66	**69**	59
8	*trouble*	63	**68**	50
9	*difficulty*	68	**75**	49
10	*reason*	43	53	**77**
11	*fact*	43	55	**68**
12	*phenomenon*	33	**62**	50
13	*all*	57	**69**	64

Table 5: Shell noun resolution results. Each column shows the percent accuracy of resolution with the respective method. Boldface is best in row.

Results Table 5 shows the evaluation results for the LSC baseline, the algorithm without Schmid's cues (A–SC), and the algorithm with Schmid's cues (A+SC). The A–SC condition in all cases and the A+SC condition in some cases outperform the LSC baseline, which proves to be rather low, especially for the shell nouns with strict syntactic expectations (e.g., *fact* and *reason*). Thus we see that our algorithm is adding value.

That said, we observe a wide range of performance for different shell nouns. On the up side, adding Schmid's cues helps resolving the shell nouns with strict syntactic expectations. The A+SC results for the shell nouns *idea*, *issue*, *concept*, *decision*, *reason*, and *fact* outperform the baseline and the A–SC results. In particular, the A+SC results for the shell nouns *fact* and *reason* are markedly better than the baseline results. These nouns have strict syntactic expectations for the shell content clauses they take: the families *Thing* and *Certainty* of the shell noun *fact* allow only a *that* clause, and the *Reason* family of the shell noun *reason* allows only *that* and *because* clauses for the shell content. These cues help in correctly resolving examples such as (11) to *None*, where the postnominal *to*-infinitive clause

[13]Note that we only extract subordinating clauses (e.g., (SBAR (IN *that*) (*clause*))) and *to*-infinitive clauses, and not relative clauses.

describes the purpose or the goal for the reason, but not the shell content itself.

On the down side, adding Schmid's cues hurts the performance of more versatile nouns, which can take a variety of clauses. Although the A–SC results for the shell nouns *plan*, *policy*, *problem*, *trouble*, *difficulty*, and *phenomenon* are well above the baseline, the A+SC results are markedly below it. That is, Schmid's cues were deleterious. Our error analysis revealed that these nouns are versatile in terms of the clauses they take as shell content, and Schmid's cues restrict these clauses to be selected as shell content. For instance, the shell noun *problem* occurs in two semantic families with *N-be-that/of* and *N-be-to* as pattern cues (Table 3), and postnominal clauses are not allowed for this noun. Although these cues help in filtering some unwanted cases, we observed a large number of cases where the shell content is given in postnominal clauses, as shown in (13).

(13) I was trying to address **the problem** of **unreliable testimony by experts** in capital cases.

Similarly, the *Plan* family does not allow the *N-of* pattern. This cue works well for the shell noun *decision* from the same family because often the postnominal *of* clause is the agent for this shell noun and not the shell content. However, it hurts the performance of the shell noun *policy*, as *N-of* is a common pattern for this shell noun (e.g., *. . . officials in Rwanda have established* ***a policy*** *of* ***refusing to protect refugees****. . . *). Other failures of the algorithm are due to parsing errors and lack of inclusion of context information.

8 Discussion and conclusion

In this paper, we proposed a general method to resolve shell nouns following cataphoric constructions. This is a first step towards end-to-end shell noun resolution. In particular, this method can be used to create training data for any given shell noun, which can later be used to resolve harder anaphoric cases of that noun using the method that we proposed earlier (Kolhatkar et al., 2013b).

The first goal of this work was to point out the difficulties associated with the resolution of cataphoric cases of shell nouns. The low resolution results of the LSC baseline demonstrate the difficulties of resolving such cases using syntax alone, suggesting the need for incorporating more linguistic knowledge in the resolution.

The second goal of this work was to examine to what extent knowledge derived from the linguistics literature helps in resolving shell nouns. We conclude that Schmid's pattern and clausal cues are useful for resolving nouns with strict syntactic expectations (e.g., *fact*, *reason*); however, these cues are defeasible: they miss a number of cases in our corpus. It is possible to improve on Schmid's cues using crowdsourcing annotation and by exploiting lexico-syntactic patterns associated with different shell nouns from a variety of corpora.

One limitation of our approach is that in our resolution framework, we do not consider the problem of ambiguity of nouns that might not be used as shell nouns. The occurrence of nouns with the lexical patterns in Table 1 does not always guarantee shell noun usage. For instance, in our data, we observed a number of instances of the noun *issue* with the publication sense (e.g., *this week's issue of Sports Illustrated*).

Our algorithm is able to deal with only a restricted number of shell noun usage constructions, but the shell content can be expressed in a variety of other constructions. A robust machine learning approach that incorporates context and deeper semantics of the sentence, along with Schmid's cues, could mitigate this limitation.

This work opens a number of new research directions. Our next planned task is clustering different shell nouns based on the kind of complements they take in different usages similar to verb clustering (Merlo and Stevenson, 2000; Schulte im Walde and Brew, 2002).

Acknowledgements

We thank the anonymous reviewers for their comments. We also thank Suzanne Stevenson, Gerald Penn, Heike Zinsmeister, Kathleen Fraser, Aida Nematzadeh, and Ryan Kiros for their feedback. This research was financially supported by the Natural Sciences and Engineering Research Council of Canada and by the University of Toronto.

References

Nicholas Asher. 1993. *Reference to Abstract Objects in Discourse*. Kluwer Academic Publishers, Dordrecht, Netherlands.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceed-*

ings of the 17th International Conference on Computational Linguistics, volume 1 of *COLING '98*, pages 86–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

Douglas Biber, Stig Johansson, Geoffrey Leech, Susan Conrad, and Edward Finegan. 1999. *Longman Grammar of Spoken and Written English*. Pearson ESL, November.

Simon Philip Botley. 2006. Indirect anaphora: Testing the limits of corpus-based linguistics. *International Journal of Corpus Linguistics*, 11(1):73–112.

Donna K. Byron. 2003. Annotation of pronouns and their antecedents: A comparison of two domains. Technical report, University of Rochester. Computer Science Department.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Miriam Eckert and Michael Strube. 2000. Dialogue acts, synchronizing units, and anaphora resolution. *Journal of Semantics*, 17:51–89.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Charles J. Fillmore. 1985. Frames and the semantics of understanding. *Quaderni di Semantica*, 6(2):222–254.

John Flowerdew. 2003. Signalling nouns in discourse. *English for Specific Purposes*, 22(4):329–346.

Gill Francis. 1994. Labelling discourse: An aspect of nominal group lexical cohesion. In M. Coulthard, editor, *Advances in written text analysis*, pages 83–101. Routledge, London.

Matthew Gerber, Joyce Chai, and Adam Meyers. 2009. The role of implicit argumentation in nominal srl. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 146–154, Boulder, Colorado, June. Association for Computational Linguistics.

Nancy Hedberg, Jeanette K. Gundel, and Ron Zacharski. 2007. Directly and indirectly anaphoric demonstrative and personal pronouns in newspaper articles. In *Proceedings of DAARC-2007 8th Discourse Anaphora and Anaphora Resolution Colloquium*, pages 31–36.

Eli Hinkel. 2004. *Teaching Academic ESL Writing: Practical Techniques in Vocabulary and Grammar (ESL and Applied Linguistics Professional)*. Lawrence Erlbaum, Mahwah, NJ, London.

Rodney D. Huddleston and Geoffrey K. Pullum. 2002. *The Cambridge Grammar of the English Language*. Cambridge University Press, April.

Roz Ivanic. 1991. Nouns in search of a context: A study of nouns with both open- and closed-system characteristics. *International Review of Applied Linguistics in Language Teaching*, 29:93–114.

Varada Kolhatkar, Heike Zinsmeister, and Graeme Hirst. 2013a. Annotating anaphoric shell nouns with their antecedents. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 112–121, Sofia, Bulgaria, August. Association for Computational Linguistics.

Varada Kolhatkar, Heike Zinsmeister, and Graeme Hirst. 2013b. Interpreting anaphoric shell nouns using antecedents of cataphoric shell nouns as training data. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 300–310, Seattle, Washington, USA, October. Association for Computational Linguistics.

Paola Merlo and Suzanne Stevenson. 2000. Automatic verb classification based on statistical distributions of argument structure. *Computational Linguistics*, 27(3):373–408.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The nombank project: An interim report. In *In Proceedings of the NAACL/HLT Workshop on Frontiers in Corpus Annotation*.

Christoph Müller. 2008. *Fully Automatic Resolution of It, This and That in Unrestricted Multi-Party Dialog*. Ph.D. thesis, Universität Tübingen.

Costanza Navarretta. 2011. Antecedent and referent types of abstract pronominal anaphora. In *Proceedings of the Workshop Beyond Semantics: Corpus-based investigations of pragmatic and discourse phenomena*, Göttingen, Germany, Feb.

Rebecca J. Passonneau. 1989. Getting at discourse referents. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 51–59, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Massimo Poesio and Ron Artstein. 2008. Anaphoric annotation in the ARRAU corpus. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May. European Language Resources Association (ELRA).

Massimo Poesio, Simone Ponzetto, and Yannick Versley. 2011. Computational models of anaphora resolution: A survey. Unpublished.

Judita Preiss, Ted Briscoe, and Anna Korhonen. 2007. A system for large-scale acquisition of verbal, nominal and adjectival subcategorization frames from corpora. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 912–919, Prague, Czech Republic, June. Association for Computational Linguistics.

Hans-Jörg Schmid. 2000. *English Abstract Nouns As Conceptual Shells: From Corpus to Cognition*. Topics in English Linguistics 34. Mouton de Gruyter, Berlin.

Sabine Schulte im Walde and Chris Brew. 2002. Inducing German semantic verb classes from purely syntactic subcategorisation information. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 223–230, Philadelphia, PA.

Leonard Talmy. 2000. The windowing of attention. In *Toward a Cognitive Semantics*, volume 1, pages 257–309. The MIT Press.

Zeno Vendler. 1968. *Adjectives and Nominalizations.* Mouton and Co., The Netherlands.

A Comparison of Selectional Preference Models for Automatic Verb Classification

Will Roberts and **Markus Egg**
Institut für Anglistik und Amerikanistik, Humboldt University
10099 Berlin, Germany
{will.roberts,markus.egg}@anglistik.hu-berlin.de

Abstract

We present a comparison of different selectional preference models and evaluate them on an automatic verb classification task in German. We find that all the models we compare are effective for verb clustering; the best-performing model uses syntactic information to induce nouns classes from unlabelled data in an unsupervised manner. A very simple model based on lexical preferences is also found to perform well.

1 Introduction

Selectional preferences (Katz and Fodor, 1963; Wilks, 1975; Resnik, 1993) are the tendency for a word to semantically select or constrain which other words may appear in a direct syntactic relation with it. Selectional preferences (SPs) have been a perennial knowledge source for NLP tasks such as word sense disambiguation (Resnik, 1997; Stevenson and Wilks, 2001; McCarthy and Carroll, 2003) and semantic role labelling (Erk, 2007); and recognising selectional violations is thought to play a role in identifying and interpreting metaphor (Wilks, 1978; Shutova et al., 2013). We focus on the SPs of verbs, since determining which arguments are typical of a given verb sheds light on the semantics of that verb.

In this study, we present the first empirical comparison of different SP models from the perspective of automatic verb classification (Schulte im Walde, 2009; Sun, 2012), the task of grouping verbs together based on shared syntactic and semantic properties.

We cluster German verbs using features capturing their *valency* or *subcategorisation*, following prior work (Schulte im Walde, 2000; Esteve Ferrer, 2004; Schulte im Walde, 2006; Sun et al., 2008; Korhonen et al., 2008; Li and Brew, 2008), and investigate the effect of adding information about verb argument preferences. SPs are represented by features capturing lexical information about the heads of arguments to the verbs; we restrict our focus here to nouns.

We operationalise a selectional preference model as a function which maps such an argument head to a concept label. We submit that the primary characteristic of such a model is its *granularity*. In our baseline condition, all nouns are mapped to the same label; this effectively captures no information about a verb's SPs (i.e., we cluster verbs using subcategorisation information only). On the other extreme, each noun is its own concept label; we term this condition *lexical preferences* (LP). Between the baseline and LP lie a spectrum of models, in which multiple concepts are distinguished, and each concept label can represent multiple nouns. Our main hypothesis is that verb clustering will work best using a model of such intermediate granularity. This follows the intuition that verbs would seem to select for classes of nouns; for instance, we suppose that *essen* 'eat' would tend to prefer as a direct object a noun from the abstract concept *Essen* ('food'). We assume that these concepts can be expressed independently of particular predicates; that is, there exist selectional preference models that will work for all verbs (and all grammatical relations). Further benefits of grouping nouns into classes include combating data sparsity, as well as deriving models which can generalise to nouns unseen in training data.

Another parameter of a selectional preference model is the methodology used to induce the conceptual classes; put another way, the success of an SP model hinges on how it represents concepts. In this paper, we investigate the choice of noun categorisation method through an empirical comparison of selectional preference models previously used in the literature.

We set out to investigate the following questions:

1. What classes of nouns are effective descriptors

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 511–522,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

of selectional preference concepts? For example, do they correspond to features such as ANIMATE?

2. What is the appropriate granularity of selectional preference concepts?

3. Which methods of classifying nouns into concepts are most effective at capturing selectional preferences for verb clustering?

This paper is structured as follows: In Section 2, we introduce our baseline method of clustering verbs using subcategorisation information and describe evaluation; Section 3 lists the models of selectional preferences that we compare in this work; Section 4 presents results and discussion; Section 5 summarises related work; and Section 6 concludes with directions for future research.

2 Automatic verb classification

Verb classifications such as VerbNet (Kipper-Schuler, 2005) allow generalisations about the syntax and semantics of verbs and have proven useful for a range of NLP tasks; however, creation of these resources is expensive and time-consuming. Automatic verb classification seeks to learn verb classes automatically from corpus data in a cheaper and faster way. This endeavour is possible due to the link between a verb's semantics and its syntactic behaviour (Levin, 1993). Recent research has found that even automatically-acquired classifications can be useful for NLP applications (Shutova et al., 2010; Guo et al., 2011). In this section, we introduce the verb classification method used by our baseline model, which clusters verbs based on subcategorisation information. Following this, Section 2.2 explains the gold standard verb clustering and cluster purity metric which we use for evaluation.

2.1 Baseline model

In this work, we take subcategorisation to mean the requirement of a verb for particular types of argument or concomitant. For example, the English verb *put* subcategorises for subject, direct object, and a prepositional phrase (PP) like *on the shelf*:

(1) *[$_{NP}$ Al] put [$_{NP}$ the book] [$_{PP}$ on the shelf].*

A *subcategorisation frame* (SCF) describes a combination of arguments required by a specific verb; a description of the set of SCFs which a verb may take is called its *subcategorisation preference*. We acquire descriptions of verbal SCF preferences on the basis of unannotated corpus data.

Our experiments use the SdeWaC corpus (Faaß and Eckart, 2013), containing 880 million words in 45 million sentences; this is a subset of deWaC (Baroni et al., 2009), a corpus of 10^9 words extracted from Web search results. SdeWaC is filtered to include only those sentences which are maximally parsable[1]. We parsed SdeWaC with the `mate-tools` dependency parser (Bohnet et al., 2013)[2], which performs joint POS and morphological tagging, as well as lemmatisation. Our subcategorisation analyses are delivered by the rule-based SCF tagger described by Roberts et al. (2014), which operates using the dependency parses and assigns each finite verb an SCF type. The SCF tags are taken from the SCF inventory proposed by Schulte im Walde (2002), which indicates combinations of nominal and verbal complement types, such as `nap:für.Acc` (transitive verb, with a PP headed by *für* 'for'). Examples of complements are `n` for nominative subject, and `a` for accusative direct object; in SCFs which include PPs (`p`), the SCF tag specifies the head of the PP and the case of the prepositional argument (`Acc` in our example indicates the accusative case of the prepositional argument). The SCF tagger undoes passivisation and analyses verbs embedded in modal and tense constructions. We record 673 SCF types in SdeWaC.

From SdeWaC, we extracted the first 3,000,000 verb instances assigned an SCF tag by the SCF tagger, where the verb lemma is one of the 168 listed in our gold standard clustering (this requires approximately 270 million words of parsed text, or 25% of SdeWaC). We refer to this as our **test set**. In this set, each verb is seen on average 17,857 times; the most common is *geben* ('give', 328,952 instances), and the least is *grinsen* ('grin', 50).

We represent verbs as vectors, where each dimension represents a different SCF type. Vector entries are initialised with SCF code counts over the test set, and each vector is then normalised to sum to 1, so that a vector represents a discrete probability distribution over the SCF inventory. We use the Jensen-Shannon divergence as a dissimilarity measure between pairs of verb vectors. The Jensen-Shannon divergence (Lin, 1991) is an information-theoretic, symmetric measure (Equation (2)) re-

[1] The filtering used a rule-based dependency parser to estimate a per-token parse error rate for each sentence, and removed those sentences with very high error rates.

[2] `https://code.google.com/p/mate-tools/`

lated to the Kullback-Leibler divergence (Equation (3)).

$$JS(p,q) = D(p||\frac{p+q}{2}) + D(q||\frac{p+q}{2}) \quad (2)$$

$$D(p||q) = \sum_i p_i \log \frac{p_i}{q_i} \quad (3)$$

With this dissimilarity measure, we use hierarchical clustering with Ward's criterion (Ward, Jr, 1963) to partition the verbs into K disjoint sets (i.e., hard clustering), where we match K to the number of classes in our gold standard (described below).

2.2 Evaluation paradigm

We evaluate the automatically induced verb clusterings against a manually-constructed gold standard, published by Schulte im Walde (2006, page 162ff.). This Levin-style classification groups 168 high- and low-frequency verbs into 43 semantic classes; examples include Aspect (e.g., *anfangen* 'begin'), Propositional Attitude (e.g., *denken* 'think'), and Weather (e.g., *regnen* 'rain'). Some of the classes are further sub-classified; for the purposes of our evaluation, we ignore the hierarchical structure of the classification and consider each class or subclass to be a separate entity. In this way, we obtain classes of fairly comparable size and sufficient semantic consistency.[3]

We evaluate a given verb clustering against the gold standard using the *pairwise F-score* (Hatzivassiloglou and McKeown, 1993). To calculate this statistic, we construct a contingency table over the $\binom{n}{2}$ pairs of verbs, the idea being that the gold standard provides binary judgements about whether two verbs should be clustered together or not. If a clustering agrees with the gold standard as to whether a pair of verbs belong together or not, this is a "correct" answer. Using the contingency table, the standard information retrieval measures of precision (P) and recall (R) can be computed; the F-score is then the harmonic mean of these: $F = 2PR/(P+R)$. The random baseline is 2.08 (calculated as the average score of 50 random partitions), and the optimal score is 95.81, calculated by evaluating the gold standard against itself. As the gold standard includes polysemous verbs, which belong to more than one cluster, the optimal score is calculated by randomly picking one of their senses; the average is then taken over 50 such trials.

The pairwise F-score is known to be somewhat nonlinear (Schulte im Walde, 2006), penalising early clustering "mistakes" more than later ones, but it has the advantage that we can easily determine statistical significance using the contingency table and McNemar's test.

We use only one clustering algorithm and one purity metric, because our prior work shows that the most important choices for verb clustering are the distance measure used, and how verbs are represented. These factors set, we expect similar performance trends from different algorithms, with predictable variation (e.g., spectral tends to outperform hierarchical clustering, which in turn outperforms k-means). Combining Ward's criterion and F-score is a trade-off at this point; the criterion is deterministic, giving reproducible results without computational complexity, but disallows estimates of density over our evaluation metric and is greedy (see discussion in Section 4.3).

3 Selectional preference models

In this section, we introduce the various SP models that we compare in this paper. In all cases, we hold the verb clustering procedure described in the previous section unchanged, with the exception that SCF tags for verbs are parameterised for selectional preferences. As an example, a verb instance observed in a simple transitive frame with a nominal subject and accusative object would receive the SCF tag `na`. Assuming that a given SP model places the subject noun in the SP concept `animate` and the object noun in the concept `concrete`, the parameterised SCF tag would be `na*subj-{animate}*obj-{concrete}`. This process captures argument co-occurrence information about verb instances, and has the effect of multiplying the SCF inventory size, making the verb vectors described in Section 2.1 both longer and sparser.

We evaluate various types of SP models: the simple lexical preferences model; three models which perform automatic unsupervised induction of noun concepts from unlabelled data; and one which uses a manually-built lexical resource. As far as we are aware, two of these, the word space and LDA models, have never been applied to verb classification before.

[3] In contrast, a top-level class like 'Transfer of Possession (Obtaining)', not only covers 25% of the gold standard, it also comprises the semantically very diverse subclasses 'Transfer of Possession (Giving)', 'Manner of Motion', and 'Emotion'.

N	Coverage of test set
100	12.08%
200	17.18%
500	26.11%
1,000	32.70%
5,000	45.31%
10,000	49.09%
50,000	55.69%
100,000	57.67%

Table 1: Fraction of verb instances in the test set parameterised by LP as a function of the number of nouns N included in the LP model.

3.1 Lexical preferences

The LP model is the simplest in our study after the baseline condition; it simply maps a noun to its own lemma. We include as a parameter of the LP model a maximum number of nouns N to admit as LP tags. In this way, the LP model parameterises SCFs using only the N most frequent nouns in SdeWaC; nouns beyond rank N are treated as if they were unseen. Table 1 indicates what fraction of the 3 million verb instances receive SCF tags specifying one or more LPs as a function of this parameter. Note that the coverage approaches an asymptote of around 60%. This is due to the fact that noun arguments are not observed for every verb instance; many verbs' arguments are pronominal or verbal and are not treated by our SP models. Setting N allows a simple way of tuning the LP model: With increasing N, the LP model should capture more data about verb instances, but after a point this benefit should be cancelled out by the increasing sparsity in the verb vectors.

3.2 Sun and Korhonen model

The SP model described in this section (SUN) was first used by Sun and Korhonen (2009) to deliver state-of-the-art verb classification performance for English; more recently, the technique was applied to successfully identify metaphor in free text (Shutova et al., 2010; Shutova et al., 2013). It uses co-occurrence counts that describe which nouns are found with which verbs in which grammatical relations; this information is used to sort the nouns into classes in a procedure almost identical to our verb clustering method described in Section 2.1.

We extract all verb instances in SdeWaC which are analysed by the SCF tagger, and count all (verb, grammatical relation, nominal argument head) triples, where the grammatical relation is subject, direct (accusative) object, indirect (dative) object, or prepositional object[4], and is listed in the verb instance's SCF tag; we undo passivisation, remove instances of auxiliary and modal verbs, and filter out those triples seen less than 10 times in the corpus.

These observations cover 60,870 noun types and 33,748,390 tokens, co-occurring with 6,705 verb types (11,426 verb-grammatical-relation types); an example is (*sprechen, obj, Wort*) ('speak' with direct object 'word', occurring 1,585 times)[5]. We represent each noun by a vector whose 11,426 dimensions are the different verb-grammatical-relation pairs; coordinates in the vector indicate the observed corpus counts. The vectors are then normalised to sum to 1, such that each represents some particular noun's discrete probability distribution over the set of verb-grammatical-relation pairs. The distance between two noun vectors is defined to be the Jensen-Shannon divergence between their probability distributions, and we partition the set of nouns into M groups using hierarchical Ward's clustering.

The SP model then maps a noun to an arbitrary label indicating which of the M disjoint sets that noun is to be found in (i.e., all nouns in the first noun class map to the concept label `concept1`); we employ the parameter M to model SP concept granularity. As with the LP model, we use the parameter N to indicate how many nouns are included in the SUN model; we search the parameter values $N = \{300, 500, 1000, 5000, 10000\}$ and $\frac{N}{M} = \{5, 10, 15, 20, 30, 50\}$.

3.3 Word space model

Word space models (WSMs, (Sahlgren, 2006; Turney and Pantel, 2010)) use word co-occurrence counts to represent the distributional semantics of a word. This strategy makes possible a clustering of nouns that does not depend on verbal dependencies in the first place.

[4]We have also experimented with adding features for each noun showing nominal modification features (e.g., (*schwarz, nmod, Haar*), 'hair' modified by 'black'), but these seem to hurt performance.

[5]Triples representing prepositional object relations are distinguished by preposition (e.g., the triple (*geben*, *prep-in*, *Auftrag*), 'give' with PP headed by 'in' with argument head 'contract', an idiomatic expression meaning 'to commission' something).

Dagan et al. (1999) address the problem of data sparseness for the automatic determination of word co-occurrence probabilities, which includes selectional preferences. They introduce the idea of estimating the probability of hitherto unseen word combinations using available information on words that are closest w.r.t. distributional word similarity. Following this idea, Erk (2007) and Padó et al. (2007) describe a memory-based SP model, using a WSM similarity measure to generalise the model to unseen data.

We build a WSM of German nouns and use it to partition nouns into disjoint sets, which we then employ as with the SUN model. We compute word co-occurrence counts across the whole SdeWaC corpus, using as features the 50,000 most common words in SdeWaC, skipping the first 50 most common words (i.e., we use words 50 through 50,050), with sentences as windows. We lemmatise the corpus and remove all punctuation; no other normalisation is performed. Co-occurrence counts between a word w_i and a feature c_j are weighted using the t-test scheme:

$$\text{ttest}(w_i, c_j) = \frac{p(w_i, c_j) - p(w_i)p(c_j)}{\sqrt{p(w_i)p(c_j)}}$$

We use a recent technique called *context selection* (Polajnar and Clark, 2014) to improve the word space model, whereby only the C most highly weighted features are kept for each word vector. We set C by optimising the correlation between the word space model's cosine similarity and a data set of human semantic relatedness judgements for 65 word pairs (Gurevych and Niederlich, 2005); at $C = 380$, we obtain Spearman $\rho = 0.813$ and Pearson $r = 0.707$ (human inter-annotator agreement for this data set is given as $r = 0.810$).

After this, we build a similarity matrix between all pairs of nouns using the cosine similarity, and then partition the set of N nouns into M disjoint classes using spectral clustering with the MNCut algorithm (Meilă and Shi, 2001). As with the SUN model, this SP model assigns labels to nouns indicating which noun class they belong to. We search the same parameter space for N and M as for the SUN model.

3.4 GermaNet

Statistical models of SPs have often used WordNet as a convenient and well-motivated inventory of concepts (e.g., Resnik (1997), Li and Abe (1998), Clark and Weir (2002)). Typically, such models make use of probabilistic treatments to determine an appropriate concept granularity separately for each predicate; we opt here for a simple model that allows more direct control over concept granularity. We take the set of concepts relevant to describing selectional preferences to be a *target set* of synsets in GermaNet (Hamp and Feldweg, 1997), and represent the target set as the set of synsets which are at some *depth* d or less in the GermaNet noun hierarchy: $\{s \mid \text{depth}(s) \leq d\}$ where $\text{depth}(s)$ counts the number of hypernym links separating s from the root of the hierarchy. We model concept granularity by varying $d = 1 \ldots 6$; at $d = 1$, the target set is of size 5, and at $d = 6$, it is of size 17,125. Nouns are attributed to concepts as follows: Given a noun belonging to a synset s, either s is in the target set, or we take s's lowest hypernym in the target set. For polysemous nouns, each synset listing a sense of the noun votes for a member of the target set; the noun observation is then spread over the target set using the votes as weights.

This procedure makes our GermaNet SP model a *soft clustering* over nouns (i.e., a noun can belong to more than one SP concept); a consequence of this is that a single verb occurrence in the corpus can contribute fractional counts to multiple SCF types.

3.5 LDA

Latent Dirichlet allocation (Blei et al., 2003) is a generative model that discovers similarities in data using latent variables; it is frequently used for topic modelling. LDA models of SPs have been proposed by Ó Séaghdha (2010) and Ritter et al. (2010); previous to this, Rooth et al. (1999) also described a latent variable model of SPs.

We implement the LDA model of selectional preferences described by Ó Séaghdha (2010). Generatively, the model produces nominal arguments to verbs as follows: For a given (verb, grammatical relation) pair (v, r), (1) Sample a noun class z from a from a multinomial distribution $\Phi_{v,r}$ with a Dirichlet prior parameterised by α; (2) Sample a noun n from a multinomial distribution Θ_z with a Dirichlet prior parameterised by β. Like Ó Séaghdha, we use an asymmetric Dirichlet prior for $\Phi_{v,r}$ (i.e., α can differ for each noun class) and a symmetric prior for Θ_z (β is the same for each Θ_z). We estimate the LDA model using the MALLET software (McCallum, 2002) using the same (verb, grammatical

relation, argument head) co-occurrence statistics used for the SUN model. We train for 1,000 iterations using the software's default parameters, allowing the LDA hyperparameters α and β to be re-estimated every 10 iterations. We build models with 50 or 100 topics as a proxy to concept granularity; models include number of nouns N of $\{500, 1000, 5000, 10000, 50000, 100000\}$.

As with the GermaNet-based model, the LDA model creates a soft clustering of nouns; the ability of a noun to have degrees of membership in multiple concepts might be a good way to model polysemy. We also experiment with a hard clustering version of the LDA model; to do this, we assign each noun n its most likely class label z using the model's estimate for $P(z|n)$.

4 Results

We experimented with applying the SP models to different combinations of grammatical relations (e.g., only subject, only object, subject+object, etc.), but generally obtained better results by parameterising SCF tags for all grammatical relations. Table 2 summarises the evaluation scores and parameter settings for the best-performing SP models, applied to verb arguments in all four grammatical relations (subject, direct, indirect and prepositional object)[6]. The table also indicates the number of SCF types constructed by each SP model (i.e., the number of dimensions of the vectors representing verbs).

All the SP models we compare help with automatic verb clustering. Using McNemar's test on the contingency tables underlying the F-scores, all models score better than the baseline at at least the $p < 0.01$ level. LDA-hard is better than the GermaNet, LDA-soft, WSM and LP models at at least the $p < 0.05$ level; SUN is better ($p \leq 0.05$) than all models except LDA-hard. All other performance differences are not statistically significant[7].

We can also demonstrate the effectiveness of the SP models with a regression analysis on the models' coverage of the test set. By varying the number of nouns N included in the SP models which use this parameter (LP, SUN, WSM, LDA), or by parameterising SCF tags with SP information only for particular combinations of grammatical relations, different numbers of the verb instances in the test data will end up with SP information in their SCF tags (this is the "coverage" statistic in Table 1); with the exception of the GermaNet model, all of the SP models we examine here show positive correlation between the number of verb instances tagged for SP information and verb clustering performance. This effect is independent of parameter settings, indicating the performance benefit conferred by the SP models is robust.

4.1 Comparison of SP models

The GermaNet model is the least successful in our study. It achieves its best performance with a depth of 5; after this, verb clustering performance drops off again. Verb clustering using the GermaNet SP model is only slightly better than the baseline condition.

Against our expectations, the hard clustering LDA models perform better than the soft clustering ones, achieving the second highest score in our evaluation; also, in contrast to the other SP models studied in this paper, LDA performs best with fewer, coarser-grained topics. We observe that the soft clustering models produce verb vectors more than an order of magnitude longer than the hard clustering models, and suggest that simple soft clustering may be causing problems with data sparsity that interfere with verb clustering. We have also observed that the topics found by LDA do not represent polysemy as we had hoped. While some of the topics discovered by the LDA models can be easily assigned labels (e.g., body parts, people, quantities, emotions, places, buildings, tools, etc.), others are less cohesive. We found that frequent words (e.g., time, person) are generated with high probability by multiple topics in ways that do not appear to reflect multiple word senses, and that the 100-topic models exhibit this property to a greater extent. For instance, *Zeit* 'time' is highly predictive of three topics in the 50-topic models, of which only the highest-weighted topic groups time expressions together; in the 100-topic models, *Zeit* is found in six topics. Again, of these six, only the topic with the highest α consists of time expressions. In the 50-topic models, we find 11 topics that we cannot assign a coherent label; in the 100-topic models, there are 38 of these mismatched topics. In our work to date, we have not found that LDA models with greater numbers of topics find more

[6] Due to space constraints, we do not present here a detailed per-model study of performance as a function of parameter settings; we feel a summary to be adequate, since the relative performances of the models reflect trends across a range of parameter settings.

[7] Using a significance criterion of $p < 0.05$.

SP model	Parameters	Granularity	F-score	Number of SCF types
SUN	10,000 nouns	1,000 noun classes	39.76	248,665
LDA (hard)	10,000 nouns	50 topics	39.10	78,409
LP	5,000 nouns		38.02	388,691
WSM	10,000 nouns	500 noun classes	36.95	149,797
LDA (soft)	10,000 nouns	50 topics	35.91	1,524,338
GermaNet	depth = 5	8,196 synsets	34.41	851,265
Baseline			33.47	673

Table 2: Evaluation of the best SP models.

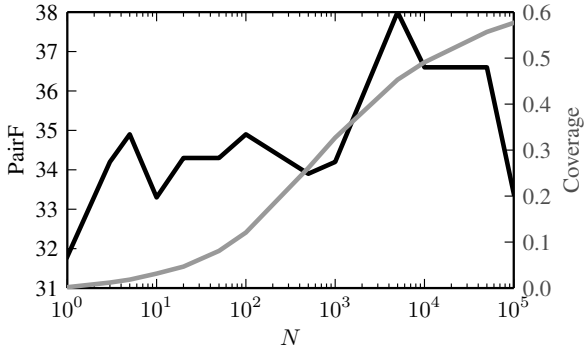


Figure 1: Verb clustering performance (black) and test set coverage (grey) of the LP model as a function of the number of nouns N included in the model.

specific concepts; it is possible that this problem might be alleviated by careful filtering of the (verb, grammatical relation, noun) triples, but we leave this question to future research.

The LP model is very effective, which is surprising given its simplicity. As expected, with increasing N, we do observe sparsity effects which hurt verb clustering performance (see Figure 1).

Our best performing model is SUN. Our best result is obtained with 10,000 nouns (the maximum value of N that we tried) in 1,000 classes, giving relatively fine-grained classes (on average 10 nouns per class). Table 3 shows some example noun classes learned by the SUN model. These include: groups with synonyms or near synonyms, often including alternate spellings of the same word (such as in the *truck* grouping); and groups of closely-related co-hyponyms, such as the body part grouping and the clothing grouping. In the latter, *bill*, *joint responsibility*, *complicity* and *inscription* are also included as things which can be *borne*, this is due to the fact that the SUN noun clustering is based on triples of verbs, grammatical relations, and nouns.

LKW (truck), *Lkw* (truck), *Lastwagen* (truck), *Castor* (container for highly radioactive material), *Laster* (truck), *Krankenwagen* (ambulance), *Transporter* (van), *Traktor* (tractor)
Hand (hand), *Kopf* (head), *Fuß* (foot), *Haar* (hair), *Bein* (leg), *Arm* (arm), *Zahn* (tooth), *Fell* (fur)
Leiche (corpse), *Leichnam* (body), *Schädel* (skull), *Skelett* (skeleton), *Wrack* (wreck), *Mumie* (mummy), *Trümmer* (debris)
Sauna (sauna), *Badezimmer* (bathroom), *Schwimmbad* (swimming pool), *Nachbildung* (replica), *Kamin* (fireplace), *Aufenthaltsraum* (common room), *Mensa* (cafeteria)
Rechnung (bill), *Kopftuch* (headscarf), *Uniform* (uniform), *Anzug* (suit), *Helm* (helmet), *Gewand* (garment), *Handschuh* (glove), *Mitverantwortung* (joint responsibility), *Bart* (beard), *Rüstung* (armour), *Mitschuld* (complicity), *Socke* (sock), *Jeans* (jeans), *Sonnenbrille* (sunglasses), *Aufschrift* (inscription), *Pullover* (sweater), *Weste* (vest), *Handschellen* (handcuffs), *Hörner* (horns), *Kennzeichen* (marking), *Tracht* (traditional costume), *Korsett* (corset), *Schuhwerk* (footwear), *Kopfbedeckung* (headgear), *Pelz* (fur), *Maulkorb* (muzzle)
Missionar (missionary), *Weihnachtsmann* (Santa Claus), *Selbstmordattentäter* (suicide bomber), *Bote* (messenger), *Nikolaus* (Nicholas), *Killer* (killer), *Bomber* (bomber), *Osterhase* (Easter bunny)

Table 3: Example noun clusters in the SUN SP model.

Furthermore, there are thematically related groups (*corpse*, *body*, etc., and *sauna*, *bathroom*, etc.). All months are placed together in one 12-word group.

Some classes can be easily subdivided into separate groups, and sometimes the source for this can be guessed: For example, sports (*football*, *golf*, *tennis*) are lumped together with musical instruments (*guitar*, *piano*, *violin*) and film roles (*starring role*, *supporting role*), these all being things that can be *played*. Many groups of personal roles (such as various kinds of government ministers) are distinguished, as are diseases and medications; other groupings contain proper names or geographical locations, sometimes of surprising specificity (e.g., authors, Biblical names, philosophers, NGOs, Eastern European countries, foreign currencies, German male first names, newspapers, television channels). The last group in Table 3 shows a grouping which appears to combine two of these semantically narrow categories, in which Santa Claus and the Easter bunny are united with killers and suicide bombers.

4.2 Noun classes as SP concepts

The WSM SP model is not as successful as SUN, but, due to the methodological similarity between these two (SP concepts modelled as hard partitions of nouns), it affords us an opportunity to investigate the question of what properties might make for an effective noun partition.

The WSM model partitions nouns based on paradigmatic information (which sentence contexts a noun appears in), rather than SUN's use of syntagmatic information (which grammatical contexts a noun appears in). Therefore, it is perhaps not surprising that the noun classes derived by the WSM are organised thematically, and the synonym/co-hyponym structure observed in the SUN noun classes is in many cases absent (e.g., {*Pferd* (horse), *Reiter* (rider), *Stall* (stable), *Sattel* (saddle), *Stute* (mare)}; these classes can easily conflate semantic roles (e.g., Agent for rider and Location for stable), which is presumably unhelpful for representing selectional preferences.

The distribution of noun classes also differs between SUN and WSM. The largest noun class in the WSM model contains 1,076 high-frequency nouns which are semantically unrelated (*day, question, case, part, reason, kind, form, week, person, month, ...*). We suppose that these nouns are thematically "neutral" and are classed together by virtue of their usage in a wide variety of sentences. This one noun class by itself subsumes 13.6% of all noun tokens in SdeWaC. WSM also includes 56 singleton noun classes; the variance in noun class size is 2800. For comparison, in SUN, the largest noun class has 73 words, and the smallest, 2 (there are 12 of these two-word classes); noun class size variance is 37. The 73-word class in SUN does indeed appear to be a grab bag (including *gas, taboo, pioneer, mustard, spy, mafia*, and *skinhead*), but these are uncommon words and account for only 0.1% of noun tokens in SdeWaC. The next two most common classes (with some 40 nouns each) are lists of names (politicians' surnames, and male first names). The noun class in the SUN model containing the largest number of high-frequency nouns (28 nouns: *human, child, woman, man, people, Mr., mother, father, ...*) only covers 3.6% of noun usages in SdeWaC and is both semantically cohesive and intuitively useful as a SP concept.

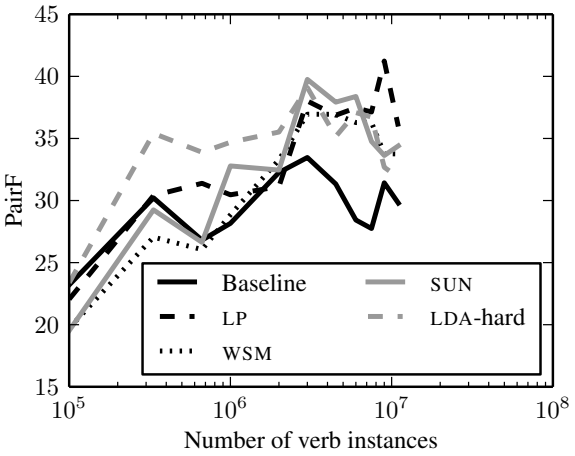


Figure 2: Verb clustering performance of SP models as a function of number of verb instances.

These issues raise the question of why the WSM model is effective at all for verb classification. We think that the larger less-related noun classes neither help nor hurt verb clustering, and we find that some of the thematic classes represent abstractions that should be useful for describing SPs. Examples include lists of body parts, countries (separate classes for Europe, Africa, Asia, etc.), diseases, human names, articles of clothing, and the group {*fruit, apple, banana, pear, strawberry*}.

4.3 Effects of test set size

We were curious if the success of the LP model might be due to the size of the test set preventing

sparsity from becoming a problem. To pursue this question, we take the four best performing SP models and run the verb clustering evaluation with the number of verb instances in the test set varying between 10,000 and the full SdeWaC corpus (11 million). The results are displayed in Figure 2. This graph indicates that below 3×10^5 verb instances, sparsity seems to become a problem for all models on this task, and the baseline delivers the best performance. Above this threshold, it seems that sparsity is not a major issue: LP performs fairly consistently, and is competitive with the SUN model. We attribute this to our use of the Jensen-Shannon divergence as a verb dissimilarity measure, which seems relatively robust to data sparsity. The LDA-hard model with its fewer topics seems to do quite well with fewer data; as the test set size increases, it drops off in the rankings. At the maximum number of verb instances, the best-performing models are SUN, WSM and the lexical preferences. The figure also shows that our evaluation metric is not smooth (note, e.g., the fluctuations in the baseline score). We believe that this reflects a degree of instability in the Ward's hierarchical clustering algorithm; this clustering method is greedy, and clustering errors can be expected to propagate, which might explain the jaggedness of the plot.

4.4 Conclusions

To conclude, we summarise the results of our analysis, using the questions formulated in the Introduction as guidelines.

First, we wanted to compare the efficiency of different classes of nouns as descriptors of selectional preference concepts. Our findings suggest that noun classes are most effective when they are semantically highly consistent, representing groups of strongly related nouns. It seems reasonable that SP concepts representing collections of synonyms would be useful for generalising observations, and should represent arguments better than simple LP. A classification of proper names (e.g., as human, corporation, country, medication) is also useful. This implies that we can expect features such as ANIMATE to be shared by all members of a noun cluster.

Second, we were interested in the appropriate granularity of selectional preference concepts. In our evaluation, we have observed a tendency for smaller, more specific noun classes to be superior; this holds because data sparsity is not a problem in our experiment. Beyond this finding, we would have liked to present a direct juxtaposition of different models on "granularity" but this is difficult: We have not yet identified a strong abstraction of granularity from the proxies we use (e.g., GermaNet depth, or SUN's N/M).

Finally, which methods of classifying nouns into concepts are most effective at capturing selectional preferences for verb clustering? In our experiments, the SUN and LDA-hard models proved to be more effective than lexical preferences, supporting our primary hypothesis that some level of SP concept granularity above the lexical level is desirable for verb clustering. On the other hand, the LP model is only slightly worse than SUN and LDA-hard, making it attractive because it is so simple. As we have shown, the potential data sparsity issues with LP can be alleviated by judiciously choosing the value of the N parameter that controls the number of nouns included in the model. In addition, comparing the SUN and WSM models, and observing the performance of the LDA-hard method, we conclude that inducing noun classes using syntagmatic information is more effective than using paradigmatic relations.

5 Related work

In this study, we have looked at the utility of selectional preferences for automatic verb classification. Some previous research has followed this line of inquiry, though prior studies have not compared alternative methods of modelling SPs. Schulte im Walde (2006) presented a detailed examination of parameters for k-means-based verb clustering in German, using the same gold standard that we employ here. She reports on the effects of adding SP information to a SCF-based verb clustering using 15 high-level GermaNet synsets as SP concepts; SP information for some combinations of grammatical relations improves clustering performance slightly, but neither are the effects consistent, nor is the improvement delivered by the SP model over the SCF-based baseline statistically significant. Schulte im Walde et al. (2008) used expectation maximisation to induce latent verb clusters from the British National Corpus while simultaneously building a tree cut model of SPs on the WordNet hierarchy using a minimum description length method; their evaluation focuses on the induced soft verb clusters, reporting the model's estimated perplexity of (verb, grammatical relation, argument head) triples. The

SPs are described qualitatively by presenting two example cases. Sun and Korhonen (2009) study the effect of adding selectional preferences to a subcategorisation-based verb clustering in English using the SUN model (see Section 3.2). They demonstrate that adding SPs to the SCF preference data leads to the best results on their two clustering evaluations; overall, their best results come from using SP information only for the subject grammatical relation. They employ coarse SP concepts (20 or 30 noun clusters) which capture general semantic categories (*Human, Building, Idea*, etc.).

Selectional preferences are usually evaluated either from a word sense disambiguation standpoint using pseudo-words (Chambers and Jurafsky, 2010), or in terms of how acceptable an argument is with a verb, via regression against human plausibility judgements. Several studies have compared SP methodologies from the latter perspective. These include Brockmann and Lapata (2003), who compared three GermaNet-based models of SP, showing that different models were most effective for describing different grammatical relations; Ó Séaghdha (2010), who compared different LDA-based models of SP, showing these to be effective for a variety of grammatical relations; and Ó Séaghdha and Korhonen (2012), who show that WordNet tree cut models, LDA, and a hybrid LDA-WordNet model are effective for describing verb-object relations.

6 Future work

Our GermaNet model delivered disappointing performance in this study; we would be interested in seeing whether a more sophisticated implementation such as the tree cut model of Li and Abe (1998) would be more competitive. We also would like to explore alternative noun clustering methods such as CBC (Pantel and Lin, 2002) and Brown clusters (Brown et al., 1992), which were not covered in this work; these would fit easily into our SP evaluation paradigm. More challenging would be a verb classification-based evaluation of the SP models of (Rooth et al., 1999) and (Schulte im Walde et al., 2008), which use expectation maximisation to simultaneously cluster verbs into verb classes and nominal arguments into noun classes; these approaches are not compatible with the evaluation framework we have used here. Finally, the SP model of Bergsma et al. (2008) has also achieved impressive results on a number of tasks, but has not been investigated for use in verb classification.

Our verb clustering evaluation in this work has matched K, the number of clusters found by Ward's method, to the number of classes in the gold standard. Since the number of clusters has an influence on the quality of the ensuing semantic classification (Schulte im Walde, 2006, page 180f.), we will also be running our experiments with different settings of K to explore whether this also influences the overall results of our evaluation.

References

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: A collection of very large linguistically processed Web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Shane Bergsma, Dekang Lin, and Randy Goebel. 2008. Discriminative learning of selectional preference from unlabeled text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 59–68.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajič. 2013. Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, 1:415–428.

Carsten Brockmann and Mirella Lapata. 2003. Evaluating and combining approaches to selectional preference acquisition. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics*, pages 27–34.

Peter F. Brown, Peter V. Desouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Nathanael Chambers and Daniel Jurafsky. 2010. Improving the use of pseudo-words for evaluating selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 445–453.

Stephen Clark and David Weir. 2002. Class-based probability estimation using a semantic hierarchy. *Computational Linguistics*, 28(2):187–206.

Ido Dagan, Lillian Lee, and Fernando C.N. Pereira. 1999. Similarity-based models of word cooccurrence probabilities. *Machine Learning*, 34(1–3):43–69.

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 216–223.

Eva Esteve Ferrer. 2004. Towards a semantic classification of Spanish verbs based on subcategorisation information. In *Proceedings of the Student Research Workshop at the Annual Meeting of the Association for Computational Linguistics*, pages 37–42.

Gertrud Faaß and Kerstin Eckart. 2013. SdeWaC - A corpus of parsable sentences from the Web. In *Language processing and knowledge in the Web*, pages 61–68. Springer, Berlin, Heidelberg.

Yufan Guo, Anna Korhonen, and Thierry Poibeau. 2011. A weakly-supervised approach to argumentative zoning of scientific documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 273–283.

Iryna Gurevych and Hendrik Niederlich. 2005. Computing semantic relatedness in German with revised information content metrics. In *Proceedings of "OntoLex 2005 - Ontologies and Lexical Resources" IJCNLP'05 Workshop*, pages 28–33.

Birgit Hamp and Helmut Feldweg. 1997. GermaNet: A lexical-semantic net for German. In *Proceedings of ACL Workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1993. Towards the automatic identification of adjectival scales: Clustering adjectives according to meaning. In *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics*, pages 172–182.

Jerrold J. Katz and Jerry A. Fodor. 1963. The structure of a semantic theory. *Language*, 39:170–210.

Karin Kipper-Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylvania.

Anna Korhonen, Yuval Krymolowski, and Nigel Collier. 2008. The choice of features for classification of verbs in biomedical texts. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 449–456.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago Press, Chicago.

Hang Li and Naoki Abe. 1998. Generalizing case frames using a thesaurus and the MDL principle. *Computational Linguistics*, 24(2):217–244.

Jianguo Li and Chris Brew. 2008. Which are the best features for automatic verb classification. In *Proceedings of ACL-08: HLT*, pages 434–442.

Jianhua Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151.

Andrew McCallum. 2002. MALLET: A machine learning for language toolkit.

Diana McCarthy and John Carroll. 2003. Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654.

Marina Meilă and Jianbo Shi. 2001. A random walks view of spectral segmentation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Diarmuid Ó Séaghdha and Anna Korhonen. 2012. Modelling selectional preferences in a lexical hierarchy. In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics*, pages 170–179.

Diarmuid Ó Séaghdha. 2010. Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 435–444.

Sebastian Padó, Ulrike Padó, and Katrin Erk. 2007. Flexible, corpus-based modelling of human plausibility judgements. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 400–409.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 613–619.

Tamara Polajnar and Stephen Clark. 2014. Improving distributional semantic vectors through context selection and normalisation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 230–238.

Philip Resnik. 1993. *Selection and information: A class-based approach to lexical relationships*. Ph.D. thesis, University of Pennsylvania.

Philip Resnik. 1997. Selectional preference and sense disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How*, pages 52–57.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A latent Dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434.

Will Roberts, Markus Egg, and Valia Kordoni. 2014. Subcategorisation acquisition from raw text for a free word-order language. In *Proceedings of the*

14th Conference of the European Chapter of the Association for Computational Linguistics, pages 298–307.

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via EM-based clustering. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 104–111.

Magnus Sahlgren. 2006. *The word-space model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Ph.D. thesis, Stockholm University.

Sabine Schulte im Walde, Christian Hying, Christian Scheible, and Helmut Schmid. 2008. Combining EM training and the MDL principle for an automatic verb classification incorporating selectional preferences. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 496–504.

Sabine Schulte im Walde. 2000. Clustering verbs semantically according to their alternation behaviour. In *Proceedings of the 18th Conference on Computational Linguistics*, pages 747–753.

Sabine Schulte im Walde. 2002. A subcategorisation lexicon for German verbs induced from a lexicalised PCFG. In *Proceedings of the 3rd Conference on Language Resources and Evaluation (LREC)*, pages 1351–1357.

Sabine Schulte im Walde. 2006. Experiments on the automatic induction of German semantic verb classes. *Computational Linguistics*, 32(2):159–194.

Sabine Schulte im Walde. 2009. The induction of verb frames and verb classes from corpora. In Anke Lüdeling and Merja Kytö, editors, *Corpus linguistics: An international handbook*, volume 2, chapter 44, pages 952–971. Mouton de Gruyter, Berlin.

Ekaterina Shutova, Lin Sun, and Anna Korhonen. 2010. Metaphor identification using verb and noun clustering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1002–1010.

Ekaterina Shutova, Simone Teufel, and Anna Korhonen. 2013. Statistical metaphor processing. *Computational Linguistics*, 39(2):301–353.

Mark Stevenson and Yorick Wilks. 2001. The interaction of knowledge sources in word sense disambiguation. *Computational Linguistics*, 27(3):321–349.

Lin Sun and Anna Korhonen. 2009. Improving verb clustering with automatically acquired selectional preferences. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 638–647.

Lin Sun, Anna Korhonen, and Yuval Krymolowski. 2008. Verb class discovery from rich syntactic data. In *Proceedings of the Ninth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 16–27.

Lin Sun. 2012. *Automatic induction of verb classes using clustering*. Ph.D. thesis, University of Cambridge, Cambridge.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

Joe H. Ward, Jr. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244.

Yorick Wilks. 1975. An intelligent analyzer and understander of English. *Communications of the ACM*, 18(5):264–274.

Yorick Wilks. 1978. Making preferences more active. *Artificial Intelligence*, 11(3):197–223.

Learning to Solve Arithmetic Word Problems with Verb Categorization

Mohammad Javad Hosseini[1], **Hannaneh Hajishirzi**[1], **Oren Etzioni**[2], **and Nate Kushman**[3]

[1]{hosseini, hannaneh}@washington.edu, [2]OrenE@allenai.org, [3]nkushman@csail.mit.edu
[1]University of Washington, [2]Allen Institute for AI, [3]Massachusetts Institute of Technology

Abstract

This paper presents a novel approach to learning to solve simple arithmetic word problems. Our system, ARIS, analyzes each of the sentences in the problem statement to identify the relevant variables and their values. ARIS then maps this information into an equation that represents the problem, and enables its (trivial) solution as shown in Figure 1. The paper analyzes the arithmetic-word problems "genre", identifying seven categories of verbs used in such problems. ARIS learns to categorize verbs with 81.2% accuracy, and is able to solve 77.7% of the problems in a corpus of standard primary school test questions. We report the first learning results on this task without reliance on predefined templates and make our data publicly available.[1]

1 Introduction

Designing algorithms to automatically solve math and science problems is a long-standing AI challenge (Feigenbaum and Feldman, 1963). For NLP, mathematical word problems are particularly attractive because the text is concise and relatively straightforward, while the semantics reduces to simple equations.

Arithmetic word problems begin by describing a partial world state, followed by simple updates or elaborations and end with a quantitative question. For a child, the language understanding part is trivial, but the reasoning may be challenging; for our system, the opposite is true. ARIS needs to make sense of multiple sentences, as shown in Figure 2, without *a priori* restrictions on the syntax or vocabulary used to describe the problem. Figure 1 shows an example where ARIS is asked to infer how many kittens Joan received based on facts and constraints expressed in the text, and represented by the state diagram and corresponding equation. While the equation is trivial, the text could have involved assembling toy aircraft, collecting coins, eating cookies, or just about any activity involving changes in the quantities of discrete objects.

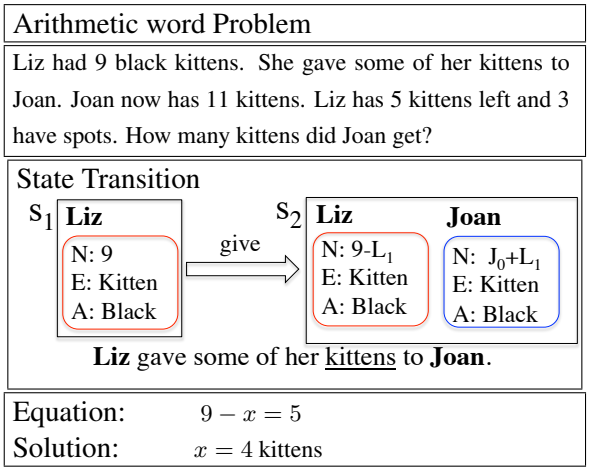


Figure 1: Example problem and solution.

This paper investigates the task of learning to solve such problems by mapping the verbs in the problem text into categories that describe their impact on the world state. While the verbs category is crucial (e.g., what happens if "give" is replaced by "receive" in Figure 1?), some elements of the problem are irrelevant. For instance, the fact that three kittens have spots is immaterial to the solution. Thus, ARIS has to determine what information is relevant to solving the problem.

To abstract from the problem text, ARIS maps the text to a state representation which consists of

[1]Our data is available at `https://www.cs.washington.edu/nlp/arithmetic`.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 523–533,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

a set of entities, their containers, attributes, quantities, and relations. A problem text is split into fragments where each fragment corresponds to an observation or an update of the quantity of an entity in one or two containers. For example in Figure 1, the sentence "Liz has 5 kittens left and 3 have spots" has two fragments of "Liz has 5 kittens left" and "3 have spots".

The verb in each sentence is associated with one or two containers, and ARIS has to classify each verb in a sentence into one of seven categories that describe the impact of the verb on the containers (Table 1). ARIS learns this classifier based on training data as described in section 4.2.

To evaluate ARIS, we compiled a corpus of about 400 arithmetic (addition and subtraction) word problems and utilized cross validation to both train ARIS and evaluate its performance over this corpus. We compare its performance to the template-based learning method developed independently and concurrently by Kushman et al. (2014). We find that our approach is much more robust to domain diversity between the training and test sets.

Our contributions are three-fold: (a) We present ARIS, a novel, fully automated method that learns to solve arithmetic word problems; (b) We introduce a method to automatically categorize verbs for sentences from simple, easy-to-obtain training data; our results refine verb senses in WordNet (Miller, 1995) for arithmetic word problems; (c) We introduce a corpus of arithmetic word problems, and report on a series of experiments showing high efficacy in solving addition and subtraction problems based on verb categorization.

2 Related Work

Understanding semantics of a natural language text has been the focus of many researchers in natural language processing (NLP). Recent work focus on learning to align text with meaning representations in specific, controlled domains. A few methods (Zettlemoyer and Collins, 2005; Ge and Mooney, 2006) use an expensive supervision in the form of manually annotated formal representations for every sentence in the training data. More recent work (Eisenstein et al., 2009; Kate and Mooney, 2007; Goldwasser and Roth, 2011; Poon and Domingos, 2009; Goldwasser et al., 2011; Kushman and Barzilay, 2013) reduce the amount of required supervision in mapping sentences to meaning representations while taking advantage of special properties of the domains. Our method, on the other hand, requires small, easy-to-obtain training data in the form of verb categories that are shared among many different problem types.

Our work is also closely related to the grounded language acquisition research (Snyder and Barzilay, 2007; Branavan et al., 2009; Branavan et al., 2012; Vogel and Jurafsky, 2010; Chen et al., 2010; Hajishirzi et al., 2011; Chambers and Jurafsky, 2009; Liang et al., 2009; Bordes et al., 2010) where the goal is to align a text into underlying entities and events of an environment. These methods interact with an environment to obtain supervision from the real events and entities in the environment. Our method, on the other hand, grounds the problem into world state transitions by learning to predict verb categories in sentences. In addition, our method combines the representations of individual sentences into a coherent whole to form the equations. This is in contrast with the previous work that study each sentence in isolation from the other sentences.

Previous work on studying math word and logic problems uses manually aligned meaning representations or domain knowledge where the semantics for all the words is provided (Lev, 2007; Lev et al., 2004). Most recently, Kushman et al. (2014) introduced an algorithm that learns to align algebra problems to equations through the use of templates. This method applies to broad range of math problems, including multiplication, division, and simultaneous equations, while ARIS only handles arithmetic problems (addition and subtraction). However, our empirical results show that for the problems it handles, ARIS is much more robust to diversity in the problem types between the training and test data.

3 Arithmetic Problem Representation

We address solving arithmetic word problems that include addition and subtraction. A problem text is split into fragments where each fragment is represented as a transition between two world states in which the quantities of entities are updated or observed (Figure 2). We refer to these fragments as sentences. We represent the world state as a tuple $\langle E, C, R \rangle$ consisting of entities E, containers C, and relations R among entities, containers, attributes, and quantities.

Entities: An *entity* is a mention in the text corre-

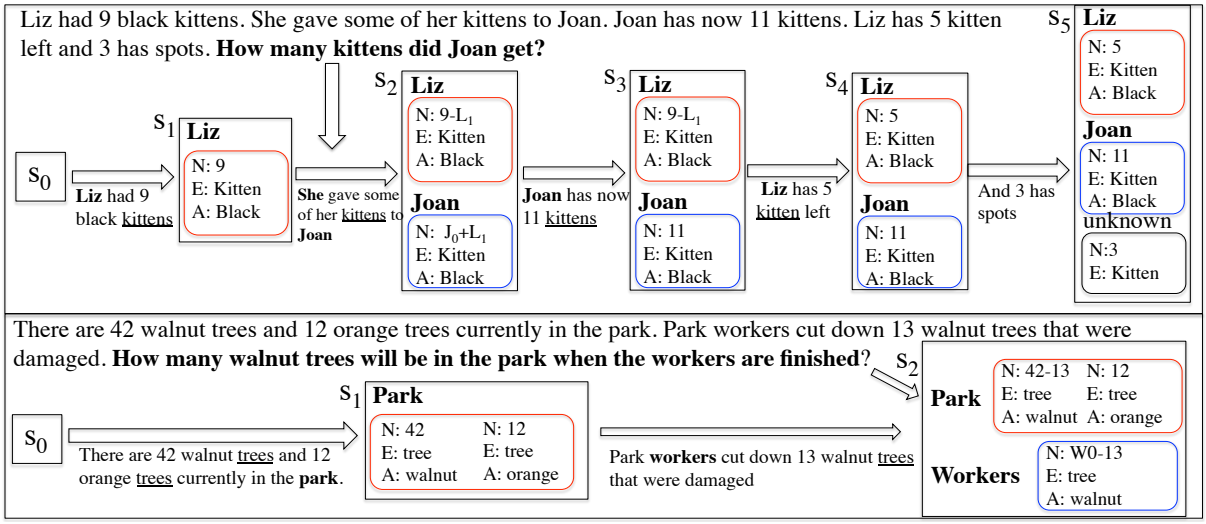


Figure 2: A figure sketching different steps of our method — a sequence of states.

sponding to an object whose quantity is observed or is changing throughout the problem. For instance, `kitten` and `tree` are entities in Figure 2. In addition, every entity has *attributes* that modify the entity. For instance, `black` is an attribute of `kittens`, and `walnut` is an attribute of `tree` (more details on attributes in section 4.1). Relations describing attributes are invariant to the state changes. For instance kittens stay `black` throughout the problem of Figure 1.

Containers: A *container* is a mention in the text representing a set of entities. For instance, `Liz`, `Joan`, `park`, and `workers` are containers in Figure 2. Containers usually correspond to the *person* possessing entities or a *location* containing entities. For example, in the sentence "There are 43 blue marbles in the basket. John found 32 marbles.", `basket` and `John` are containers of `marbles`.

Quantities: Containers include entities with their corresponding quantities in a particular world state. Quantities can be known numbers (e.g. 9), unknown variables (e.g. L_1), or numerical expressions over unknown quantities and numbers (e.g. $9-L_1$). For instance, in state 2 of Figure 2, the numerical expression corresponding to `Liz` is $9-L_1$ and corresponding to `Joan` is $J_0 + L_1$, where J_0 is a variable representing the number of `kittens` that `Joan` has started with.

Hereinafter, we will refer to a generic entity as e, container as c, number as num, attribute as a. We represent the relation between a container, an entity, and a number in the form of a quantity expression `N(c,e)`. Figure 2 shows the quantity relations in different world states.

Category	Example
Observation	There **were** 28 bales of hay in the *barn*.
Positive	*Joan* **went** to 4 football games this year.
Negative	*John* **lost** 3 of the violet balloons.
Positive Transfer	*Mike's dad* **borrowed** 7 nickels from *Mike*.
Negative Transfer	*Jason* **placed** 131 erasers in the *drawer*.
Construct	*Karen* **added** 1/4 of a cup of walnuts to a *batch* of trail mix.
Destroy	The *rabbits* **ate** 4 of *Dan's* potatoes.

Table 1: Examples for different verb categories in sentences. Entities are underlined; *containers* are italic, and **verbs** are bolded.

State transitions: Sentences depict progression of the world state (Figure 2) in the form of observations of updates of quantities. We assume that every sentence w consists of a verb v, an entity e, a quantity num (might be unknown), one or two containers c_1, c_2, and attributes a. The presence of the second container, c_2, will be dictated by the category of the verb, as we discuss below. Sentences abstract transitions $(s_t \rightarrow s_{t+1})$ between states in the form of an algebraic operation of addition or subtraction. For every sentence, we model the state transition according to the verb category and containers in the sentence. There are three verb categories for sentences with one container: *Observation*: the quantity is initialized in the container, *Positive*: the quantity is increased in the container, and *Negative*: the quantity is decreased in the container. Moreover, there are four categories for sentences with two containers: *Pos-*

itive transfer: the quantity is transferred from the second container to the first one, *Negative transfer*: the quantity is transferred from the first container to the second one, *Construct*: the quantity is increased for both containers, and *Destroy*: the quantity is decreased for both containers.

Figure 2 shows how the state transitions are determined by the verb categories. The sentence "Liz has 9 black kittens" initializes the quantity of `kittens` in the container `Liz` to `9`. In addition, the sentence "She gave some of her kittens to Joan." shows the negative transfer of L_1 kittens from `Liz` to `Joan` represented as `N(Liz,kitten)=9-`L_1 and `N(Joan,kitten)=`$J_0 + L_1$.

Given a math word problem, ARIS grounds the world state into entities (e.g., `kitten`), containers (e.g., `Liz`), attributes (e.g., `black`), and quantities (e.g., `9`) (Section 4.1). In addition, ARIS learns state transitions by classifying verb categories in sentences (Section 4.2). Finally, from the world state and transitions, it generates an arithmetic equation which can be solved to generate the numeric answer to the word problem.

4 Our Method

In this section we describe how ARIS maps an arithmetic word problem into an equation (Figure 2). ARIS consists of three main steps (Figure 3): (1) grounding the problem into entities and containers, (2) training a model to classify verb categories in sentences, and (3) solving the problem by updating the world states with the learned verb categories and forming equations.

4.1 Grounding into Entities and Containers

ARIS automatically identifies entities, attributes, containers, and quantities corresponding to every sentence fragment (details in Figure 3 step 1). For every problem, this module returns a sequence of sentence fragments $\langle w_1, \ldots, w_T, w_x \rangle$ where every w_t consists of a verb v_t, an entity e_t, its quantity num_t, its attributes a_t, and up to two containers c_{t_1}, c_{t_2}. w_x corresponds to the question sentence inquiring about an unknown entity. ARIS applies the Stanford dependency parser, named entity recognizer and coreference resolution system to the problem text (de Marneffe et al., 2006; Finkel et al., 2005; Raghunathan et al., 2010). It uses the predicted coreference relationships to replace pronouns (including possessive pronouns) with their coreferenent links. The named entity recognition output is used to identify numbers and people.

Entities: Entities are references to some object whose quantity is observed or changing throughout the problem. So to determine the set of entities, we define h as the set of noun types which have a dependent number (in the dependency parse) somewhere in the problem text. The set of entities is then defined as all noun phrases which are headed by a noun type in h. For instance `kitten` in the first sentence of Figure 1 is an entity because it is modified by the number `9`, while `kitten` in the second sentence of Figure 1 is an entity because `kitten` was modified by a number in the first sentence. Every number in the text is associated with one entity. Numbers which are dependents of a noun are associated with its entity. Bare numbers (not dependent on a noun) are associated with the previous entity in the text. The entity in the last sentence is identified as the question entity e_x . Finally, ARIS splits the problem text into $T + 1$ sentence fragments $\langle w_1, \ldots w_T, w_x \rangle$ such that each fragment contains a single entity and it's containers. For simplicity we refer to these fragments as a sentences.

Containers: Each entity is associated with one or two container noun phrases using the algorithm described in in Figure 3 step 1c. As we saw earlier with numbers, arithmetic problems often include sentences with missing information. For example in Figure 2, the second container in the the sentence "Park workers had to cut down 13 walnut trees that were damaged." is not explicitly mentioned. To handle this missing information, we use the *circumscription* assumption (McCarthy, 1980). The circumscription assumption formalizes the commonsense assumption that things are as expected unless otherwise specified. In this setting, we assume that the set of containers are fixed in a problem. Thus if the container(s) for a given entity cannot be identified they are set to the container(s) for the previous entity with the same head word. For example in Figure 2 we know from the previous sentence that `trees` were in the `park`. Therefore, we assume that the unmentioned container is the `park`.

Attributes: ARIS selects attributes A as modifiers for every entity from the dependency parser (details in Figure 3 step 1a). For example `black` is an attribute of the entity `kitten` and is an adjective modifier in the parser. These attributes are

1. **Grounding into entities and containers: for every problem p in dataset** (Section 4.1)
   (a) $\langle e_1, \ldots, e_T, e_x \rangle_p \leftarrow$ extract all entities and the question entity
      i. Extract all numbers and noun phrases (NP).
      ii. $h \leftarrow$ all noun types which appear with a number as a dependant (in the dependency parse tree) somewhere in the problem text.
      iii. $e_t \leftarrow$ all NPs which are headed by a noun type in h.
      iv. $num_t \leftarrow$ the dependant number of e_t if one exists. Bare numbers (not directly dependant on any noun phrase) are associated with the previous entity in the text. All other num_t are set to unknown.
      v. $e_x \leftarrow$ the last identified entity.
      vi. $a_t \leftarrow$ adjective and noun modifiers of e_t. Update implicit attributes using the previously observed attributes.
      vii. $v_t \leftarrow$ the verb with the shortest path to e_t in the dependency parse tree.
   (b) $\langle w_1, \ldots, w_T, w_x \rangle_p \leftarrow$ split the problem text into fragments based on the entities and verbs
   (c) $\langle c_{t_1}, c_{t_2}, \ldots, c_{T_1}, c_{T_2}, c_x \rangle_p \leftarrow$ the list of containers for each entity
      i. $c_{t_1} \leftarrow$ the subject of w_t.
      If w_t contains *There is/are*, c_{t_1} is the first adverb of place to the verb.
      ii. $c_{t_2} \leftarrow$ An NP that is direct object of the verb. If not found, c_{t_2} is the object of the first adverbial phrase of the verb.
      iii. Circumscription assumption: When c_{t_1} or c_{t_2} are not found, they are set to the previous containers.

2. **Training for sentence categorization** (Section 4.2)
   (a) $instances_1, instances_2 \leftarrow \emptyset$
   (b) for every sentence $w_t \in \langle w_1, \ldots, w_T, w_x \rangle_p$ in the training set:
      i. $features_t \leftarrow$ extract features (similarity based, WordNet based, structural) (Section 4.2.1)
      ii. $l_{t_1}, l_{t_2} \leftarrow$ determine labels for containers c_{t_1} and c_{t_2} based on the verb category of w_t.
      iii. append $\langle features_t, l_{t,1} \rangle$, $\langle features_t, l_{t,2} \rangle$ to $instances_1, instances_2$.
   (c) $M_1, M_2 \leftarrow$ train two SVMs for $instances_1, instances_2$

3. **Solving: for every problem p in the test set** (Section 4.3)
   (a) **Identifying verb categories in sentences**
      i. for every sentence $w_t \in \langle w_1, \ldots, w_T, w_x \rangle_p$:
         A. $features_t \leftarrow$ extract features (similarity based, WordNet based, structural).
         B. $l_{t_1}, l_{t_2} \leftarrow$ classify w_t for both containers c_{t_1} and c_{t_2} using models M_1, M_2.
   (b) **State progression**: Form $\langle s_0, \ldots, s_T \rangle$ (Section 4.3.1)
      i. $s_0 \leftarrow null$.
      ii. for $t \in \langle 1, \ldots, T \rangle$: $s_t \leftarrow progress(s_{t-1}, w_t)$.
         A. if $e_t = e_x$ and $a_t = a_x$:
            if w_t is an observation: $N_t(c_{t_1}, e_t) = num_t$.
            else: update $N_t(c_{t_1}, e_t)$ and $N_t(c_{t_2}, e_t)$ given verb categories l_{t_1}, l_{t_2}.
         B. copy $N_{t-1}(c, e)$ to $N_t(c, e)$ for all other (c, e) pairs.
   (c) **Forming equations and solution** (Section 4.3.2)
      i. Mark each w_t that matches with w_x if:
      a) c_{t_1} matches with c_x and verb categories are equal or verbs are similar.
      b) c_{t_2} matches with c_x and the verbs are in opposite categories.
      ii. $x \leftarrow$ the unknown quantity if w_x matches with a sentence introducing an unknown number
      iii. If the question asks about an unknown variable x or a start variable (w_x contains "begin" or "start"):
      For some container c, find two states s_t (quantity expression contains x) and s_{t+1} (quantity is a known number). Then, form an equation for x: $N_t(c, e_x) = N_{t+1}(c, e_x)$.
      iv. else: form equation as $x = N_t(c_x, e_x)$.
      v. Solve the equation and return the absolute value of x.

Figure 3: ARIS: a method for solving arithmetic word problems.

used to prune the irrelevant information in progressing world states.

Arithmetic problems usually include sentences with no attributes for the entities. For example, the attribute `black` has not been explicitly mentioned for the `kitten` in the second sentence. In particular, ARIS updates an implicit attribute using the previously observed attribute. For example, in "Joan went to 4 football games this year. She went to 9 games last year.", ARIS assigns `football` as an attribute of the `game` in both sentences.

4.2 Training for Verb Categories

This step involves training a model to identify verb categories for sentences. This entails predicting one label (increasing, decreasing) for each (verb, container) pair in the sentence. Each possible setting of these binary labels corresponds to one of the seven verb categories discussed earlier. For example, if c_1 is increasing and c_2 is decreasing this is a *positive transfer* verb.

Our dataset includes word problems from different domains (more details in Section 5.2). Each verb in our dataset is labeled with one of the 7 cat-

egories from Table 1.

For training, we compile a list of sentences from all the problems in the dataset and split sentences into training and test sets in two settings. In the first setting no instance from the same domain appears in the training and test sets in order to study the robustness of our method to new problem types. In the second setting no verb is repeated in the training and test sets in order to study how well our method predicts categories of unseen verbs.

For every sentence w_t in the problems, we build two data instances, (w_t, c_1) and (w_t, c_2), where c_1 and c_2 are containers extracted from the sentence. For every instance in the training data, we assign training labels using the verb categories of the sentences instead of labeling every sentence individually. The verb can be increasing or decreasing corresponding to every container in the sentence. For positive (negative) and construction (destruction) verbs, both instances are labeled positive (negative). For transfer positive (negative) verbs, the first instance is labeled positive (negative) and the second instance is labeled negative (positive). For observation verbs, both instances are labeled positive. We assume that the observation verbs are known (total of 5 verbs). Finally, we train Support Vector Machines given the extracted features and training labels explained above (Figure 3 step 2). In the following, we describe the features used for training.

4.2.1 Features

There are three sets of features: similarity based, Wordnet-based, and structural features. The first two sets of features focus on the verb and the third set focuses on the dependency structure of the sentence. All of our features are unlexicalized. This allows ARIS to handle verbs in the test questions which are completely different from those seen in the training data.

Similarity-based Features: For every instance (w, c), the feature vector includes similarity between the verb of the sentence w and a list of seed verbs. The list of seed verbs is automatically selected from a set $\mathcal{V}$ containing the 2000 most common English verbs using ℓ_1 regularized feature selection technique. We select a small set of seed verbs to avoid dominating the other feature types (structural and WordNet-based features).

The goal is to automatically select verbs from $\mathcal{V}$ that are most discriminative for each of the 7 verb categories in Table 1. We define 7 classification tasks: "Is a verb a member of each category?" Then, we select the three most representative verbs for each category. To do so, we randomly select a set of 65 verbs $\mathcal{V}_l$, from all the verbs in our dataset (118 in total) and manually annotate the verb categories. For every classification task, the feature vector $\mathbf{X}$ includes the similarity scores (Equation 1) between the verb v and all the verbs in the $\mathcal{V}$. We train an ℓ_1 regularized regression model (Park and Hastie, 2007) over the feature vector $\mathbf{X}$ to learn each category individually. The number of original (similarity based) features in $\mathbf{X}$ is relatively large, but ℓ_1 regularization provides a sparse weight vector. ARIS then selects the three most common verbs (without replacement) among the features (verbs) with non-zero weights. This accounts for 21 total seed verbs to be used for the main classification task. We find that in practice using this selection technique leads to better performance than using either all the verbs in $\mathcal{V}$ or using just the 65 randomly selected verbs.

Our method computes the similarity between two verbs v_1 and v_2 from the similarity between all the senses (from WordNet) of these verbs (Equation 1). We compute the similarity between two senses using linear similarity (Lin, 1998). The similarity between two synsets sv_1 and sv_2 are penalized according to the order of each sense for the corresponding verb. Intuitively, if a synset appears earlier in the set of synsets of a verb, it is more likely to be considered as the correct meaning. Therefore, later occurrences of a synset should result in reduced similarity scores. The similarity between two verbs v_1 and v_2 is the maximum similarity between two synsets of the verbs:

$$\text{sim}(v_1, v_2) = \max_{sv:\text{synsets}(v)} \frac{\text{lin-sim}(sv_1, sv_2)}{log(p_1 + p_2)} \quad (1)$$

where sv_1, sv_2 are two synsets, p_1, p_2 are the position of each synset match, and lin-sim is the linear similarity. Our experiments show better performance using linear similarity compared to other common similarity metrics (e.g., WordNet path similarity and Resnik similarity (Resnik, 1995)).

WordNet-based Features: We use WordNet verb categories in the feature vector. For each part of speech in WordNet, the synsets are organized into different categories. There are 15 categories for verbs. Some examples in-

clude "verb.communication", "verb.possession", and "verb.creation". In addition, WordNet includes the frequency measure $f_{c_{sv}}$ indicating how often the sense sv has appeared in a reference corpus. For each category i, we define the feature f_i as the ratio of the frequency of the sense sv_i over the total frequency of the verb i.e., $f_i = f_{c_{sv_i}}/f_{c_v}$.

Structural Features: For structural features, we use the dependency relations between the verb and the sentence elements since they can be a good proxy of the sentence structure. ARIS uses a binary vector including 35 dependency relations between the verb and other elements. For example, in the sentence "Joan picked 2 apples from the apple tree", the dependency between ('picked' and 'tree') and ('picked' and 'apples') are depicted as 'prep-from' and 'dobj' relations in the dependency parser, respectively. In addition, we include the length of the path in the dependency parse from the entity to the verb.

4.3 Solving the Problem

So far, ARIS grounds every problem into entities, containers, and attributes, and learns verb categories in sentences. Solving the problem consists of two main steps: (1) progressing states based on verb categories in sentences and (2) forming the equation.

4.3.1 State Progression with Verb Categories

This step (Figure 3 step 3b) involves forming states $\langle s_1, \ldots, s_T \rangle$ by updating quantities in every container using learned verb categories (Figure 3 step 3a). ARIS initializes s_0 to an empty state. It then iteratively updates the state s_t by progressing the state s_{t-1} given the sentence w_t with the verb v, entity e, number num, and containers c_1 and c_2.

For a given sentence t, ARIS attempts to match e_t and c_t to entities and categories in s_{t-1}. An entity/category is matched if has the same head word and same set of attributes as an existing entity/category. If an entity or category cannot be matching to one in s_{t-1}, then a new one is created in s_t.

The progress subroutine prunes the irrelevant sentences by checking if the entity e and its attributes a agree with the question entity e_x and its attributes a_x in the question. For example both `game` entities agree with the question entity in the problem "Joan went to 4 football games this year. She went to 9 games last year. How many football games did Joan go?". The first entity has an explicit `football` attribute, and the second entity has been assigned the same attribute (Section 4.1). Even if the question asks about `games` without mentioning `football`, the two sentences will match the question. Note that the second sentence would have not been matched if there was an explicit mention of the 'basketball game' in the second sentence.

For the matched entities, ARIS initializes or updates the values of the containers c_1, c_2 in the state s_t. ARIS uses the learned verb categories in sentences (Section 4.2) to update the values of containers. For an observation sentence w_t, the value of c_1 in the state s_t is assigned to the observed quantity num. For other sentence types, if the container c does not match to a container the previous state, its value is initialized with a start variable C_0. For example, the container `Joan` is initialized with J_0 at the state s_1 (Figure 2). Otherwise, the values of c_1 and c_2 are updated according to the verb category in the sentence. For instance, if the verb category in the sentence is a positive transfer then $N_t(c_1, e) = N_{t-1}(c_1, e) - num$ and $N_t(c_2, e) = N_{t-1}(c_2, e) + num$ where $N_t(c, e)$ represents the quantity of e in the container c at state s_t (Figure 2).

4.3.2 Forming Equations and Solution

The question entity e_x can match either to an entity in the final state, or to some unknown generated during the state progression. Concretely, the question sentence w_x asks about the quantity x of the entity e_x in a container c_x at a particular state s_u or a transition after the sentence w_u (Figure 3 step 3c).

To determine if e_x matches to an unknown variable, we define a matching subroutine between the question sentence w_x and every sentence w_t to check entities, containers, and verbs (Figure 3 step 3(c)i). We consider two cases. 1) When w_x contains the words "begin", or "start", the unknown variable is about the initial value of an entity, and it is set to the start variable of the container c_x (Figure 3 step 3(c)iii). For example, in "Bob had balloons. He gave 9 to his friends. He now has 4 balloons. How many balloons did he have to start with?", the unknown variable is set to the start variable B_0. 2) When the question verb is not one of the defined set of observation verbs, ARIS attempts to match e_x with an unknown introduced by one of the state transitions (Figure 3

step 3(c)iii). For example, the second sentence in Figure 1 introduces an unknown variable over `kittens`. The matching subroutine matches this entity with the question entity since the question container, i.e. `Joan`, matches with the second container and verb categories are complementary.

In order to solve for the unknown variable x, ARIS searches through consecutive states s_t and s_{t+1}, where in s_t, the quantity of e_x for a container c is an expression over x, and in s_{t+1}, the quantity is a known number for a container matched to c. It then forms an equation by comparing the quantities for containers matched between the two states. In the previous example, the equation will be $B_0 - 9 = 4$ by comparing states s_2 and s_3, where the numerical expression over `balloons` is $B_0 - 9$ in the state s_2, and the quantity is a known number in the state s_3.

When neither of the two above cases apply, ARIS matches e_x to an entity in the final state, s_T and returns its quantity, (Figure 3 step 3(c)iv). In the `football` example of the previous section, the equation will be $x = N_t(c_x, e_x)$, where $N_t(c_x, e_x)$ is the quantity in the final state.

Finally, the equation will be solved for the unknown variable x and the absolute value of the unknown variable is returned.

5 Experiments

To experimentally evaluate our method we build a dataset of arithmetic word problems along with their correct solutions. We test our method on the accuracy of solving arithmetic word problems and identifying verb categories in sentences.

5.1 Experimental Setup

Datasets: We compiled three diverse datasets MA1, MA2, IXL (Table 2) of Arithmetic word problems on addition and subtraction for third, fourth, and fifth graders. These datasets have similar problem types, but have different characteristics. Problem types include combinations of additions, subtractions, one unknown equations, and U.S. money word problems. Problems in MA2 include more irrelevant information compared to the other two datasets, and IXL includes more information gaps. In total, they include 395 problems, 13,632 words, 118 verbs, and 1,483 sentences.

Tasks and Baselines: We evaluate ARIS on two tasks: 1) solving arithmetic word problems in the three datasets and 2) classifying verb categories in sentences. We use the percentage of correct answers to the problems as the evaluation metric for the first task and accuracy as the evaluation metric for the second task. We use Weka's SVM (Witten et al., 1999) with default parameters for classification which is trained with verb categories in sentences (as described in Section 4.2).

	Source	#Tests	Avg.# Sentences
MA1	math-aids.com	134	3.5
IXL	ixl.com	140	3.36
MA2	math-aids.com	121	4.48

Table 2: Properties of the datasets.

	MA1	IXL	MA2	Total
3-fold Cross validation				
ARIS	83.6	**75.0**	**74.4**	**77.7**
ARIS$_2$	83.9	75.4$^+$	69.8$^+$	76.5$^+$
KAZB	**89.6**	51.1	51.2	64.0
Majority	45.5	71.4	23.7	48.9
Gold sentence categorization				
Gold ARIS	94.0	77.1	81.0	84.0

Table 3: Accuracy of solving arithmetic word problems in three datasets MA1, IXL, and MA2. This table compares our method, ARIS, ARIS$_2$ with the state-of-the-art KAZB. All methods are trained on two (out of three) datasets and tested on the other one. ARIS$_2$ is trained when no verb is repeated in the training and test sets. Gold ARIS uses gold verb categories. The improvement of ARIS (boldfaced) and ARIS$_2$ (denoted by $^+$) are significant over KAZB and the majority baseline with $p < 0.05$.

For the first task, we compare ARIS with KAZB (Kushman et al., 2014), majority baseline, ARIS$_2$, and Gold ARIS. KAZB requires training data in the form of equation systems and numerical answers to the problems. The majority baseline classifies every instance as increasing. In ARIS$_2$ (a variant of ARIS) the system is trained in a way that no verb is repeated in the training and test sets. Gold ARIS uses the ground-truth sentence categories instead of predicted ones. For the second task, we compare ARIS with a baseline that uses WordNet verb senses.

5.2 Results

We evaluate ARIS in solving arithmetic word problems in the three datasets and then evaluate its ability in classifying verb categories in sentences.

5.2.1 Solving Arithmetic Problems

Table 3 shows the accuracy of ARIS in solving problems in each dataset (when trained on the other two datasets).Table 3 shows that ARIS

significantly outperforms KAZB and the majority baseline. As expected, ARIS shows a larger gain on the two more complex datasets MA2 and IXL; our method shows promising results in dealing with irrelevant information (dataset MA2) and information gaps (dataset IXL). This is because ARIS learns to classify verb categories in sentences and does not require observing similar patterns/templates in the training data. Therefore, ARIS is more robust to differences between the training and test datasets and can generalize across different dataset types. As discussed in the experimental setup, the datasets have mathematically similar problems, but differ in the natural language properties such as in the sentence length and irrelevant information (Table 2).

Table 3 also shows that the sentence categorization is performed with high accuracy even if the problem types and also the verbs are different. In particular, there are a total of 118 verbs among which 64 verbs belong to MA datasets and 54 are new to IXL. To further study this, we train our method ARIS_2 in which no verb can be repeated in the training and test sets. ARIS_2 still significantly outperforms KAZB. In addition, we observe only a slight change in accuracy between ARIS and ARIS_2.

To further understand our method, we study the effect of verb categorization in sentences in solving problems. Table 3 shows the results of Gold ARIS in solving arithmetic word problems with gold sentence categorizations. In addition, comparing ARIS with Gold ARIS suggests that our method is able to reliably identify verb categories in sentences.

We also perform an experiment where we pool all of the problems in the three datasets and randomly choose 3 folds for the data (instead of putting each original dataset into it's own fold). We compare our method with KAZBin this scenario. In this setting, our method's accuracy is 79.5% while KAZB's accuracy is 81.8%. As expected, our method's performance has not changed significantly from the previous setting, while KAZB's performance significantly improves because of the reduced diversity between the training and test sets in this scenario.

5.2.2 Sentence Categorization

Table 4 compares accuracy scores of sentence categorization for our method with different features, a baseline that uses WordNet verb senses, and the majority baseline that assigns every (verb, container) pair as increasing. Similar to ARIS_2, we randomly split verbs into three equal folds and assign the corresponding sentences to each fold. No verb is shared between training and test sets. We then directly evaluate the accuracy of the SVM's verb categorization (explained in Section 4.2). This table shows that ARIS performs well in classifying sentence categories even with new verbs in the test set. This suggests that our method can generalize well to predict verb categories for unseen verbs.

Table 4 also details the performance of four variants of our method that ablate various features of ARIS. The table shows that similarity, contextual, and WordNet features are all important to the performance of ARIS in verb categorization, whereas the WordNet features are less important for solving the problems. In addition, it shows that similarity features play more important roles. We also performed another experiment to study the effect of the proposed feature selection method for similarity-based features. The accuracy of ARIS in classifying sentence categories is 69.7% when we use all the verbs in $\mathcal{V}$ in the similarity feature vector. This shows that our feature selection algorithm for selecting seed verbs is important towards categorizing verbs.

Finally, Table 4 shows that our method significantly outperforms the baseline that only uses WordNet verb sense. An interesting observation is that the majority baseline in fact outperforms WordNet verb senses in verb categorization, but is significantly worse in solving arithmetic word problems. In addition, we evaluate the accuracy of predicting only verb categories by assigning the verb label according to the majority of its labels in the sentence categories. The accuracy of verb categories is 78.2% confirming that ARIS is able to successfully categorize verbs.

5.2.3 Error Analysis

We analyzed all 63 errors of Gold ARIS and present our findings in Table 5. There are five major classes of errors. In the first category, some information is not mentioned explicitly and should be entailed. For example, 'washing cars' is the source of 'making money'. Despite the improvements that come from ARIS, a large portion of the errors can still be attributed to irrelevant information. For example, 'short' is not a 'toy'. The third category refers to errors that require knowledge

	Categorization	Solution
ARIS	**81.2**$^+$	**76.5**$^+$
No similarity features	68.8	65.4
No WordNet features	75.3	78.0$^+$
No structural features	75.5	72.4$^+$
Baseline (WordNet)	67.8	68.4
Majority Baseline	73.4	48.9

Table 4: Ablation study and baseline comparisons: this table reports the accuracy of verb categorization in sentences and solutions for ARIS with ablating features. It also provides comparisons to WordNet and majority baselines. The improvement of ARIS (boldfaced) and ablations denoted by $^+$ are statistically significant over the baselines (with $p < 0.05$) for both tasks.

Error type	Example
Entailment, Implicit Action (26%)	Last week Tom had $74. He **washed cars** over the weekend and now has $86. How much money did he make washing cars?
Irrelevant Information (19%)	Tom bought a skateboard for $9.46, and spent $9.56 on marbles. Tom also spent $14.50 on **shorts**. In total, how much did Tom spend on **toys**?
Set Completion (13%)	Sara's school **played** 12 games this year. They **won** 4 games. How many games did they **lose**?
Parsing Issues (21%)	Sally had 27 Pokemon cards. Dan gave **her** 41 new Pokemon cards. How many Pokemon cards does Sally have now?
Others (21%)	In March it rained 0.81 inches. It rained 0.35 inches **less** in April than in March. How much did it rain in April?

Table 5: Examples of different error categories and relative frequencies. The cause of error is **bolded**.

about set completions. For example, the 'played' games can be split into 'win' and 'lost' games. Finally, parsing and coreference mistakes are another source of errors for ARIS.

6 Discussions and Conclusion

In this paper we introduce ARIS, a method for solving arithmetic word problems. ARIS learns to predict verb categories in sentences using syntactic and (shallow) semantic features from small, easy-to-obtain training data. ARIS grounds the world state into entities, sets, quantities, attributes, and their relations and takes advantage of the circumscription assumption and successfully fills in the information gaps. Finally, ARIS makes use of attributes and discards irrelevant information in the problems. Together these provide a new representation and a learning algorithm for solving arithmetic word problems.

This paper is one step toward building a system that can solve any math and logic word problem. Our empirical evaluations show that our method outperforms a template-based learning method (developed recently by Kushman et al. (2014)) on solving addition and subtraction problems with diversity between the training and test sets. In particular, our method generalizes better to data from different domains because ARIS only relies on learning verb categories which alleviates the need for equation templates for arithmetic problems. In this paper, we have focused on addition and subtraction problems. However, KAZB can deal with more general types of problems such as multiplication, division, and simultaneous equations.

We have observed a complementary behavior between our method and that of Kushman et al. This suggests a hybrid approach that can benefit from the strengths of both methods while being applicable to more general problems while robust to the errors specific to each. In addition, we plan to focus on incrementally collecting domain knowledge to deal with missing information gaps. Another possible direction is to improve parsing and coreference resolution.

Acknowledgments

The research was supported by the Allen Institute for AI, and grants from the NSF (IIS-1352249) and UW-RRF (65-2775). We thank Ben Hixon and the anonymous reviewers for helpful comments and the feedback on the work.

References

Antoine Bordes, Nicolas Usunier, and Jason Weston. 2010. Label ranking under ambiguous supervision for learning semantic correspondences. In *Proc. International Conference on Machine Learning (ICML)*.

SRK Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proc. of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the AFNLP (ACL-AFNLP)*.

SRK Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. 2012. Learning high-level planning from text. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proc. of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the AFNLP (ACL-AFNLP)*.

David Chen, Joohyun Kim, and Raymond Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *Journal of Artificial Intelligence Research*, 37.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. Language Resources and Evaluation Conference (LREC)*.

Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. 2009. Reading to learn: Constructing features from semantic abstracts. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Edward A. Feigenbaum and Julian Feldman, editors. 1963. *Computers and Thought*. McGraw Hill, New York.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ruifang Ge and Raymond J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Dan Goldwasser and Dan Roth. 2011. Learning from natural instructions. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hannaneh Hajishirzi, Julia Hockenmaier, Erik T. Mueller, and Eyal Amir. 2011. Reasoning about robocup soccer narratives. In *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*.

Rohit J. Kate and Raymond J. Mooney. 2007. Learning language semantics from ambiguous supervision. In *Proc. Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceeding of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Iddo Lev, Bill MacCartney, Christopher D. Manning, , and Roger Levy. 2004. Solving logic puzzles: From robust processing to precise semantics. In *Workshop on Text Meaning and Interpretation at Association for Computational Linguistics (ACL)*.

Iddo Lev. 2007. *Packed Computation of Exact Meaning Representations*. Ph.D. thesis, CS, Stanford University.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proc. of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the AFNLP (ACL-AFNLP)*.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proc. International Conference on Machine Learning (ICML)*.

John McCarthy. 1980. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38.

Mee Young Park and Trevor Hastie. 2007. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *International joint conference on Artificial intelligence (IJCAI)*.

Benjamin Snyder and Regina Barzilay. 2007. Database-text alignment via structured multilabel classification. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

Adam Vogel and Daniel Jurafsky. 2010. Learning to follow navigational directions. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ian H Witten, Eibe Frank, Leonard E Trigg, Mark A Hall, Geoffrey Holmes, and Sally Jo Cunningham. 1999. Weka: Practical machine learning tools and techniques with java implementations.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*.

NaturalLI: Natural Logic Inference for Common Sense Reasoning

Gabor Angeli
Stanford University
Stanford, CA 94305
angeli@cs.stanford.edu

Christopher D. Manning
Stanford University
Stanford, CA 94305
manning@cs.stanford.edu

Abstract

Common-sense reasoning is important for AI applications, both in NLP and many vision and robotics tasks. We propose NaturalLI: a Natural Logic inference system for inferring common sense facts – for instance, that *cats have tails* or *tomatoes are round* – from a very large database of known facts. In addition to being able to provide strictly valid derivations, the system is also able to produce derivations which are only *likely* valid, accompanied by an associated confidence. We both show that our system is able to capture strict Natural Logic inferences on the FraCaS test suite, and demonstrate its ability to predict common sense facts with 49% recall and 91% precision.

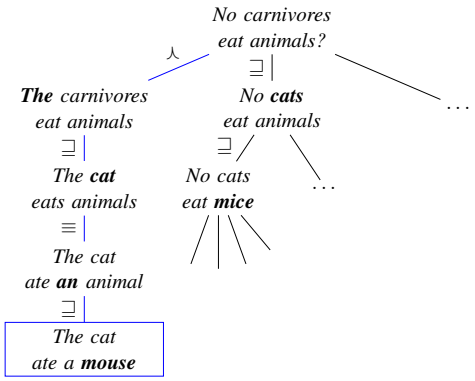


Figure 1: Natural Logic inference cast as search. The path to the boxed premise *the cat ate a mouse* disproves the query *no carnivores eat animals*, as it passes through the negation relation (人). This path is one of many candidates taken; the premise is one of many known facts in the database. The edge labels denote Natural Logic inference steps.

1 Introduction

We approach the task of *database completion*: given a database of true facts, we would like to predict whether an unseen fact is true and should belong in the database. This is intuitively cast as an inference problem from a collection of candidate premises to the truth of the query. For example, we would like to infer that *no carnivores eat animals* is false given a database containing *the cat ate a mouse* (see Figure 1).

These inferences are difficult to capture in a principled way while maintaining high recall, particularly for large scale open-domain tasks. Learned inference rules are difficult to generalize to arbitrary relations, and standard IR methods easily miss small but semantically important lexical differences. Furthermore, many methods require explicitly modeling either the database, the query, or both in a formal meaning representation (e.g., Freebase tuples).

Although projects like the Abstract Meaning Representation (Banarescu et al., 2013) have made headway in providing broad-coverage meaning representations, it remains appealing to use human language as the vessel for inference. Furthermore, OpenIE and similar projects have been very successful at collecting databases of natural language snippets from an ever-increasing corpus of unstructured text. These factors motivate our use of Natural Logic – a proof system built on the syntax of human language – for broad coverage database completion.

Prior work on Natural Logic has focused on inferences from a single relevant premise, making use of only formally valid inferences. We improve upon computational Natural Logic in three ways: (i) our approach operates over a very large set of candidate premises simultaneously; (ii) we do not require explicit alignment between a premise and the query; and (iii) we allow imprecise inferences at an associated cost learned from data.

Our approach casts inference as a single unified search problem from a query to any valid

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 534–545,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

supporting premise. Each transition along the search denotes a (reverse) inference step in Natural Logic, and incurs a cost reflecting the system's confidence in the validity of that step. This approach offers two contributions over prior work in database completion: (i) it allows for unstructured text as the input database without any assumptions about the schema or domain of the text, and (ii) it proposes Natural Logic for inference, rather than translating to a formal logic syntax. Moreover, the entire pipeline is implemented in a single elegant search framework, which scales easily to large databases.

2 MacCartney's Natural Logic

Natural Logic aims to capture common logical inferences by appealing directly to the structure of language, as opposed to running deduction on an abstract logical form. The logic builds upon traditional rather than first-order logic: to a first approximation, Natural Logic can be seen as an enhanced version of Aristotle's syllogistic system (van Benthem, 2008). A working understanding of the logic as syllogistic reasoning is sufficient for understanding the later contributions of the paper. While some inferences of first-order logic are not captured by Natural Logic, it nonetheless allows for a wide range of intuitive inferences in a computationally efficient and conceptually clean way.

We build upon the variant of the logic introduced by the NatLog system (MacCartney and Manning, 2007; 2008; 2009), based on earlier theoretical work on Natural Logic and Monotonicity Calculus (van Benthem, 1986; Valencia, 1991). Later work formalizes many aspects of the logic (Icard, 2012; Djalali, 2013); we adopt the formal semantics of Icard and Moss (2014), along with much of their notation.

At a high level, Natural Logic proofs operate by mutating spans of text to ensure that the mutated sentence follows from the original – each step is much like a syllogistic inference. We construct a complete proof system in three parts: we define MacCartney's atomic relations between lexical entries (Section 2.1), the effect these lexical mutations have on the validity of the sentence (Section 2.2), and a practical approach for executing these proofs. We review MacCartney's alignment-based approach in Section 2.3, and show that we can generalize and simplify this system in Section 3.

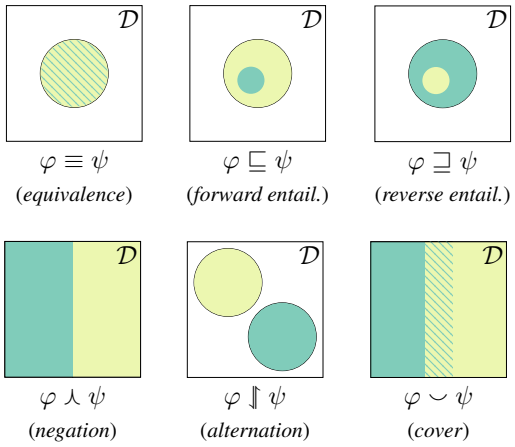


Figure 2: The model-theoretic interpretation of the MacCartney relations. The figure shows the relation between the denotation of φ (dark) and ψ (light). The universe is denoted by $\mathcal{D}$.

2.1 Lexical Relations

MacCartney and Manning (2007) introduce seven set-theoretic relations between the denotations of any two lexical items. The denotation of a lexical item is the set of objects in the domain of discourse $\mathcal{D}$ to which that lexical item refers. For instance, the denotation of *cat* would be the set of all cats. Two denotations can then be compared in terms of set algebra: if we define the set of cats to be φ and the set of animals to be ψ, we can state that $\varphi \subseteq \psi$.

The six informative relations are summarized in Figure 2; a seventh relation ($\#$) corresponds to to the completely uninformative relation. For instance, the example search path in Figure 1 makes use of the following relations:

$$\begin{array}{rcl} \textit{No } x\ y & \perp\!\!\!\perp & \textit{The } x\ y \\ \textit{cat} & \sqsubseteq & \textit{carnivore} \\ \textit{animal} & \equiv & \textit{a animal} \\ \textit{animal} & \sqsupseteq & \textit{mouse} \end{array}$$

Denotations are not required to be in the space of predicates (e.g., *cat*, *animal*). In the first example, the denotations of *No* and *The* are in the space of operators $p \to (p \to t)$: functions from predicates p to truth values t. The $\perp\!\!\!\perp$ relation becomes the conjunction of two claims: $\forall x \forall y\ \neg(\textit{no}\ x\ y \wedge \textit{the}\ x\ y)$ and $\forall x \forall y\ (\textit{no}\ x\ y \vee \textit{the}\ x\ y)$. This is analogous to the construction of the set-theoretic definition of $\perp\!\!\!\perp$ in Figure 2: $\varphi \cap \psi = \varnothing$ and $\varphi \cup \psi = \mathcal{D}$ (see Icard and Moss (2014)).

Examples of the last two relations ($\Rightarrow$ and $\smile$) and the complete independence relation ($\#$) include:

$$\begin{array}{rcl} \textit{cat} & \Rightarrow & \textit{dog} \\ \textit{animal} & \smile & \textit{nonhuman} \\ \textit{cat} & \# & \textit{friendly} \end{array}$$

2.2 Monotonicity and Polarity

The previous section details the relation between lexical items; however, we still need a theory for how to "project" the relation induced by a lexical mutation as a relation between the two containing sentences. For example, *cat* $\sqsubseteq$ *animal*, and *some cat meows* $\sqsubseteq$ *some animal meows*, but *no cat barks* $\not\sqsubseteq$ *no animal barks*. Despite differing by the same lexical relation, the first example describes a valid entailment, while the second does not.

We appeal to two important concepts: *monotonicity* as a property of arguments to natural language operators, and *polarity* as a property of lexical items in a sentence. Much like monotone functions in calculus, an [upwards] monotone operator has an output truth value which is non-decreasing (i.e., material implication) as the input "increases" (i.e., the subset relation). From the example above, *some* is upwards monotone in its first argument, and *no* is downwards monotone in its first argument.

Polarity is a property of lexical items in a sentence determined by the operators acting on it. All lexical items have *upward* polarity by default; upwards monotone operators preserve polarity, and downwards monotone operators reverse polarity. For example, *mice* in ***no*** *cats eat mice* has downward polarity, whereas *mice* in ***no*** *cats do**n't** eat mice* has upward polarity (it is in the scope of two downward monotone operators). The relation between two sentences differing by a single lexical relation is then given by the projection function ρ in Table 1.[1]

2.3 Proof By Alignment

MacCartney and Manning (2007) approach the inference task in the context of inferring whether a single relevant premise entails a query. Their approach first generates an alignment between the premise and the query, and then classifies each aligned segment into one of the lexical relations in Figure 2. Inference reduces to projecting each

r	$\equiv$	$\sqsubseteq$	$\sqsupseteq$	$\Rightarrow$	$\smile$	$\wedge$	$\#$
$\rho(r)$	$\equiv$	$\sqsupseteq$	$\sqsubseteq$	$\smile$	$\Rightarrow$	$\wedge$	$\#$

Table 1: The projection function ρ, shown for downward polarity contexts only. The input r is the lexical relation between two words in a sentence; the projected relation $\rho(r)$ is the relation between the two sentences differing only by that word. Note that ρ is the identity function in upward polarity contexts.

$\bowtie$	$\equiv$	$\sqsubseteq$	$\sqsupseteq$	$\wedge$	$\Rightarrow$	$\smile$	$\#$
$\equiv$	$\equiv$	$\sqsubseteq$	$\sqsupseteq$	$\wedge$	$\Rightarrow$	$\smile$	$\#$
$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$	$\#$	$\Rightarrow$	$\Rightarrow$	$\#$	$\#$
$\sqsupseteq$	$\sqsupseteq$	$\#$	$\sqsupseteq$	$\smile$	$\#$	$\smile$	$\#$
$\wedge$	$\wedge$	$\smile$	$\Rightarrow$	$\equiv$	$\sqsupseteq$	$\sqsubseteq$	$\#$
$\Rightarrow$	$\Rightarrow$	$\#$	$\Rightarrow$	$\sqsubseteq$	$\#$	$\sqsubseteq$	$\#$
$\smile$	$\smile$	$\smile$	$\#$	$\sqsupseteq$	$\sqsupseteq$	$\#$	$\#$
$\#$	$\#$	$\#$	$\#$	$\#$	$\#$	$\#$	$\#$

Table 2: The join table as shown in Icard (2012). Entries in the table are the result of joining a row with a column.

of these relations according to the projection function ρ (Table 1) and iteratively *joining* two projected relations together to get the final entailment relation. This join relation, denoted as $\bowtie$, is given in Table 2.

To illustrate, we can consider MacCartney's example inference from *Stimpy is a cat* to *Stimpy is not a poodle*. An alignment of the two statements would provide three lexical mutations: $r_1 \coloneqq \textit{cat} \rightarrow \textit{dog}$, $r_2 \coloneqq \cdot \rightarrow \textit{not}$, and $r_3 \coloneqq \textit{dog} \rightarrow \textit{poodle}$. Each of these are then projected with the projection function ρ, and are joined using the join relation:

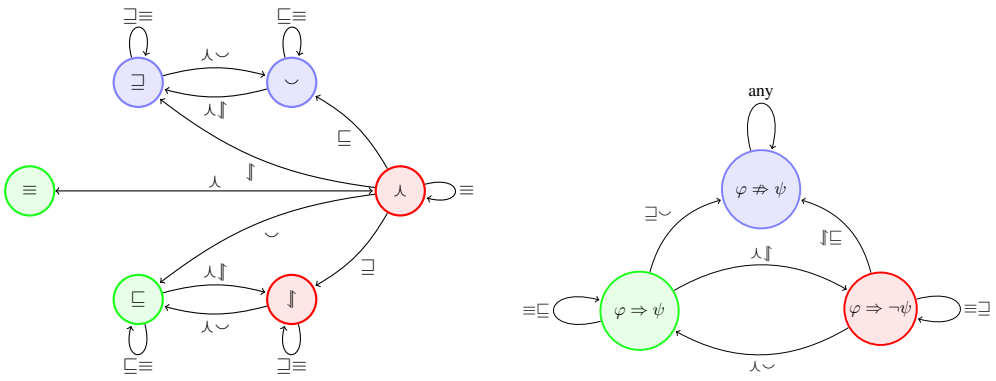
$$r_0 \bowtie \rho(r_1) \bowtie \rho(r_2) \bowtie \rho(r_3),$$

where the initial relation r_0 is axiomatically $\equiv$. In MacCartney's work this style of proof is presented as a table. The last column (s_i) is the relation between the premise and the i^{th} step in the proof, and is constructed inductively as $s_i \coloneqq s_{i-1} \bowtie \rho(r_i)$:

	Mutation	r_i	$\rho(r_i)$	s_i
r_1	*cat*$\rightarrow$*dog*	$\Rightarrow$	$\Rightarrow$	$\Rightarrow$
r_2	$\cdot \rightarrow$*not*	$\wedge$	$\wedge$	$\sqsubseteq$
r_3	*dog*$\rightarrow$*poodle*	$\sqsupseteq$	$\sqsubseteq$	$\sqsubseteq$

In our example, we would conclude that *Stimpy is a cat* $\sqsubseteq$ *Stimpy is not a poodle* since s_3 is $\sqsubseteq$; therefore the inference is valid.

[1] Note that this table optimistically assumes every operator is *additive* and *multiplicative*, as defined in Icard (2012).



(a) (b)

Figure 3: (a) Natural logic inference expressed as a finite state automaton. Omitted edges go to the unknown state (#), with the exception of omitted edges from ≡, which go to the state of the edge type. Green states (≡, ⊑) denote valid inferences; red states (⇂, ⅄) denote invalid inferences; blue states (⊒, ⌣) denote inferences of unknown validity. (b) The join table collapsed into the three meaningful states over truth values.

3 Inference as a Finite State Machine

We show that the tabular proof formulation from Section 2.3 can be viewed as a finite state machine, and present a novel observation that we can losslessly collapse this finite state machine into only three intuitive inference states. These observations allow us to formulate our search problem such that a search path corresponds to an input to (i.e., path through) this collapsed state machine.

Taking notation from Section 2.3, we construct a finite state machine over states $s \in \{\sqsubseteq, \sqsupseteq, \dots\}$. A machine in state s_i corresponds to relation s_i holding between the initial premise and the derived fact so far. States therefore correspond to states of *logical validity*. The start state is ≡. Outgoing transitions correspond to *inference steps*. Each transition is labeled with a projected relation $\rho(r) \in \{\sqsubseteq, \sqsupseteq, \dots\}$, and spans from a source state s to a target s' according to the join table. That is, the transition $s \xrightarrow{\rho(r)} s'$ exists iff $s' = s \bowtie \rho(r)$. For example, the path in Figure 1 yields the transitions $\equiv \xrightarrow{⅄} ⅄ \xrightarrow{\sqsupseteq} ⇂ \xrightarrow{\equiv} ⇂ \xrightarrow{\sqsupseteq} ⇂$. Figure 3a shows the automaton, with trivial edges omitted for clarity.

Our second contribution is collapsing this automaton into the three meaningful states we use as output: *valid* ($\varphi \Rightarrow \psi$), *invalid* ($\varphi \Rightarrow \neg\psi$), and *unknown validity* ($\varphi \nRightarrow \psi$). We can cluster states in Figure 3a into these three categories. The relations ≡ and ⊑ correspond to valid inferences; ⅄ and ⇂ correspond to invalid inferences; ⊒, ⌣ and # correspond to unknown validity. This clustering mirrors that used by MacCartney for his textual entailment experiments.

Collapsing the FSA into the form in Figure 3b becomes straightforward from observing the regularities in Figure 3a. Nodes in the valid cluster transition to invalid nodes always and only on the relations ⅄ and ⇂. Symmetrically, invalid nodes transition to valid nodes always and only on ⅄ and ⌣. A similar pattern holds for the other transitions.

Formally, for every relation r and nodes a_1 and a_2 in the same cluster, if we have transitions $a_1 \xrightarrow{r} b_1$ and $a_2 \xrightarrow{r} b_2$ then b_1 and b_2 are necessarily in the same cluster. As a concrete example, we can take $r = ⅄$ and the two states in the *invalid* cluster: $a_1 = ⅄$, $a_2 = ⇂$. Although $⅄ \xrightarrow{⅄} \equiv$ and $⇂ \xrightarrow{⅄} \sqsubseteq$, both ≡ and ⊑ are in the same cluster (*valid*). It is not trivial *a priori* that the join table should have this regularity, and it certainly simplifies the logic for inference tasks.

A few observations deserve passing remark. First, even though the states ⊒ and ⌣ appear meaningful, in fact there is no "escaping" these states to either a valid or invalid inference. Second, the hierarchy over relations presented in Icard (2012) becomes apparent – in particular, ⅄ always behaves as negation, whereas its two "weaker" versions (⇂ and ⌣) only behave as negation in certain contexts. Lastly, with probabilistic inference, transitioning to the unknown state can be replaced with staying in the current state at a (potentially arbitrarily large) cost to the confidence of validity. This allows us to make use of only two states: *valid* and *invalid*.

4 Inference As Search

Natural Logic allows us to formalize our approach elegantly as a single search problem. Given a query, we search over the space of possible facts for a valid premise in our database. The nodes in our search problem correspond to candidate facts (Section 4.1); the edges are mutations of these facts (Section 4.2); the costs over these edges encode the confidence that this edge maintains an informative inference (Section 4.5). This mirrors the automaton defined in Section 3, except importantly we are constructing a reversed derivation, and are therefore "traversing" the FSA backwards.

This approach is efficient over a large database of 270 million entries without making use of explicit queries over the database; nor does the approach make use of any sort of approximate matching against the database, beyond lemmatizing individual lexical items. The motivation in prior work for approximate matches – to improve the recall of candidate premises – is captured elegantly by relaxing Natural Logic itself. We show that allowing invalid transitions with appropriate costs generalizes JC distance (Jiang and Conrath, 1997) – a common thesaurus-based similarity metric (Section 4.3). Importantly, however, the entire inference pipeline is done within the framework of weighted lexical transitions in Natural Logic.

4.1 Nodes

The space of possible nodes in our search is the set of possible partial derivations. To a first approximation, this is a pair (w, s) of a surface form w tagged with word sense and polarity, and an inference state $s \in \{\text{valid}, \text{invalid}\}$ in our collapsed FSA (Figure 3b). For example, the search path in Figure 1 traverses the nodes:

(*No carnivores eat animals*,	valid)
(*The carnivores eat animals*,	invalid)
(*The cat eats animals*,	invalid)
(*The cat eats an animal*,	invalid)
(*The cat ate a mouse*,	invalid)

During search, we assume that the validity states s are reversible – if we know that *the cat ate a mouse* is true, we can infer that *no carnivores eat animals* is false. In addition, our search keeps track of some additional information:

Mutation Index Edges between sentences are most naturally defined to correspond to mutations of individual lexical items. We therefore maintain an index of the next item to mutate at each search state. Importantly, this enforces that each derivation orders mutations left-to-right; this is computationally efficient, at the expense of rare search errors. A similar observation is noted in MacCartney (2009), where prematurely collapsing to # occasionally misses inferences.

Polarity Mutating operators can change the polarity on a span in the fact. Since we do not have the full parse tree at our disposal at search time, we track a small amount of metadata to guess the scope of the mutated operator.

4.2 Transitions

We begin by introducing some terminology. A *transition template* is a broad class of transitions; for instance WordNet hypernymy. A *transition* (or *transition instance*) is a particular instantiation of a transition template. For example, the transition from *cat* to *feline*. Lastly, an *edge* in the search space connects two nodes, which are separated by a single transition instance. For example, an edge exists between *some felines have tails* and *some cats have tails*. Transition [instances] are stored statically in memory, whereas edges are constructed on demand.

Transition templates provide a means of defining transitions and subsequently edges in our search space using existing lexical resources (e.g., WordNet, distributional similarity, etc.). We can then define a mapping from these templates to Natural Logic lexical relations. This allows us to map every edge in our search graph back to the Natural Logic relation it instantiates. The full table of transition templates is given in Table 3, along with the Natural Logic relation that instances of the template introduce. We include most relations in WordNet as transitions, and parametrize insertions and deletions by the part of speech of the token being inserted/deleted.

Once we have an edge defining a lexical mutation with an associated Natural Logic relation r, we can construct the corresponding end node (w', s') such that w' is the sentence with the lexical mutation applied, and s' is the validity state obtained from the FSA in Section 3. For instance, if our edge begins at (w, s), and there exists a transition in the FSA from $s' \xrightarrow{r} s$, then we define the end point of the edge to be (w', s'). To illustrate concretely, suppose our search state is:

(*some felines have tails*, valid)

Transition Template	Relation
WordNet hypernym	$\sqsubseteq$
WordNet hyponym	$\sqsupseteq$
WordNet antonym†	$\Downarrow$
WordNet synonym/pertainym†	$\equiv$
Distributional nearest neighbor	$\equiv$
Delete word†	$\sqsubseteq$
Add word†	$\sqsupseteq$
Operator weaken	$\sqsubseteq$
Operator strengthen	$\sqsupseteq$
Operator negate	$\curlywedge$
Operator synonym	$\equiv$
Change word sense	$\equiv$

Table 3: The edges allowed during inference. Entries with a dagger (†) are parametrized by their part-of-speech tag, from the restricted list of {noun,adjective,verb,other}. The first column describes the type of the transition. The set-theoretic relation introduced by each relation is given in the second column.

The transition template for WordNet hypernymy gives us a transition instance from *feline* to *cat*, corresponding to the Natural Logic inference *cat* $\stackrel{\sqsubseteq}{\Longrightarrow}$ *feline*. Recall, we are constructing the inference in reverse, starting from the consequent (query). We then notice that the transition $valid \stackrel{\sqsubseteq}{\Longrightarrow} valid$ in the FSA ends in our current inference state (*valid*), and set our new inference state to be the start state of the FSA transition – in this case, we maintain validity.

Note that negation is somewhat subtle, as the transitions are not symmetric from valid to invalid and visa versa, and we do not know our true inference state with respect to the premise yet. In practice, the search procedure treats all three of $\{\curlywedge, \Downarrow, \smile\}$ as negation, and re-scores complete derivations once their inference states are known.

It should be noted that the mapping from transition templates to relation types is intentionally imprecise. For instance, clearly nearest neighbors do not preserve equivalence ($\equiv$); more subtly, while *all cats like milk* $\Downarrow$ *all cats hate milk*, it is not the case that *some cats like milk* $\Downarrow$ *some cats hate milk*.[2] We mitigate this imprecision by introducing a cost for each transition, and learning the appropriate value for this cost (see Section 5). The cost of an edge from fact (w, v) with surface form w and validity v to a new fact (w', v'), using a transition instance t_i of template t and mutating a word with polarity p, is given by $f_{t_i} \cdot \theta_{t,v,p}$. We define this as:

f_{t_i}: A value associated with every transition instance t_i, intuitively corresponding to how "far" the endpoints of the transition are.

$\theta_{t,v,p}$: A learned cost for taking a transition of template t, if the source of the edge is in a inference state of v and the word being mutated has polarity p.

The notation for f_{t_i} is chosen to evoke an analogy to features. We set f_{t_i} to be 1 in most cases; the exceptions are the edges over the WordNet hypernym tree and the nearest neighbors edges. In the first case, taking the hypernymy relation from w to w' to be $\uparrow_{w \to w'}$, we set:

$$f_{\uparrow_{w \to w'}} = \log \frac{p(w')}{p(w)} = \log p(w') - \log p(w).$$

The value $f_{\downarrow_{w \to w'}}$ is set analogously. We define $p(w)$ to be the "probability" of a concept – that is, the normalized frequency of a word w or any of its hyponyms in the Google N-Grams corpus (Brants and Franz, 2006). Intuitively, this ensures that relatively long paths through fine-grained sections of WordNet are not unduly penalized. For instance, the path from *cat* to *animal* traverses six intermediate nodes, naïvely yielding a prohibitive search depth of 6. However, many of these transitions have low weight: for instance $f_{\uparrow_{cat \to feline}}$ is only 0.37.

For nearest neighbors edges, we take Neural Network embeddings learned in Huang et al. (2012) corresponding to each vocabulary entry. We then define $f_{NN_{w \to w'}}$ to be the arc cosine of the cosine similarity (i.e., the angle) between word vectors associated with lexical items w and w':

$$f_{NN_{w \to w'}} = \arccos\left(\frac{w \cdot w'}{\|w\|\|w'\|}\right).$$

For instance, $f_{NN_{cat \to dog}} = 0.43$. In practice, we explore the 100 nearest neighbors of each word.

We can express f_{t_i} as a feature vector by representing it as a vector with value f_{t_i} at the index corresponding to (t, v, p) – the transition template, the validity of the inference, and the polarity of the mutated word. Note that the size of this vector mirrors the number of cost parameters $\theta_{t,v,p}$, and

[2] The latter example is actually a consequence of the projection function in Table 1 being overly optimistic.

is in general smaller than the number of transition instances.

A search path can then be parametrized by a sequence of feature vectors $f_1, f_2, \ldots, f_n$, which in turn can be collapsed into a single vector $\mathbf{f} = \sum_i f_i$. The cost of a path is defined as $\theta \cdot \mathbf{f}$, where θ is the vector of $\theta_{t,v,p}$ values. Both $\mathbf{f}$ and θ are constrained to be non-negative, or else the search problem is misspecified.

4.3 Generalizing Similarities

An elegant property of our definitions of f_{t_i} is its ability to generalize JC distance. Let us assume we have lexical items w_1 and w_2, with a least common subsumer lcs. The JC distance $\text{dist}_{\text{jc}}(w_1, w_2)$ is:

$$\text{dist}_{\text{jc}}(w_1, w_2) = \log \frac{p(\text{lcs})^2}{p(w_1) \cdot p(w_2)}. \quad (1)$$

For simplicity, we simplify $\theta_{\uparrow,v,p}$ and $\theta_{\downarrow,v,p}$ as simply $\theta_\uparrow$ and $\theta_\downarrow$. Without loss of generality, we also assume that a path in our search is only modifying a single lexical item w_1, eventually reaching a mutated form w_2.

We can factorize the cost of a path, $\theta \cdot \mathbf{f}$, along the path from w_1 to w_2 through its lowest common subsumer (lcs), $[w_1, w_1^{(1)}, \ldots, \text{lcs}, \ldots, w_2^{(1)}, w_2]$, as follows:

$$\begin{aligned}
\theta \cdot \phi &= \theta_\uparrow \left(\left[\log p(w_1^{(1)}) - \log p(w_1) \right] + \ldots \right) + \\
&\quad \theta_\downarrow \left(\left[\log p(\text{lcs}) - \log p(w_1^{(n)}) \right] + \ldots \right) \\
&= \theta_\uparrow \left(\log \frac{p(\text{lcs})}{p(w_1)} \right) + \theta_\downarrow \left(\log \frac{p(\text{lcs})}{p(w_2)} \right) \\
&= \log \frac{p(\text{lcs})^{\theta_\uparrow + \theta_\downarrow}}{p(w_1)^{\theta_\uparrow} \cdot p(w_2)^{\theta_\downarrow}}.
\end{aligned}$$

Note that setting both $\theta_\uparrow$ and $\theta_\downarrow$ to 1 exactly yields Formula (1) for JC distance. This, in addition to the inclusion of nearest neighbors as transitions, allows the search to capture the intuition that similar objects have similar properties (e.g., as used in Angeli and Manning (2013)).

4.4 Deletions in Inference

Although inserting lexical items in a derivation (deleting words from the reversed derivation) is trivial, the other direction is not. For brevity, we refer to a deletion in the derivation as an insertion, since from the perspective of search we are inserting lexical items.

Naïvely, at every node in our search we must consider every item in the vocabulary as a possible insertion. We can limit the number of items we consider by storing the database as a trie. Since the search mutates the fact left-to-right (as per Section 4.1), we can consider children of a trie node as candidate insertions. To illustrate, given a search state with fact $w_0\, w_1 \ldots w_n$ and mutation index i, we would look up completions w_{i+1} for $w_0\, w_1 \ldots w_i$ in our trie of known facts.

Although this approach works well when i is relatively large, there are too many candidate insertions for small i. We special case the most extreme example for this, where $i = 0$ – that is, when we are inserting into the beginning of the fact. In this case, rather than taking all possible lexical items that start any fact, we take all items which are followed by the first word of our current fact. To illustrate, given a search state with fact $w_0\, w_1 \ldots w_n$, we would propose candidate insertions w_{-1} such that $w_{-1}\, w_0\, w_1' \ldots w_k'$ is a known fact for some $w_1' \ldots w_k'$. More concretely, if we know that *fluffy cats have tails*, and are at a node corresponding to *cats like boxes*, we propose *fluffy* as a possible insertion: *fluffy cats like boxes*.

4.5 Confidence Estimation

The last component in inference is translating a search path into a probability of truth. We notice from Section 4.2 that the *cost* of a path can be represented as $\theta \cdot \mathbf{f}$. We can normalize this value by negating every element of the cost vector θ and passing it through a sigmoid:

$$\text{confidence} = \frac{1}{1 + e^{-(-\theta \cdot \mathbf{f})}}.$$

Importantly, note that the cost vector must be non-negative for the search to be well-defined, and therefore the confidence value will be constrained to be between 0 and $\frac{1}{2}$.

At this point, we have a confidence that the given path has not violated strict Natural Logic. However, to translate this value into a probability we need to incorporate whether the inference path is confidently valid, or confidently invalid. To illustrate, a fact with a low confidence should translate to a probability of $\frac{1}{2}$, rather than a probability of 0. We therefore define the probability of validity as follows: We take v to be 1 if the query is in the *valid* state with respect to the premise, and -1 if the query is in the *invalid* state. For completeness, if no path is given we can set $v = 0$. The

probability of validity becomes:

$$p(\text{valid}) = \frac{v}{2} + \frac{1}{1 + e^{v\theta \cdot \mathbf{f}}}. \quad (2)$$

Note that in the case where $v = -1$, the above expression reduces to $\frac{1}{2}$ − confidence; in the case where $v = 0$ it reduces to simply $\frac{1}{2}$. Furthermore, note that the probability of truth makes use of the same parameters as the cost in the search.

5 Learning Transition Costs

We describe our procedure for learning the transition costs θ. Our training data $\mathcal{D}$ consists of query facts q and their associated gold truth values y. Equation (2) gives us a probability that a particular inference is *valid*; we axiomatically consider a valid inference from a known premise to be justification for the truth of the query. This is at the expense of the (often incorrect) assumption that our database is clean and only contains true facts.

We optimize the likelihood of our gold annotations according to this probability, subject to the constraint that all elements in our cost vector θ be non-negative. We run the search algorithm described in Section 4 on every query $q_i \in \mathcal{D}$. This produces the highest confidence path x_1, along with its inference state v_i. We now have annotated tuples: $((x_i, v_i), y_i)$ for every element in our training set. Analogous to logistic regression, the log likelihood of our training data $\mathcal{D}$, subject to costs θ, is:

$$l_\theta(\mathcal{D}) = \sum_{0 \le i < |\mathcal{D}|} \Big[y_i \log \left(\frac{v_i}{2} + \frac{1}{1 + e^{v_i \theta \cdot \mathbf{f}(x_i)}} \right) + (1 - y_i) \log \left(\frac{-v_i}{2} + \frac{1}{1 + e^{-v_i \theta \cdot \mathbf{f}(x_i)}} \right) \Big],$$

where y_i is 1 if the example is annotated true and 0 otherwise, and $\mathbf{f}(x_i)$ are the features extracted for path x_i. The objective function is the negative log likelihood with an L_2 regularization term and a log barrier function to prohibit negative costs:

$$O(\mathcal{D}) = -l_\theta(\mathcal{D}) + \frac{1}{2\sigma^2} \|\theta\|_2^2 - \epsilon \log(\theta).$$

We optimize this objective using conjugate gradient descent. Although the objective is non-convex, in practice we can find a good initialization of weights to reduce the risk of arriving at local optima.

An elegant property of this formulation is that the weights we are optimizing correspond directly to the costs used during search. This creates a positive feedback loop – as better weights are learned, the search algorithm is more likely to find confident paths, and more data is available to train from. We therefore run this learning step for multiple epochs, re-running search after each epoch. The weights for the first epoch are initialized to an approximation of valid Natural Logic weights. Subsequent epochs initialize their weights to the output of the previous epoch.

§	Category	Count	Precision		Recall		Accuracy		
			N	M08	N	M08	N	M07	M08
1	**Quantifiers**	**44**	**91**	**95**	**100**	**100**	**95**	**84**	**97**
2	Plurals	24	80	90	29	64	38	42	75
3	Anaphora	6	100	100	20	60	33	50	50
4	Ellipses	25	100	100	5	5	28	28	24
5	**Adjectives**	**15**	**80**	**71**	**66**	**83**	**73**	**60**	**80**
6	**Comparatives**	**16**	**90**	**88**	**100**	**89**	**87**	**69**	**81**
7	Temporal	36	75	86	53	71	52	61	58
8	Verbs	8	–	80	0	66	25	63	62
9	Attitudes	9	–	100	0	83	22	55	89
Applicable (1,5,6)		**75**	**89**	**89**	**94**	**94**	**89**	**76**	**90**

Table 4: Results on the FraCaS textual entailment suite. N is this work; M07 refers to MacCartney and Manning (2007); M08 refers to MacCartney and Manning (2008). The relevant sections of the corpus intended to be handled by this system are sections 1, 5, and 6 (although not 2 and 9, which are also included in M08).

6 Experiments

We evaluate our system on two tasks: the FraCaS test suite, used by MacCartney and Manning (2007; 2008), evaluates the system's ability to capture Natural Logic inferences even without the explicit alignments of these previous systems. In addition, we evaluate the system's ability to predict common-sense facts from a large corpus of OpenIE extractions.

6.1 FraCaS Entailment Corpus

The FraCaS corpus (Cooper et al., 1996) is a small corpus of entailment problems, aimed at providing a comprehensive test of a system's handling of various entailment patterns. We process the corpus following MacCartney and Manning (2007). It should be noted that many of the sections of the corpus are not directly applicable to Natural Logic inferences; MacCartney and Manning (2007) identify three sections which are in the scope of their system, and consequently our system as well.

Results on the dataset are given in Table 4.

System	P	R	F_1	Accuracy
Lookup	100.0	12.1	21.6	56.0
NaturalLI Only	88.8	40.1	55.2	67.5
NaturalLI + Lookup	90.6	49.1	63.7	72.0

Table 5: Accuracy inferring common-sense facts on a balanced test set. *Lookup* queries the lemmatized lower-case fact directly in the 270M fact database. *NaturalLI Only* disallows such lookups, and infers every query from only distinct premises in the database. *NaturalLI + Lookup* takes the union of the two systems.

Since the corpus is not a blind test set, the results are presented less as a comparison of performance, but rather to validate the expressive power of our search-based approach against MacCartney's align-and-classify approach. For the experiments, costs were set to express valid Natural Logic inference as a hard constraint.

The results show that the system is able to capture Natural Logic inferences with similar accuracy to the state-of-the-art system of MacCartney and Manning (2008). Note that our system is comparatively crippled in this framework along at least two dimensions: It cannot appeal to the premise when constructing the search, leading to the introduction of a class of search errors which are entirely absent from prior work. Second, the derivation process itself does not have access to the full parse tree of the candidate fact.

Although precision is fairly high even on the non-applicable sections of FraCaS, recall is significantly lower than prior work. This is a direct consequence of not having alignments to appeal to. For instance, we can consider two inferences:

Jack saw Jill is playing $\overset{?}{\Rightarrow}$ *Jill is playing*
Jill saw Jack is playing $\overset{?}{\Rightarrow}$ *Jill is playing*

It is clear from the parse of the sentence that the first is valid and the second is not; however, from the perspective of the search algorithm both make the same two edits: inserting *Jack* and *saw*. In order to err on the side of safety, we disallow deleting the verb *saw*.

6.2 Common Sense Reasoning

We validate our system's ability to infer unseen common sense facts from a large database of such facts. Whereas evaluation on FraCaS shows that our search formulation captures applicable inferences as well as prior work, this evaluation presents a novel use-case for Natural Logic inference.

For our database of facts, we run the Ollie OpenIE system (Mausam et al., 2012) over Wikipedia,[3] Simple Wikipedia,[4] and a random 5% of CommonCrawl. Extractions with confidence below 0.25 or which contained pronouns were discarded. This yielded a total of 305 million unique extractions composed entirely of lexical items which mapped into our vocabulary (186 707 items). Each of these extracted triples (e_1, r, e_2) was then flattened into a plain-text fact $e_1\ r\ e_2$ and lemmatized. This yields 270 million unique lemmatized premises in our database. In general, each fact in the database could be arbitrary unstructured text; our use of Ollie extractions is motivated only by a desire to extract short, concise facts.

For our evaluation, we infer the top 689 most confident facts from the ConceptNet project (Tandon et al., 2011). To avoid redundancy with WordNet, we take facts from eight ConceptNet relations: MemberOf, HasA, UsedFor, CapableOf, Causes, HasProperty, Desires, and CreatedBy. We then treat the *surface text* field of these facts as our candidate query. This yields queries like the following:

not all birds can fly
noses are used to smell
nobody wants to die
music is used for pleasure

For negative examples, we take the 689 ReVerb extractions (Fader et al., 2011) judged as false by Mechanical Turk workers (Angeli and Manning, 2013). This provides a set of plausible but nonetheless incorrect examples, and ensures that our recall is not due to over-zealous search. Search costs are tuned from an additional set of 540 true ConceptNet and 540 false ReVerb extractions.

Results are shown in Table 5. We compare against the baseline of looking up each fact verbatim in the fact database. Note that both the query and the facts in the database are short snippets, already lemmatized and lower-cased; therefore, it is not in principle unreasonable to expect a database of 270 million extractions to contain these facts. Nonetheless, only 12% of facts were found via a direct lookup. We show that NaturalLI (allowing lookups) improves this recall four-fold, at only an 9.4% drop in precision.

[3] `http://wikipedia.org/` (2013-07-03)
[4] `http://simple.wikipedia.org/` (2014-03-25)

7 Related Work

A large body of work is devoted to compiling open-domain knowledge bases. For instance, OpenIE systems (Yates et al., 2007; Fader et al., 2011) extract concise facts via surface or dependency patterns. In a similar vein, NELL (Carlson et al., 2010; Gardner et al., 2013) continuously learns new high-precision facts from the internet.

Many NLP applications query large knowledge bases. Prominent examples include question answering (Voorhees, 2001), semantic parsing (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2013; Berant and Liang, 2014), and information extraction systems (Hoffmann et al., 2011; Surdeanu et al., 2012). A goal of this work is to improve accuracy on these downstream tasks by providing a *probabilistic* knowledge base for likely true facts.

A natural alternative to the approach taken in this paper is to extend knowledge bases by inferring and adding new facts directly. For instance, Snow et al. (2006) present an approach to enriching the WordNet taxonomy; Tandon et al. (2011) extend ConceptNet with new facts; Soderland et al. (2010) use ReVerb extractions to enrich a domain-specific ontology. Chen et al. (2013) and Socher et al. (2013) use Neural Tensor Networks to predict unseen relation triples in WordNet and Freebase, following a line of work by Bordes et al. (2011) and Jenatton et al. (2012). Yao et al. (2012) and Riedel et al. (2013) present a related line of work, inferring new relations between Freebase entities via inference over both Freebase and OpenIE relations. In contrast, this work runs inference over arbitrary text, without restricting itself to a particular set of relations, or even entities.

The goal of tackling common-sense reasoning is by no means novel in itself. Work by Reiter and McCarthy (Reiter, 1980; McCarthy, 1980) attempts to reason about the truth of a consequent in the absence of strict logical entailment. Similarly, Pearl (1989) presents a framework for assigning confidences to inferences which can be reasonably assumed. Our approach differs from these attempts in part in its use of Natural Logic as the underlying inference engine, and more substantially in its attempt at creating a broad-coverage system. More recently, work by Schubert (2002) and Van Durme et al. (2009) approach common sense reasoning with *episodic logic*; we differ in our focus on inferring truth from an arbitrary query, and in making use of longer inferences.

This work is similar in many ways to work on recognizing textual entailment – e.g., Schoenmackers et al. (2010), Berant et al. (2011). Work by Lewis and Steedman (2013) is particularly relevant, as they likewise evaluate on the FraCaS suite (Section 1; 89% accuracy with gold trees). They approach entailment by constructing a CCG parse of the query, while mapping questions which are paraphrases of each other to the same logical form using distributional relation clustering. However, their system is unlikely to scale to either our large database of premises, or our breadth of relations.

Fader et al. (2014) propose a system for question answering based on a sequence of paraphrase rewrites followed by a fuzzy query to a structured knowledge base. This work can be thought of as an elegant framework for unifying this two-stage process, while explicitly tracking the "risk" taken with each paraphrase step. Furthermore, our system is able to explore mutations which are only valid in one direction, rather than the bidirectional entailment of paraphrases, and does not require a corpus of such paraphrases for training.

8 Conclusion

We have presented NaturalLI, an inference system over unstructured text intended to infer common sense facts. We have shown that we can run inference over a large set of premises while maintaining Natural Logic semantics, and that we can learn how to infer unseen common sense facts.

Future work will focus on enriching the class of inferences we can make with Natural Logic. For example, extending the approach to handle meronymy and relation entailments. Furthermore, we hope to learn richer lexicalized parameters, and use the syntactic structure of a fact during search.

Acknowledgements

We thank the anonymous reviewers for their thoughtful comments. We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Gabor Angeli and Christopher Manning. 2013. Philosophers are mortal: Inferring the truth of unseen facts. In *CoNLL*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. *Proc. Linguistic Annotation Workshop*.

J. Berant and P. Liang. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of ACL*, Portland, OR.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *AAAI*.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. *Linguistic Data Consortium*.

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

Danqi Chen, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2013. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. *arXiv preprint arXiv:1301.3618*.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. Using the framework. Technical report, The FraCaS Consortium.

Alex J Djalali. 2013. Synthetic logic. *Linguistic Issues in Language Technology*, 9:1–18.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *KDD*.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. *EMNLP*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL-HLT*.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. *ACL*.

Thomas Icard, III and Lawrence Moss. 2014. Recent progress on monotonicity. *Linguistic Issues in Language Technology*.

Thomas Icard, III. 2012. Inclusion and exclusion in natural language. *Studia Logica*.

Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. 2012. A latent factor model for highly multi-relational data. In *NIPS*.

Jay J Jiang and David W Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. *Proceedings of the 10th International Conference on Research on Computational Linguistics*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *TACL*, 1:179–192.

Bill MacCartney and Christopher D Manning. 2007. Natural logic for textual inference. In *ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Bill MacCartney and Christopher D Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Coling*.

Bill MacCartney and Christopher D Manning. 2009. An extended model of natural logic. In *Proceedings of the eighth international conference on computational semantics*.

Bill MacCartney. 2009. *Natural Language Inference*. Ph.D. thesis, Stanford.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *EMNLP*.

John McCarthy. 1980. Circumscription—a form of non-monotonic reasoning. *Artificial intelligence*.

Judea Pearl. 1989. Probabilistic semantics for non-monotonic reasoning: A survey. *Principles of Knowledge Representation and Reasoning*.

Raymond Reiter. 1980. A logic for default reasoning. *Artificial intelligence*, 13(1):81–132.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL-HLT*.

Stefan Schoenmackers, Oren Etzioni, Daniel S Weld, and Jesse Davis. 2010. Learning first-order horn clauses from web text. In *EMNLP*.

Lenhart Schubert. 2002. Can we derive general world knowledge from texts? In *HLT*.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *ACL*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*.

Stephen Soderland, Brendan Roof, Bo Qin, Shi Xu, Oren Etzioni, et al. 2010. Adapting open information extraction to domain-specific relations. *AI Magazine*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP*.

Niket Tandon, Gerard de Melo, and Gerhard Weikum. 2011. Deriving a web-scale common sense fact database. In *AAAI*.

Víctor Manuel Sánchez Valencia. 1991. *Studies on natural logic and categorial grammar*. Ph.D. thesis, University of Amsterdam.

Johan van Benthem. 1986. *Essays in logical semantics*. Springer.

Johan van Benthem. 2008. A brief history of natural logic. Technical Report PP-2008-05, University of Amsterdam.

Benjamin Van Durme, Phillip Michalak, and Lenhart K Schubert. 2009. Deriving generalized knowledge from corpora using wordnet abstraction. In *EACL*.

Ellen M Voorhees. 2001. Question answering in TREC. In *Proceedings of the tenth international conference on Information and knowledge management*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Probabilistic databases of universal schema. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. TextRunner: Open information extraction on the web. In *ACL-HLT*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, Portland, OR.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *EMNLP-CoNLL*.

Modeling Term Translation for Document-informed Machine Translation

Fandong Meng[1, 2] **Deyi Xiong**[3] **Wenbin Jiang**[1, 2] **Qun Liu**[4, 1]
[1]Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
{mengfandong,jiangwenbin,liuqun}@ict.ac.cn
[3]School of Computer Science and Technology, Soochow University
dyxiong@suda.edu.cn
[4]Centre for Next Generation Localisation, Dublin City University

Abstract

Term translation is of great importance for statistical machine translation (SMT), especially document-informed SMT. In this paper, we investigate three issues of term translation in the context of document-informed SMT and propose three corresponding models: (a) a term translation disambiguation model which selects desirable translations for terms in the source language with domain information, (b) a term translation consistency model that encourages consistent translations for terms with a high strength of translation consistency throughout a document, and (c) a term bracketing model that rewards translation hypotheses where bracketable source terms are translated as a whole unit. We integrate the three models into hierarchical phrase-based SMT and evaluate their effectiveness on NIST Chinese-English translation tasks with large-scale training data. Experiment results show that all three models can achieve significant improvements over the baseline. Additionally, we can obtain a further improvement when combining the three models.

1 Introduction

A term is a linguistic expression that is used as the designation of a defined concept in a language (ISO 1087). As terms convey concepts of a text, term translation becomes crucial when the text is translated from its original language to another language. The translations of terms are often affected by the domain in which terms are used and the context that surrounds terms (Vasconcellos et al., 2001). In this paper, we study domain-specific and context-sensitive term translation for SMT.

In order to achieve this goal, we focus on three issues of term translation: 1) translation ambiguity, 2) translation consistency and 3) bracketing. First, term translation ambiguity is related to translations of the same term in different domains. A source language term may have different translations when it occurs in different domains. Second, translation consistency is about consistent translations for terms that occur in the same document. Usually, it is undesirable to translate the same term in different ways as it occurs in different parts of a document. Finally, bracketing concerns whether a multi-word term is bracketable during translation. Normally, a multi-word term is translated as a whole unit into a contiguous target string.

We study these three issues in the context of document-informed SMT. We use document-informed information to disambiguate term translations in different documents and maintain consistent translations for terms that occur in the same document. We propose three different models for term translation that attempt to address the three issues mentioned above. In particular,

- *Term Translation Disambiguation Model:* In this model, we condition the translations of terms in different documents on corresponding per-document topic distributions. In doing so, we enable the decoder to favor translation hypotheses with domain-specific term translations.
- *Term Translation Consistency Model:* This model encourages the same terms with a high strength of translation consistency that occur in different parts of a document to be translated in a consistent fashion. We calculate the translation consistency strength of a term based on the topic distribution of the documents where the term occurs in this model.
- *Term Bracketing Model:* We use the bracketing model to reward translation hypothe-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 546–556,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

ses where bracketable multi-word terms are translated as a whole unit.

We integrate the three models into hierarchical phrase-based SMT (Chiang, 2007). Large-scale experiment results show that they are all able to achieve significant improvements of up to 0.89 BLEU points over the baseline. When simultaneously integrating the three models into SMT, we can gain a further improvement, which outperforms the baseline by up to 1.16 BLEU points.

In the remainder of this paper, we begin with a brief overview of related work in Section 2, and bilingual term extraction in Section 3. We then elaborate the proposed three models for term translation in Section 4. Next, we conduct experiments to validate the effectiveness of the proposed models in Section 5. Finally, we conclude and provide directions for future work in Section 6.

2 Related Work

In this section, we briefly introduce related work and highlight the differences between our work and previous studies.

As we approach term translation disambiguation and consistency via topic modeling, our models are related to previous work that explores the topic model (Blei et al., 2003) for machine translation (Zhao and Xing, 2006; Su et al., 2012; Xiao et al., 2012; Eidelman et al., 2012). Zhao and Xing (2006) employ three models that enable word alignment process to leverage topical contents of document-pairs with topic model. Su et al. (2012) establish the relationship between out-of-domain bilingual corpus and in-domain monolingual corpora via topic mapping and phrase-topic distribution probability estimation for translation model adaptation. Xiao et al. (2012) propose a topic similarity model for rule selection. Eidelman et al. (2012) use topic models to adapt lexical weighting probabilities dynamically during translation. In these studies, the topic model is not used to address the issues of term translation mentioned in Section 1.

Our work is also related to document-level SMT in that we use document-informed information for term translation. Tiedemann (2010) propose cache-based language and translation models, which are built on recently translated sentences. Gong et al. (2011) extend this by further introducing two additional caches. They employ a static cache to store bilingual phrases extracted from documents in training data that are similar to the document being translated and a topic cache with target language topic words. Recently we have also witnessed efforts that model lexical cohesion (Hardmeier et al., 2012; Wong and Kit, 2012; Xiong et al., 2013a; Xiong et al., 2013b) as well as coherence (Xiong and Zhang, 2013) for document-level SMT. Hasler et al. (2014a) use topic models to learn document-level translation probabilities. Hasler et al. (2014b) use topic-adapted model to improve lexical selection. The significant difference between our work and these studies is that term translation has not been investigated in these document-level SMT models.

Itagaki and Aikawa (2008) employ bilingual term bank as a dictionary for machine-aided translation. Ren et al. (2009) propose a binary feature to indicate whether a bilingual phrase contains a term pair. Pinis and Skadins (2012) investigate that bilingual terms are important for domain adaptation of machine translation. These studies do not focus on the three issues of term translation as discussed in Section 1. Furthermore, domain and document-informed information is not used to assist term translation.

Itagaki et al. (2007) propose a statistical method to calculate translation consistency for terms with explicit domain information. Partially inspired by their study, we introduce a term translation consistency metric with document-informed information. Furthermore, we integrate the proposed term translation consistency model into an actual SMT system, which has not been done by Itagaki et al. (2007). Ture et al. (2012) use IR-inspired tf-idf scores to encourage consistent translation choice. Guillou (2013) investigates what kind of words should be translated consistently. Term translation consistency has not been investigated in these studies.

Our term bracketing model is also related to Xiong et al. (2009)'s syntax-driven bracketing model for phrase-based translation, which predicts whether a phrase is bracketable or not using rich syntactic constraints. The difference is that we construct the model with automatically created bilingual term bank and do not depend on any syntactic knowledge.

3 Bilingual Term Extraction

Bilingual term extraction is to extract terms from two languages with the purpose of creating or ex-

tending a bilingual term bank, which in turn can be used to improve other tasks such as information retrieval and machine translation. In this paper, we want to automatically build a bilingual term bank so that we can model term translation to improve translation quality of SMT. Our interest is to extract multi-word terms.

Currently, there are mainly two strategies to conduct bilingual term extraction from parallel corpora. One of them is to extract term candidates separately for each language according to monolingual term metrics, such as C-value/NC-value (Frantzi et al., 1998; Vu et al., 2008), or other common cooccurrence measures such as Log-Likelihood Ratio, Dice coefficient and Pointwise Mutual Information (Daille, 1996; Piao et al., 2006). The extracted monolingual terms are then paired together (Hjelm, 2007; Fan et al., 2009; Ren et al., 2009). The other strategy is to align words and word sequences that are translation equivalents in parallel corpora and then classify them into terms and non-terms (Merkel and Foo, 2007; Lefever et al., 2009; Bouamor et al., 2012). In this paper, we adopt the first strategy. In particular, for each sentence pair, we collect all source phrases which are terms and find aligned target phrases for them via word alignments. If the target side is also a term, we store the source and target term as a term pair.

We conduct monolingual term extraction using the C-value/NC-value metric and Log-Likelihood Ratio (LLR) measure respectively. We then combine terms extracted according to the two metrics mentioned above. For the C-value/NC-value metric based term extraction, we implement it in the same way as described in Frantzi et al. (1998). This extraction method recognizes linguistic patterns (mainly noun phrases) listed as follows.

$$((Adj|Noun)^{+}|((Adj|Noun)^{*}$$
$$(NounPrep)^{?})(Adj|Noun)^{*})Noun$$

It captures the linguistic structures of terms. For the LLR metric based term extraction, we implement it according to Daille (1996), who estimate the propensity of two words to appear together as a multi-word expression. We then adopt LLR-based hierarchical reducing algorithm proposed by Ren et al. (2009) to extract terms with arbitrary lengths. Since the C-value/NC-value metric based extraction method can obtain terms in strict linguistic patterns while the LLR measure based method extracts more flexible terms, these two methods are complementary to each other. Therefore, we use these two methods to extract monolingual multi-word terms and then combine the extracted terms.

4 Models

This section presents the three models of term translation. They are the term translation disambiguation model, term translation consistency model and term bracketing model respectively.

4.1 Term Translation Disambiguation Model

The most straightforward way to disambiguate term translations in different domains is to calculate the conditional translation probability of a term given domain information. We use the topic distribution of a document obtained by a topic model to represent the domain information of the document. Since Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is the most widely-used topic model, we exploit it for inferring topic distributions of documents. Xiao et al. (2012) proposed a topic similarity model for rule selection. Different from their work, we take an easier strategy that estimates topic-conditioned term translation probabilities rather than rule-topic distributions. This makes our model easily scalable on large training data.

With the bilingual term bank created from the training data, we calculate the source-to-target term translation probability for each term pair conditioned on the topic distribution of the source document where the source term occurs. We maintain a K-dimension (K is the number of topics) vector for each term pair. The k-th component $p(t_e|t_f, z = k)$ measures the conditional translation probability from source term t_f to target term t_e given the topic k.

We calculate $p(t_e|t_f, z = k)$ via maximum likelihood estimation with counts from training data. When the source part of a bilingual term pair occurs in a document D with topic distribution $p(z|D)$ estimated via LDA tool, we collect an instance $(t_f, t_e, p(z|D), c)$, where c is the fraction count of the instance as described in Chiang (2007). After collection, we get a set of instances $I = \{(t_f, t_e, p(z|D), c)\}$ with different document-topic distributions for each bilingual term pair. Using these instances, we calculate the probability

$p(t_e|t_f, z = k)$ as follows:

$$p(t_e|t_f, z = k) = \frac{\sum_{i \in I, i.t_f = t_f, i.t_e = t_e} i.c * p(z = k|D)}{\sum_{i \in I, i.t_f = t_f} i.c * p(z = k|D)} \quad (1)$$

We associate each extracted term pair in our bilingual term bank with its corresponding topic-conditioned translation probabilities estimated in the Eq. (1). When translating sentences of document $D^{'}$, we first get the topic distribution of $D^{'}$ using LDA tool. Given a sentence which contains T terms $\{t_{f_i}\}_1^T$ in $D^{'}$, our term translation disambiguation model $TermDis$ can be denoted as

$$TermDis = \prod_{i=1}^{T} P_d(t_{e_i}|t_{f_i}, D^{'}) \quad (2)$$

where the conditional source-to-target term translation probability $P_d(t_{e_i}|t_{f_i}, D^{'})$ given the document $D^{'}$ is formulated as follows:

$$P_d(t_{e_i}|t_{f_i}, D^{'}) = \sum_{k=1}^{K} p(t_{e_i}|t_{f_i}, z = k) * p(z = k|D^{'}) \quad (3)$$

Whenever a source term t_{f_i} is translated into t_{e_i}, we check whether the pair of t_{f_i} and its translation t_{e_i} can be found in our bilingual term bank. If it can be found, we calculate the conditional translation probability from t_{f_i} to t_{e_i} given the document $D^{'}$ according to Eq. (3).

The term translation disambiguation model is integrated into the log-linear model of SMT as a feature. Its weight is tuned via minimum error rate training (MERT) (Och, 2003). Through the feature, we can enable the decoder to favor translation hypotheses that contain target term translations appropriate for the domain represented by the topic distribution of the corresponding document.

4.2 Term Translation Consistency Model

The term translation disambiguation model helps the decoder select appropriate translations for terms that are in accord with their domains. Yet another translation issue related to the domain-specific term translation is to what extent a term should be translated consistently given the domain where it occurs. Term translation consistency indicates the translation stability that a source term is translated into the same target term (Itagaki et al., 2007). When translating a source term, if the translation consistency strength of the source term is high, we should take the corresponding target term as the translation for it. Otherwise, we may need to create a new translation for it according to its context. In particular, we want to enable the decoder to choose between: 1) translating a given source term into the extracted corresponding target term or 2) translating it in another way according to the strength of its translation consistency. In doing so, we can encourage consistent translations for terms with a high translation consistency strength throughout a document.

Our term translation consistency model can exactly measure the strength of term translation consistency in a document. Since the essential component of our term translation consistency model is the translation consistency strength of the source term estimated under the topic distribution, we describe how to calculate it before introducing the whole model.

With the bilingual term bank created from training data, we first group each source term and all its corresponding target terms into a 2-tuple $G\langle t_f, Set(t_e)\rangle$, where t_f is the source term and $Set(t_e)$ is the set of t_f's corresponding target terms. We maintain a K-dimension (K is the number of topics) vector for each 2-tuple $G\langle t_f, Set(t_e)\rangle$. The k-th component measures the translation consistency strength $cons(t_f, k)$ of the source term t_f given the topic k.

We calculate $cons(t_f, k)$ for each $G\langle t_f, Set(t_e)\rangle$ with counts from training data as follows:

$$cons(t_f, k) = \sum_{m=1}^{M} \sum_{n=1}^{N_m} (\frac{q_{mn} * p(k|m)}{Q_k})^2 \quad (4)$$

$$Q_k = \sum_{m=1}^{M} \sum_{n=1}^{N_m} q_{mn} * p(k|m) \quad (5)$$

where M is the number of documents in which the source term t_f occurs, N_m is the number of unique corresponding term translations of t_f in the mth document, q_{mn} is the frequency of the nth translation of t_f in the mth document, $p(k|m)$ is the conditional probability of the mth document over topic k, and Q_k is the normalization factor. All translations of t_f are from $Set(t_e)$. We adapt Itagaki et al. (2007)'s translation consistency metric for terms to our topic-based translation consistency measure in the Eq. (4). This equation calculates the translation consistency strength of the source term t_f given the topic k according to the distribution of t_f's translations in each document

where they occur. According to Eq. (4), the translation consistency strength is a score between 0 and 1. If a source term only occurs in a document and all its translations are the same, the translation consistency strength of this term is 1.

We reorganize our bilingual term bank into a list of 2-tuples $G\langle t_f, Set(t_e)\rangle$s, each of which is associated with a K-dimension vector storing the topic-conditioned translation consistency strength calculated in the Eq. (4). When translating sentences of document D, we first get the topic distribution of D via LDA tool. Given a sentence which contains T terms $\{t_{f_i}\}_1^T$ in D, our term translation consistency model $TermCons$ can be denoted as

$$TermCons = \prod_{i=1}^{T} exp(S_c(t_{f_i}|D)) \quad (6)$$

where the strength of translation consistency for t_{f_i} given the document D is formulated as follows:

$$S_c(t_{f_i}|D) = log(\sum_{k=1}^{K} cons(t_{f_i}, k) * p(k|D)) \quad (7)$$

During decoding, whenever a hypothesis just translates a source term t_{f_i} into t_e, we check whether the translation t_e can be found in $Set(t_e)$ of t_{f_i} from the reorganized bilingual term bank. If it can be found, we calculate the strength of translation consistency for t_{f_i} given the document D according to Eq. (7) and take it as a soft constraint. If the $S_c(t_{f_i}|D)$ of t_{f_i} is high, the decoder should translate t_{f_i} into the extracted corresponding target terms. Otherwise, the decoder will select translations from outside of $Set(t_e)$ for t_{f_i}. In doing so, we encourage terms to be translated in a topic-dependent consistency pattern in the test data similar to that in the training data so that we can control the translation consistency of terms in the test data.

The term translation consistency model is also integrated into the log-linear model of SMT as a feature. Through the feature, we can enable the decoder to translate terms with a high translation consistency in a document into corresponding target terms from our bilingual term bank rather than other translations in a consistent fashion.

4.3 Term Bracketing Model

The term translation disambiguation model and consistency model concern the term translation accuracy with domain information. We further propose a term bracketing model to guarantee the integrality of term translation. Xiong et al. (2009) proposed a syntax-driven bracketing model for phrase-based translation, which predicts whether a phrase is bracketable or not using rich syntactic constraints. If a source phrase remains contiguous after translation, they refer to this type of phrase as bracketable phrase, otherwise unbracketable phrase. For multi-word terms, it is also desirable to be bracketable since a source term should be translated as a whole unit and its translation should be contiguous.

In this paper, we adapt Xiong et al. (2009)'s bracketing approach to term translation and build a classifier to measure the probability that a source term should be translated in a bracketable manner. For all source parts of the extracted bilingual term bank, we find their target counterparts in the word-aligned training data. If the corresponding target counterpart remains contiguous, we take the source term as a bracketable instance, otherwise an unbracketable instance. With these bracketable and unbracketable instances, we train a maximum entropy binary classifier to predict bracketable (b) probability of a given source term t_f within particular contexts $c(t_f)$. The binary classifier is formulated as follows:

$$P_b(b|c(t_f)) = \frac{exp(\sum_j \theta_j h_j(b, c(t_f)))}{\sum_{b'} exp(\sum_j \theta_j h_j(b', c(t_f)))} \quad (8)$$

where $h_j \in \{0, 1\}$ is a binary feature function and θ_j is the weight of h_j. We use the following features: 1) the word sequence of the source term, 2) the first word of the source term, 3) the last word of the source term, 4) the preceding word of the first word of the source term, 5) the succeeding word of the last word of the source term, and 6) the number of words in the source term.

Given a source sentence which contains T terms $\{t_{f_i}\}_1^T$, our term bracketing model $TermBrack$ can be denoted as

$$TermBrack = \prod_{i=1}^{T} P_b(b|c(t_{f_i})) \quad (9)$$

Whenever a hypothesis just covers a source term t_{f_i}, we calculate the bracketable probability of t_{f_i} according to Eq. (8).

The term bracketing model is integrated into the log-linear model of SMT as a feature. Through the feature, we want the decoder to translate source terms with a high bracketable probability as a whole unit.

Source	Target	D	M
Fángyù Xìtǒng	defence mechanisms	470	56
Fángyù Xìtǒng	defence systems		
Fángyù Xìtǒng	defense programmes		
Fángyù Xìtǒng	prevention systems		
...	...		
Zhànluè Dǎodàn Fángyù Xìtǒng	strategic missile defense system	7	0

Table 1: Examples of bilingual terms extracted from the training data. "D" means the total number of documents in which the corresponding source term occurs and "M" denotes the number of documents in which the corresponding source term is translated into different target terms. The source side is Chinese Pinyin. To save space, we do not list all the 23 different translations of the source term "Fángyù Xìtǒng".

5 Experiments

In this section, we conducted experiments to answer the following three questions.

1. Are our term translation disambiguation, consistency and bracketing models able to improve translation quality in BLEU?
2. Does the combination of the three models provide further improvements?
3. To what extent do the proposed models affect the translations of test sets?

5.1 Setup

Our training data consist of 4.28M sentence pairs extracted from LDC[1] data with document boundaries explicitly provided. The bilingual training data contain 67,752 documents, 124.8M Chinese words and 140.3M English words. We chose NIST MT05 as the MERT (Och, 2003) tuning set, NIST MT06 as the development test set, and NIST MT08 as the final test set. The numbers of documents/sentences in NIST MT05, MT06 and MT08 are 100/1082, 79/1664 and 109/1357 respectively.

The word alignments were obtained by running GIZA++ (Och and Ney, 2003) on the corpora in both directions and using the "grow-diag-final-and" balance strategy (Koehn et al., 2003). We adopted SRI Language Modeling Toolkit (Stolcke and others, 2002) to train a 4-gram language model with modified Kneser-Ney smoothing on the Xinhua portion of the English Gigaword corpus. For the topic model, we used the open source LDA tool GibbsLDA++[2] with the default setting for training and inference. We performed 100 iterations of the L-BFGS algorithm implemented in the MaxEnt toolkit[3] with both Gaussian prior and event cutoff set to 1 to train the term bracketing prediction model (Section 4.3).

We performed part-of-speech tagging for monolingual term extraction (C-value/NC-vaule method in Section 3) of the source and target languages with the Stanford NLP toolkit[4]. The bilingual term bank was extracted based on the following parameter settings of term extraction methods. Empirically, we set the maximum length of a term to 6 words[5]. For both the C-value/NC-value and LLR-based extraction methods, we set the context window size to 5 words, which is a widely-used setting in previous work. And we set C-value/NC-value score threshold to 0 and LLR score threshold to 10 according to the training corpora.

We used the case-insensitive 4-gram BLEU[6] as our evaluation metric. In order to alleviate the impact of the instability of MERT (Och, 2003), we ran it three times for all our experiments and presented the average BLEU scores on the three runs following the suggestion by Clark et al. (2011).

We used an in-house hierarchical phrase-based decoder to verify our proposed models. Although the decoder translates a document in a sentence-by-sentence fashion, it incorporates document-informed information for sentence translation via the proposed term translation models trained on documents.

[1]The corpora include LDC2003E07, LDC2003E14, LDC2004T07, LDC2004E12, LDC2005E83, LDC2005T06, LDC2005T10, LDC2006E24, LDC2006E34, LDC2006E85, LDC2006E92, LDC2007E87, LDC2007E101, LDC2008E40, LDC2008E56, LDC2009E16 and LDC2009E95.

[2]http://sourceforge.net/projects/gibbslda/

[3]http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html

[4]http://nlp.stanford.edu/software/tagger.shtml

[5]We determine the maximum length of a term by testing {5, 6, 7, 8} in our preliminary experiments. We find that length 6 produces a slightly better performance than other values.

[6]ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl

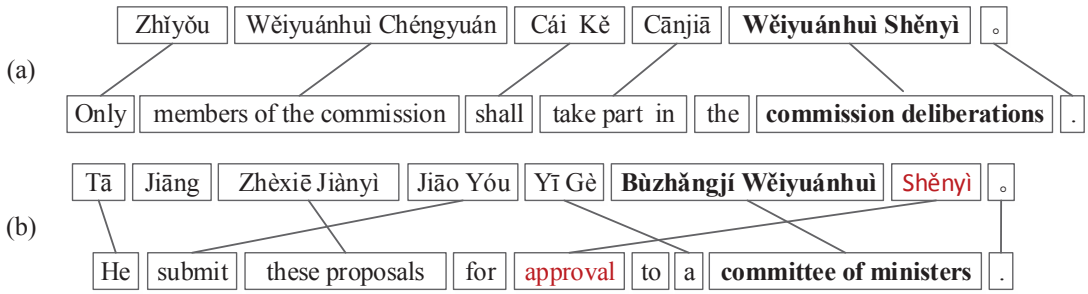


Figure 1: An example of unbracketable source term in the training data. In (a), "Wěiyuánhuì Shěnyì" is bracketable while in (b) it is unbracketable. The solid lines connect bilingual phrases. The source side is Chinese Pinyin.

5.2 Bilingual Term Bank

Before reporting the results of the proposed models, we provide some statistics of the bilingual term bank extracted from the training data.

According to our statistics, about 1.29M bilingual terms are extracted from the training data. 65.07% of the sentence pairs contain bilingual terms in the training data. And on average, a source term has about 1.70 different translations. These statistics indicate that terms are frequently used in real-world data and that a source term can be translated into different target terms.

We also present some examples of bilingual terms extracted from the training data in Table 1. Accordingly, we show the total number of documents in which the corresponding source term occurs and the number of documents in which the corresponding source term is translated into different target terms. The source term "Fángyù Xìtǒng" has 23 different translations in total. They are distributed in 470 documents in the training data. In 414 documents, "Fángyù Xìtǒng" has only one single translation. However, in the other 56 documents it has different translations. This indicates that "Fángyù Xìtǒng" is not consistently translated in these 56 documents. Different from this, the source term "Zhànluè Dǎodàn Fángyù Xìtǒng" only has one translation. And it is translated consistently in all 7 documents where it occurs. In fact, according to our statistics, there are about 5.19% source terms whose translations are not consistent even in the same document.

These examples and statistics suggest 1) that source terms have domain-specific translations and 2) that source terms are not necessarily translated in a consistent manner even in the same document. These are exactly the reasons why we propose the term translation disambiguation and consistency model based on domain information represented by topic distributions.

Actually, 36.13% of the source terms are not necessarily translated into target strings as a whole unit. We show an example of such terms in Figure 1. In Figure 1-(a), "Wěiyuánhuì Shěnyì" is a term, and is translated into "commission deliberations" as a whole unit. Therefore "Wěiyuánhuì Shěnyì" is bracketable in this sentence. However, in Figure 1-(b), "Wěiyuánhuì" and "Shěnyì" are translated separately. Therefore "Wěiyuánhuì Shěnyì" is an unbracketable term in this sentence. This is the reason why we propose a bracketing model to predict whether a source term is bracketable or not.

5.3 Effect of the Proposed Models

In this section, we validate the effectiveness of the proposed term translation disambiguation model, consistency model and bracketing model respectively. In addition to the traditional hiero (Chiang, 2007) system, we also compare against the "CountFeat" method in Ren et al. (2009) who use a binary feature to indicate whether a bilingual phrase contains a term pair. Although Ren et al. (2009)'s experiments are conducted in a phrase-based system, the idea can be easily applied to a hierarchical phrase-based system.

We carried out experiments to investigate the effect of the term translation disambiguation model (Dis-Model) and report the results in Table 2. In order to find the topic number setting with which our model has the best performance, we ran experiments using the MT06 as the development test set. From Table 2, we observe that the Dis-Model obtains steady improvements over the baseline and "CountFeat" method with the topic number K

Models		MT06	MT08	Avg
Baseline		32.43	24.14	28.29
CountFeat		32.77	24.29	28.53
Dis-Model	$K = 50$	32.94*	24.53	28.74
	$K = 100$	33.10*	24.57	28.84
	$K = 150$	**33.16***	**24.67***	**28.92**
	$K = 200$	33.08*	24.55	28.81
Cons-Model	$K = 50$	33.09*	24.59	28.84
	$K = 100$	33.13*	24.74*	28.94
	$K = 150$	**33.32*+**	**24.84*+**	**29.08**
	$K = 200$	33.02*	24.73*	28.88
Brack-Model		**33.09***	**24.66***	**28.88**
Combined-Model		**33.59*+**	**24.99*+**	**29.29**

Table 2: BLEU-4 scores (%) of the term translation disambiguation model (Dis-Model), the term translation consistency model (Cons-Model), the term bracketing model (Brack-Model), and the combination of the three models, on the development test set MT06 and the final test set MT08. $K \in \{50, 100, 150, 200\}$ which is the number of topics for the Dis-Model and the Cons-Model. "Combined-Model" is the combination of the three single modes with topic number 150 for the Dis-Model and the Cons-Model. "Baseline" is the traditional hierarchical phrase-based system. "CountFeat" is the method that adds a counting feature to reward translation hypotheses containing bilingual term pairs. The "*" and "+" denote that the results are significantly (Clark et al., 2011) better than those of the baseline system and the CountFeat method respectively ($p<0.01$).

ranging from 50 to 150. However, when we set K to 200, the performance drops. The highest BLEU scores 33.16 and 24.67 are obtained at the topic setting $K = 150$. In fact, our Dis-Model gains higher performance in BLEU than both the traditional hiero baseline and the "CountFeat" method with all topic settings. The "CountFeat" method rewards translation hypotheses containing bilingual term pairs. However it does not explore any domain information. Our Dis-Model incorporates domain information to conduct translation disambiguation and achieves higher performance. When the topic number is set to 150, we gain the highest BLEU score, which is higher than that of the baseline by 0.73 and 0.53 BLEU points on MT06 and MT08, respectively. The final gain over the baseline is on average 0.63 BLEU points.

We conducted the second group of experiments to study whether the term translation consistency model (Cons-Model) is able to improve the performance in BLEU, as well as to investigate the impact of different topic numbers on the Cons-Model. Results are shown in Table 2, from which we observe the similar phenomena to what we have found in the Dis-Model. Our Cons-Model gains higher BLEU scores than the baseline system and the "CountFeat" method with all topic settings. Setting topic number to 150 achieves the highest BLEU score, which is higher than baseline by 0.89 BLEU points and 0.70 BLEU points on MT06 and MT08 respectively, and on average 0.79 BLEU points.

We also conducted experiments to verify the effectiveness of the term bracketing model (Brack-Model), which conducts bracketing prediction for source terms. Results in Table 2 show that our Brack-Model gains higher BLEU scores than those of the baseline system and the "CountFeat" method. The final gain of Brack-Model over the baseline is 0.66 BLEU points and 0.52 points on MT06 and MT08 respectively, and on average 0.59 BLEU points.

5.4 Combination of the Three Models

As shown in the previous subsection, the term translation disambiguation model, consistency model and bracketing model substantially outperform the baseline. Now, we investigate whether using these three models simultaneously can lead to further improvements. The last row in Table 2 shows that the combination of the three models (Combined-Model) achieves higher BLEU score than all single models, when we set the topic number to 150 for the term translation disambiguation model and consistency model. The final gain

Models	MT06	MT08
Best-Dis-Model	30.89	30.14
Best-Cons-Model	38.04	36.70
Brack-Model	60.46	55.78
Combined-Model	54.39	50.85

Table 3: Percentage (%) of 1-best translations which are generated by the Combined-Model and the three single models with best settings on the development test set MT06 and the final test set MT08. The topic number is 150 for Best-Dis-Model and Best-Cons-Model.

of the Combined-Model over the baseline is 1.16 BLEU points and 0.85 points on MT06 and MT08 respectively, and on average 1.00 BLEU points.

5.5 Analysis

In this section, we investigate to what extent the proposed models affect the translations of test sets. In Table 3, we show the percentage of 1-best translations affected by the Combined-Model and the three single models with best settings on test sets MT06 and MT08. For single models, if the corresponding feature (disambiguation, consistency or bracketing) is activated in the 1-best derivation, the corresponding model has impact on the 1-best translation. For the Combined-Model, if any of the corresponding features is activated in the 1-best derivation, the Combined-Model affects the 1-best translation.

From Table 3, we can see that 1-best translations of source sentences affected by any of the proposed models account for a high proportion (30%∼60%) on both MT06 and MT08. This indicates that all proposed models play an important role in the translation of both test sets. Among the three proposed models, the Brack-Model is the one that affects the largest number of 1-best translations in both test sets. And the percentage is 60.46% and 55.78% on MT06 and MT08 respectively. The Brack-Model only considers source terms during decoding, while the Dis-Model and Cons-Model need to match both source and target terms. The Brack-Model is more likely to be activated. Hence the percentage of 1-best translations affected by this model is higher than those of the other two models. Since we only investigate the 1-best translations generated by the Combined-Model and single models, the translations generated by some single models (e.g., Brack-Model) may not be generated by the Combined-Model. Therefore it is hard to say that the numbers of 1-best translations affected by the Combined-Model must be greater than those of single models.

6 Conclusion and Future Work

We have studied the three issues of term translation and proposed three different term translation models for document-informed SMT. The term translation disambiguation model enables the decoder to favor the most suitable domain-specific translations with domain information for source terms. The term translation consistency model encourages the decoder to translate source terms with a high domain translation consistency strength into target terms rather than other new strings. Finally, the term bracketing model rewards hypotheses that translate bracketable terms into continuous target strings as a whole unit. We integrate the three models into a hierarchical phrase-based SMT system[7] and evaluate their effectiveness on the NIST Chinese-English translation task with large-scale training data. Experiment results show that all three models achieve significant improvements over the baseline. Additionally, combining the three models achieves a further improvement. For future work, we would like to evaluate our models on term translation across a range of different domains.

Acknowledgments

This work was supported by National Key Technology R&D Program (No. 2012BAH39B03) and CAS Action Plan for the Development of Western China (No. KGZD-EW-501). Deyi Xiong's work was supported by Natural Science Foundation of Jiangsu Province (Grant No. BK20140355). Qun Liu's work was partially supported by Science Foundation Ireland (Grant No. 07/CE/I1142) as part of the CNGL at Dublin City University. Sincere thanks to the anonymous reviewers for their thorough reviewing and valuable suggestions. The corresponding author of this paper, according to the meaning given to this role by University of Chinese Academy of Sciences and Soochow University, is Deyi Xiong.

[7] Our models are not limited to hierarchical phrase-based SMT. They can be easily applied to other SMT formalisms, such as phrase- and syntax-based SMT.

References

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Houda Bouamor, Aurélien Max, and Anne Vilnat. 2012. Validation of sub-sentential paraphrases acquired from parallel monolingual corpora. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 716–725. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Jonathan H Clark, Chris Dyer, Alon Lavie, and Noah A Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 176–181.

Béatrice Daille. 1996. Study and implementation of combined techniques for automatic extraction of terminology. *Journal of The balancing act: Combining symbolic and statistical approaches to language*, 1:49–66.

Vladimir Eidelman, Jordan Boyd-Graber, and Philip Resnik. 2012. Topic models for dynamic translation model adaptation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 115–119. Association for Computational Linguistics.

Xiaorong Fan, Nobuyuki Shimizu, and Hiroshi Nakagawa. 2009. Automatic extraction of bilingual terms from a chinese-japanese parallel corpus. In *Proceedings of the 3rd International Universal Communication Symposium*, pages 41–45. ACM.

Katerina T Frantzi, Sophia Ananiadou, and Junichi Tsujii. 1998. The c-value/nc-value method of automatic recognition for multi-word terms. In *Research and Advanced Technology for Digital Libraries*, pages 585–604. Springer.

Zhengxian Gong, Min Zhang, and Guodong Zhou. 2011. Cache-based document-level statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 909–919.

Liane Guillou. 2013. Analysing lexical consistency in translation. In *Proceedings of the Workshop on Discourse in Machine Translation*, pages 10–18.

Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. 2012. Document-wide decoding for phrase-based statistical machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1179–1190.

Eva Hasler, Phil Blunsom, Philipp Koehn, and Barry Haddow. 2014a. Dynamic topic adaptation for phrase-based mt. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, Gothenburg, Sweden*.

Eva Hasler, Barry Haddow, and Philipp Koehn. 2014b. Dynamic topic adaptation for smt using distributional profiles. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 445–456, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

Hans Hjelm. 2007. Identifying cross language term equivalents using statistical machine translation and distributional association measures. In *Proceedings of 16th Nordic Conference of Computational Linguistics Nodalida*, pages 97–104.

Masaki Itagaki and Takako Aikawa. 2008. Post-mt term swapper: Supplementing a statistical machine translation system with a user dictionary. In *LREC*.

Masaki Itagaki, Takako Aikawa, and Xiaodong He. 2007. Automatic validation of terminology translation consistency with statistical method. *Proceedings of MT summit XI*, pages 269–274.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54.

Els Lefever, Lieve Macken, and Veronique Hoste. 2009. Language-independent bilingual terminology extraction from a multilingual parallel corpus. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 496–504.

Magnus Merkel and Jody Foo. 2007. Terminology extraction and term ranking for standardizing term banks. In *Proceedings of 16th Nordic Conference of Computational Linguistics Nodalida*, pages 349–354.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167.

Scott SL Piao, Guangfan Sun, Paul Rayson, and Qi Yuan. 2006. Automatic extraction of chinese multiword expressions with a statistical tool. In *Workshop on Multi-word-expressions in a Multilingual Context held in conjunction with the 11th EACL, Trento, Italy*, pages 17–24.

Pinis and Skadins. 2012. Mt adaptation for under-resourced domains–what works and what not. In *Human Language Technologies–The Baltic Perspective: Proceedings of the Fifth International Conference Baltic HLT 2012*, volume 247, page 176. IOS Press.

Zhixiang Ren, Yajuan Lü, Jie Cao, Qun Liu, and Yun Huang. 2009. Improving statistical machine translation using domain bilingual multiword expressions. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*, pages 47–54.

Andreas Stolcke et al. 2002. Srilm-an extensible language modeling toolkit. In *Proceedings of the international conference on spoken language processing*, volume 2, pages 901–904.

Jinsong Su, Hua Wu, Haifeng Wang, Yidong Chen, Xiaodong Shi, Huailin Dong, and Qun Liu. 2012. Translation model adaptation for statistical machine translation with monolingual topic information. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 459–468.

Jörg Tiedemann. 2010. Context adaptation in statistical machine translation using models with exponentially decaying cache. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 8–15.

Ferhan Ture, Douglas W Oard, and Philip Resnik. 2012. Encouraging consistent translation choices. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 417–426. Association for Computational Linguistics.

Muriel Vasconcellos, Brian Avey, Claudia Gdaniec, Laurie Gerber, Marjorie León, and Teruko Mitamura. 2001. Terminology and machine translation. *Handbook of Terminology Management*, 2:697–723.

Thuy Vu, Ai Ti Aw, and Min Zhang. 2008. Term extraction through unithood and termhood unification. In *Proceedings of the third international joint conference on natural language processing*.

Billy Wong and Chunyu Kit. 2012. Extending machine translation evaluation metrics with lexical cohesion to document level. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1060–1068.

Xinyan Xiao, Deyi Xiong, Min Zhang, Qun Liu, and Shouxun Lin. 2012. A topic similarity model for hierarchical phrase-based translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 750–758.

Deyi Xiong and Min Zhang. 2013. A topic-based coherence model for statistical machine translation. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI-13), Bellevue, Washington, USA, July*.

Deyi Xiong, Min Zhang, Aiti Aw, and Haizhou Li. 2009. A syntax-driven bracketing model for phrase-based translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 315–323.

Deyi Xiong, Guosheng Ben, Min Zhang, Yajuan Lü, and Qun Liu. 2013a. Modeling lexical cohesion for document-level machine translation. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2183–2189. AAAI Press.

Deyi Xiong, Yang Ding, Min Zhang, and Chew Lim Tan. 2013b. Lexical chain based cohesion models for document-level statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1563—1573.

Bing Zhao and Eric P Xing. 2006. Bitam: Bilingual topic admixture models for word alignment. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 969–976.

Beyond Parallel Data: Joint Word Alignment and Decipherment Improves Machine Translation

Qing Dou , Ashish Vaswani, and Kevin Knight
Information Sciences Institute
Department of Computer Science
University of Southern California
{qdou,avaswani,knight}@isi.edu

Abstract

Inspired by previous work, where decipherment is used to improve machine translation, we propose a new idea to combine word alignment and decipherment into a single learning process. We use EM to estimate the model parameters, not only to maximize the probability of parallel corpus, but also the monolingual corpus. We apply our approach to improve Malagasy-English machine translation, where only a small amount of parallel data is available. In our experiments, we observe gains of 0.9 to 2.1 B over a strong baseline.

1 Introduction

State-of-the-art machine translation (MT) systems apply statistical techniques to learn translation rules automatically from parallel data. However, this reliance on parallel data seriously limits the scope of MT application in the real world, as for many languages and domains, there is not enough parallel data to train a decent quality MT system.

However, compared with parallel data, there are much larger amounts of non parallel data. The ability to learn a translation lexicon or even build a machine translation system using monolingual data helps address the problems of insufficient parallel data. Ravi and Knight (2011) are among the first to learn a full MT system using only non parallel data through decipherment. However, the performance of such systems is much lower compared with those trained with parallel data. In another work, Klementiev et al. (2012) show that, given a phrase table, it is possible to estimate parameters for a phrase-based MT system from non parallel data.

Given that we often have some parallel data, it is more practical to improve a translation system trained on parallel data by using additional non parallel data. Rapp (1995) shows that with a seed lexicon, it is possible to induce new word level translations from non parallel data. Motivated by the idea that a translation lexicon induced from non parallel data can be used to translate out of vocabulary words (OOV), a variety of prior research has tried to build a translation lexicon from non parallel or comparable data (Fung and Yee, 1998; Koehn and Knight, 2002; Haghighi et al., 2008; Garera et al., 2009; Bergsma and Van Durme, 2011; Daumé and Jagarlamudi, 2011; Irvine and Callison-Burch, 2013b; Irvine and Callison-Burch, 2013a; Irvine et al., 2013).

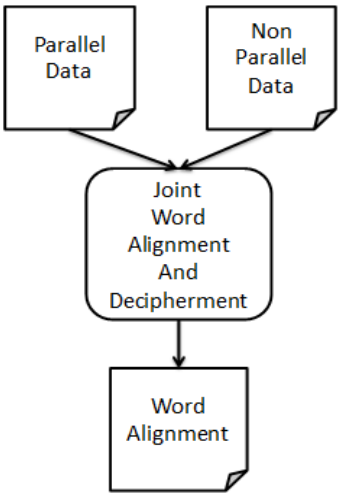


Figure 1: Combine word alignment and decipherment into a single learning process.

Lately, there has been increasing interest in learning translation lexicons from non parallel data with decipherment techniques (Ravi and Knight, 2011; Dou and Knight, 2012; Nuhn et al., 2012; Dou and Knight, 2013). Decipherment views one language as a cipher for another and learns a translation lexicon that produces fluent text in the target (plaintext) language. Previous work has shown that decipherment not only helps find translations for OOVs (Dou and Knight, 2012), but also improves translations of observed words (Dou and Knight, 2013).

We find that previous work using monolingual or comparable data to improve quality of machine translation separates two learning tasks: first, translation rules are learned from parallel data, and then the information learned from parallel data is used to bootstrap learning with non parallel data. Inspired by approaches where joint inference reduces the problems of error propagation and improves system performance, we combine the two separate learning processes into a single one, as shown in Figure 1. The contributions of this work are:

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 557–565,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

- We propose a new objective function for word alignment that combines the process of word alignment and decipherment into a single learning task.

- In experiments, we find that the joint process outperforms the previous pipeline approach, and observe B gains of 0.9 and 2.1 on two different test sets.

- We release 15.3 million tokens of monolingual Malagasy data from the web, as well as a small Malagasy dependency tree bank containing 20k tokens.

2 Joint Word Alignment and Decipherment

2.1 A New Objective Function

In previous work that uses monolingual data to improve machine translation, a seed translation lexicon learned from parallel data is used to find new translations through either word vector based approaches or decipherment. In return, selection of a seed lexicon needs to be careful as using a poor quality seed lexicon could hurt the downstream process. Evidence from a number of previous work shows that a joint inference process leads to better performance in both tasks (Jiang et al., 2008; Zhang and Clark, 2008).

In the presence of parallel and monolingual data, we would like the alignment and decipherment models to benefit from each other. Since the decipherment and word alignment models contain word-to-word translation probabilities $t(f \mid e)$, having them share these parameters during learning will allow us to pool information from both data types. This leads us to develop a new objective function that takes both learning processes into account. Given our parallel data, $(\mathbf{E}^1, \mathbf{F}^1), \ldots, (\mathbf{E}^m, \mathbf{F}^m), \ldots, (\mathbf{E}^M, \mathbf{F}^M)$, and monolingual data $\mathbf{F}^1_{\mathbf{mono}}, \ldots, \mathbf{F}^n_{\mathbf{mono}}, \ldots, \mathbf{F}^N_{\mathbf{mono}}$, we seek to maximize the likelihood of both. Our new objective function is defined as:

$$F_{joint} = \sum_{m=1}^{M} \log P(\mathbf{F}^m \mid \mathbf{E}^m) + \alpha \sum_{n=1}^{N} \log P(\mathbf{F}^n_{\mathbf{mono}}) \quad (1)$$

The goal of training is to learn the parameters that maximize this objective, that is

$$\theta^* = \arg\max_{\theta} F_{joint} \quad (2)$$

In the next two sections, we describe the word alignment and decipherment models, and present how they are combined to perform joint optimization.

2.2 Word Alignment

Given a source sentence $\mathbf{F} = \mathbf{f_1}, \ldots, \mathbf{f_j}, \ldots, \mathbf{f_J}$ and a target sentence $\mathbf{E} = \mathbf{e_1}, \ldots, \mathbf{e_i}, \ldots, \mathbf{e_I}$, word alignment models describe the generative process employed to produce the French sentence from the English sentence through alignments $\mathbf{a} = \mathbf{a_1}, \ldots, \mathbf{a_j}, \ldots, \mathbf{a_J}$.

The IBM models 1-2 (Brown et al., 1993) and the HMM word alignment model (Vogel et al., 1996) use two sets of parameters, *distortion* probabilities and *translation* probabilities, to define the joint probability of a target sentence and alignment given a source sentence.

$$P(\mathbf{F}, \mathbf{a} \mid \mathbf{E}) = \prod_{j=1}^{J} d(a_j \mid a_{j-1}, j) t(f_j \mid e_{a_j}). \quad (3)$$

These alignment models share the same translation probabilities $t(f_j \mid e_{a_j})$, but differ in their treatment of the distortion probabilities $d(a_j \mid a_{j-1}, j)$. Brown et al. (1993) introduce more advanced models for word alignment, such as Model 3 and Model 4, which use more parameters to describe the generative process. We do not go into details of those models here and the reader is referred to the paper describing them.

Under the Model 1-2 and HMM alignment models, the probability of target sentence given source sentence is:

$$P(\mathbf{F} \mid \mathbf{E}) = \sum_{\mathbf{a}} \prod_{j=1}^{J} d(a_j \mid a_{j-1}, j) t(f_j \mid e_{a_j}).$$

Let θ denote all the parameters of the word alignment model. Given a corpus of sentence pairs $(\mathbf{E}^1, \mathbf{F}^1), \ldots, (\mathbf{E}^m, \mathbf{F}^m), \ldots, (\mathbf{E}^M, \mathbf{F}^M)$, the standard approach for training is to learn the maximum likelihood estimate of the parameters, that is,

$$\begin{aligned} \theta^* &= \arg\max_{\theta} \sum_{m=1}^{M} \log P(\mathbf{F}^m \mid \mathbf{E}^m) \\ &= \arg\max_{\theta} \log \left(\sum_{\mathbf{a}} P(\mathbf{F}^m, \mathbf{a} \mid \mathbf{E}^m) \right). \end{aligned}$$

We typically use the EM algorithm (Dempster et al., 1977), to carry out this optimization.

2.3 Decipherment

Given a corpus of N foreign text sequences (ciphertext), $\mathbf{F}^1_{\mathbf{mono}}, \ldots, \mathbf{F}^n_{\mathbf{mono}}, \ldots, \mathbf{F}^N_{\mathbf{mono}}$, decipherment finds word-to-word translations that best describe the ciphertext.

Knight et al. (2006) are the first to study several natural language decipherment problems with unsupervised learning. Since then, there has been increasing interest in improving decipherment techniques and its application to machine translation (Ravi and Knight, 2011;

Dou and Knight, 2012; Nuhn et al., 2012; Dou and Knight, 2013; Nuhn et al., 2013).

In order to speed up decipherment, Dou and Knight (2012) suggest that a frequency list of bigrams might contain enough information for decipherment. According to them, a monolingual ciphertext bigram $\mathbf{F_{mono}}$ is generated through the following generative story:

- Generate a sequence of two plaintext tokens e_1e_2 with probability $P(e_1e_2)$ given by a language model built from large numbers of plaintext bigrams.
- Substitute e_1 with f_1 and e_2 with f_2 with probability $t(f_1|e_1) \cdot t(f_2|e_2)$.

The probability of any cipher bigram F is:

$$P(\mathbf{F_{mono}}) = \sum_{e_1e_2} P(e_1e_2) \cdot t(f_1|e_1) \cdot t(f_2|e_2) \quad (4)$$

And the probability of the corpus is:

$$P(corpus) = \prod_{n=1}^{N} P(\mathbf{F}^n_{\mathbf{mono}}) \quad (5)$$

Given a plaintext bigram language model, the goal is to manipulate $t(f|e)$ to maximize $P(corpus)$. Theoretically, one can directly apply EM to solve the problem (Knight et al., 2006). However, EM has time complexity $O(N \cdot V_e^2)$ and space complexity $O(V_f \cdot V_e)$, where V_f, V_e are the sizes of ciphertext and plaintext vocabularies respectively, and N is the number of cipher bigrams.

There have been previous attempts to make decipherment faster. Ravi and Knight (2011) apply Bayesian learning to reduce the space complexity. However, Bayesian decipherment is still very slow with Gibbs sampling (Geman and Geman, 1987). Dou and Knight (2012) make sampling faster by introducing slice sampling (Neal, 2000) to Bayesian decipherment. Besides Bayesian decipherment, Nuhn et al. (2013) show that beam search can be used to solve a very large 1:1 word substitution cipher. In subsection 2.4.1, we describe our approach that uses slice sampling to compute expected counts for decipherment in the EM algorithm.

2.4 Joint Optimization

We now describe our EM approach to learn the parameters that maximize F_{joint} (equation 2), where the distortion probabilities, $d(a_j \mid a_{j-1}, j)$ in the word alignment model are only learned from parallel data, and the translation probabilities, $t(f \mid e)$ are learned using both parallel and non parallel data. The E step and M step are illustrated in Figure 2.

Our algorithm starts with EM learning only on parallel data for a few iterations. When the joint inference starts, we first compute expected counts from parallel data and non parallel data using parameter values from the last M step separately. Then, we add the expected counts from both parallel data and non parallel data together with different weights for the two. Finally we renormalize the translation table and distortion table to update parameters in the new M step.

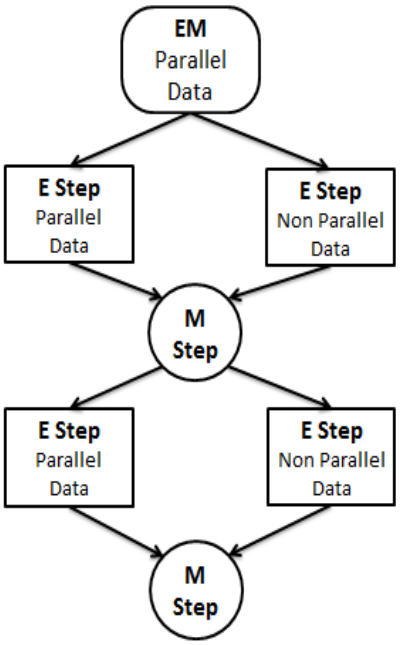


Figure 2: Joint Word Alignment and Decipherment with EM

The E step for parallel part can be computed efficiently using the forward-backward algorithm (Vogel et al., 1996). However, as we pointed out in Section 2.3, the E step for the non parallel part has a time complexity of $O(V^2)$ with the forward-backward algorithm, where V is the size of English vocabulary, and is usually very large. Previous work has tried to make decipherment scalable (Ravi and Knight, 2011; Dou and Knight, 2012; Nuhn et al., 2013; Ravi, 2013). However, all of them are designed for decipherment with either Bayesian inference or beam search. In contrast, we need an algorithm to make EM decipherment scalable. To overcome this problem, we modify the slice sampling (Neal, 2000) approach used by Dou and Knight (2012) to compute expected counts from non parallel data needed for the EM algorithm.

2.4.1 Draw Samples with Slice Sampling

To start the sampling process, we initialize the first sample by performing approximate Viterbi decoding using results from the last EM iteration. For each foreign dependency bigram f_1, f_2, we find the top 50 candidates for f_1 and f_2 ranked by $t(e|f)$, and find the English sequence e_1, e_2 that maximizes $t(e_1|f_1) \cdot t(e_2|f_2) \cdot P(e_1, e_2)$.

Suppose the derivation probability for current sample *e_current* is $P(e\_current)$, we use slice sampling to draw a new sample in two steps:

- Select a threshold T uniformly between 0 and $P(e\_current)$.
- Draw a new sample *e_new* uniformly from a pool

of candidates: $\{e\_new|P(e\_new) > T\}$.

The first step is straightforward to implement. However, it is not trivial to implement the second step. We adapt the idea from Dou and Knight (2012) for EM learning.

Suppose our current sample *e_current* contains English tokens e_{i-1}, e_i, and e_{i+1} at position $i-1$, i, and $i+1$ respectively, and f_i be the foreign token at position i. Using point-wise sampling, we draw a new sample by changing token e_i to a new token e'. Since the rest of the sample remains the same, only the probability of the trigram $P(e_{i-1}e'e_{i+1})$ (The probability is given by a bigram language model.), and the channel model probability $t(f_i|e')$ change. Therefore, the probability of a sample is simplified as shown Equation 6.

$$P(e_{i-1}e'e_{i+1}) \cdot t(f_i|e') \tag{6}$$

Remember that in slice sampling, a new sample is drawn in two steps. For the first step, we choose a threshold T uniformly between 0 and $P(e_{i-1}e_ie_{i+1}) \cdot t(f_i|e_i)$. We divide the second step into two cases based on the observation that two types of samples are more likely to have a probability higher than T (Dou and Knight, 2012): (1) those whose trigram probability is high, and (2) those whose channel model probability is high. To find candidates that have high trigram probability, Dou and Knight (2012) build a top k sorted lists ranked by $P(e_{i-1}e'e_{i+1})$, which can be pre-computed off-line. Then, they test if the last item e_k in the list satisfies the following inequality:

$$P(e_{i-1}e_ke_{i+1}) \cdot c < T \tag{7}$$

where c is a small constant and is set to *prior* in their work. In contrast, we choose c empirically as we do not have a prior in our model. When the inequality in Equation 7 is satisfied, a sample is drawn in the following way: Let set $A = \{e'|e_{i-1}e'e_{i+1} \cdot c > T\}$ and set $B = \{e'|t(f_i|e') > c\}$. Then we only need to sample e' uniformly from $A \cup B$ until $P(e_{i-1}e'e_{i+1}) \cdot t(f_i|e')$ is greater than T. It is easy to prove that all other candidates that are not in the sorted list and with $t(f_i|e') \leq c$ have a upper bound probability: $P(e_{i-1}e_ke_{i+1}) \cdot c$. Therefore, they do not need to be considered.

Second, when the last item e_k in the list does not meet the condition in Equation 7, we keep drawing samples e' randomly until its probability is greater than the threshold T.

As we mentioned before, the choice of the small constant c is empirical. A large c reduces the number of items in set B, but makes the condition $P(e_{i-1}e_ke_{i+1}) \cdot c < T$ less likely to satisfy, which slows down the sampling. On the contrary, a small c increases the number of items in set B significantly as EM does not encourage a sparse distribution, which also slows down the sampling. In our experiments, we set c to 0.001 based on the speed of decipherment. Furthermore, to reduce the size of set B, we rank all the candidate translations of f_i by $t(e'|f_i)$, then we add maximum the first 1000 candidates whose $t(f_i|e') >= c$ into set B. For the rest of the candidates, we set $t(f_i|e')$ to a value smaller than c (0.00001 in experiments).

	Spanish	English
Parallel	10.3k	9.9k
Non Parallel	80 million	400 million

Table 1: Size of parallel and non parallel data for word alignment experiments (Measured in number of tokens)

2.4.2 Compute Expected Counts from Samples

With the ability to draw samples efficiently for decipherment using EM, we now describe how to compute expected counts from those samples. Let f_1, f_2 be a specific ciphertext bigram, N be the number of samples we want to use to compute expected counts, and e_1, e_2 be one of the N samples. The expected counts for pairs (f_1, e_1) and (f_2, e_2) are computed as:

$$\alpha \cdot \frac{count(f_1, f_2)}{N}$$

where $count(f_1, f_2)$ is count of the bigram, and α is the weight for non parallel data as shown in Equation 1. Expected counts collected for f_1, f_2 are accumulated from each of its N samples. Finally, we collect expected counts using the same approach from each foreign bigram.

3 Word Alignment Experiments

In this section, we show that joint word alignment and decipherment improves the quality of word alignment. We choose to evaluate word alignment performance for Spanish and English as manual gold alignments are available. In experiments, our approach improves alignment F score by as much as 8 points.

3.1 Experiment Setup

As shown in Table 1, we work with a small amount of parallel, manually aligned Spanish-English data (Lambert et al., 2005), and a much larger amount of monolingual data.

The parallel data is extracted from Europarl, which consists of articles from European parliament plenary sessions. The monolingual data comes from English and Spanish versions of Gigaword corpra containing news articles from different news agencies.

We view Spanish as a cipher of English, and follow the approach proposed by Dou and Knight (2013) to extract dependency bigrams from parsed Spanish and English monolingual data for decipherment. We only keep bigrams where both tokens appear in the parallel data. Then, we perform Spanish to English (English generating Spanish) word alignment and Spanish to English decipherment simultaneously with the method discussed in section 2.

3.1.1 Results

We align all 500 sentences in the parallel corpus, and tune the decipherment weight (α) for Model 1 and HMM using the last 100 sentences. The best weights are 0.1 for Model 1, and 0.005 for HMM. We start with Model 1 with only parallel data for 5 iterations, and switch to the joint process for another 5 iterations with Model 1 and 5 more iterations of HMM. In the end, we use the first 100 sentence pairs of the corpus for evaluation.

Figure 3 compares the learning curve of alignment F-score between EM without decipherment (baseline) and our joint word alignment and decipherment. From the learning curve, we find that at the 6th iteration, 2 iterations after we start the joint process, alignment F-score is improved from 34 to 43, and this improvement is held through the rest of the Model 1 iterations. The alignment model switches to HMM from the 11th iteration, and at the 12th iteration, we see a sudden jump in F-score for both the baseline and the joint approach. We see consistent improvement of F-score till the end of HMM iterations.

4 Improving Low Density Languages Machine Translation with Joint Word Alignment and Decipherment

In the previous section, we show that the joint word alignment and decipherment process improves quality of word alignment significantly for Spanish and English. In this section, we test our approach in a more challenging setting: improving the quality of machine translation in a real low density language setting.

In this task, our goal is to build a system to translate Malagasy news into English. We have a small amount of parallel data, and larger amounts of monolingual data collected from online websites. We build a dependency parser for Malagasy to parse the monolingual data to perform dependency based decipherment (Dou and Knight, 2013). In the end, we perform joint word alignment and decipherment, and show that the joint learning process improves B scores by up to 2.1 points over a phrase-based MT baseline.

4.1 The Malagasy Language

Malagasy is the official language of Madagascar. It has around 18 million native speakers. Although Madagascar is an African country, Malagasy belongs to the Malayo-Polynesian branch of the Austronesian language family. Malagasy and English have very different word orders. First of all, in contrast to English, which has a subject-verb-object (SVO) word order, Malagasy has a verb-object-subject (VOS) word order. Besides that, Malagasy is a typical head initial language: Determiners precede nouns, while other modifiers and relative clauses follow nouns (e.g. ny "the" boky "book" mena "red"). The significant differences in word order pose great challenges for both machine translation and decipherment.

Source	Malagasy	English
Parallel		
Global Voices	2.0 million	1.8 million
Web News	2.2k	2.1k
Non Parallel		
Gigaword	N/A	2.4 billion
allAfrica	N/A	396 million
Local News	15.3 million	N/A

Table 2: Size of Malagasy and English data used in our experiments (Measured in number of tokens)

4.2 Data

Table 2 shows the data available to us in our experiments. The majority of parallel text comes from Global Voices[1] (GV). The website contains international news translated into different foreign languages. Besides that, we also have a very small amount of parallel text containing local web news, with English translations provided by native speakers at the University of Texas, Austin. The Malagasy side of this small parallel corpus also has syntactical annotation, which is used to train a very basic Malagasy part of speech tagger and dependency parser.

We also have much larger amounts of non parallel data for both languages. For Malagasy, we spent two months manually collecting 15.3 million tokens of news text from local news websites in Madagascar.[2] We have released this data for future research use. For English, we have 2.4 billion tokens from the Gigaword corpus. Since the Malagasy monolingual data is collected from local websites, it is reasonable to argue that those data contain significant amount of information related to Africa. Therefore, we also collect 396 million tokens of African news in English from allAfrica.com.

4.3 Building A Dependency Parser for Malagasy

Since Malagasy and English have very different word orders, we decide to apply dependency based decipherment for the two languages as suggested by Dou and Knight (2013). To extract dependency relations, we need to parse monolingual data in Malagasy and English. For English, there are already many good parsers available. In our experiments, we use Turbo parser (Martins et al., 2013) trained on the English Penn Treebank (Marcus et al., 1993) to parse all our English monolingual data. However, there is no existing good parser for Malagasy.

The quality of a dependency parser depends on the amount of training data available. State-of-the-art English parsers are built from Penn Treebank, which contains over 1 million tokens of annotated syntactical

[1] globalvoicesonline.org

[2] aoraha.com, gazetiko.com, inovaovao.com, expressmada.com, lakroa.com

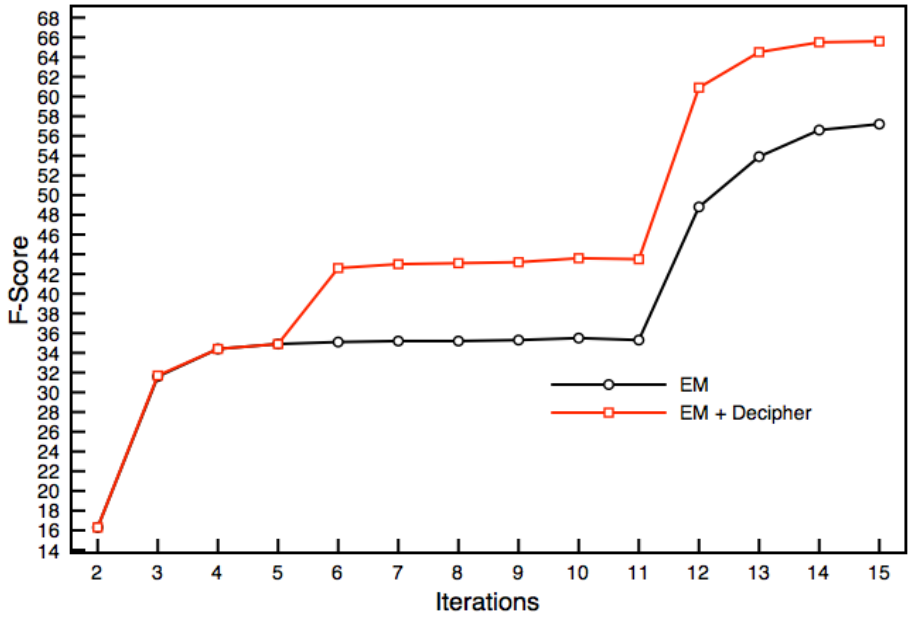


Figure 3: Learning curve showing our joint word alignment and decipherment approach improves word alignment quality over the traditional EM without decipherment (Model 1: Iteration 1 to 10, HMM: Iteration 11 to 15)

trees. In contrast, the available data for training a Malagasy parser is rather limited, with only 168 sentences, and 2.8k tokens, as shown in Table 2. At the very beginning, we use the last 120 sentences as training data to train a part of speech (POS) tagger using a toolkit provided by Garrette et al. (2013) and a dependency parser with the Turbo parser. We test the performance of the parser on the first 48 sentences and obtain 72.4% accuracy.

One obvious way to improve tagging and parsing accuracy is to get more annotated data. We find more data with only part of speech tags containing 465 sentences and 10k tokens released by (Garrette et al., 2013), and add this data as extra training data for POS tagger. Also, we download an online dictionary that contains POS tags for over 60k Malagasy word types from malagasyword.org. The dictionary is very helpful for tagging words never seen in the training data.

It is natural to think that creation of annotated data for training a POS tagger and a parser requires large amounts of efforts from annotators who understand the language well. However, we find that through the help of parallel data and dictionaries, we are able to create more annotated data by ourselves to improve tagging and parsing accuracy. This idea is inspired by previous work that tries to learn a semi-supervised parser by projecting dependency relations from one language (with good dependency parsers) to another (Yarowsky and Ngai, 2001; Ganchev et al., 2009). However, we find those automatic approaches do not work well for Malagasy.

To further expand our Malagasy training data, we first use a POS tagger and parser with poor performance to parse 788 sentences (20k tokens) on the Malagasy side of the parallel corpus from Global Voices. Then, we correct both the dependency links and POS tags based on information from dictionaries[3] and the English translation of the parsed sentence. We spent 3 months to manually project English dependencies to Malagasy and eventually improve test set parsing accuracy from 72.4% to 80.0%. We also make this data available for future research use.

4.4 Machine Translation Experiments

In this section, we present the data used for our MT experiments, and compare three different systems to justify our joint word alignment and decipherment approach.

4.4.1 Baseline Machine Translation System

We build a state-of-the-art phrase-based MT system, PBMT, using Moses (Koehn et al., 2007). PBMT has 3 models: a translation model, a distortion model, and a language model. We train the other models using half of the Global Voices parallel data (the rest is reserved for development and testing), and build a 5-gram language model using 834 million tokens from AFP section of English Gigaword, 396 million tokens from allAfrica, and the English part of the parallel corpus for training. For alignment, we run 10 iterations of Model 1, and 5 iterations of HMM. We did not run Model 3 and Model 4 as we see no improvements in B scores from running those models. We do word

[3] an online dictionary from malagasyword.org, as well as a lexicon learned from the parallel data

alignment in two directions and use grow-diag-final-and heuristic to obtain final alignment. During decoding, we use 8 standard features in Moses to score a candidate translation: direct and inverse translation probabilities, direct and inverse lexical weighting, a language model score, a distortion score, phrase penalty, and word penalty. The weights for the features are learned on the tuning data using minimum error rate training (MERT) (Och, 2003).

To compare with previous decipherment approach to improve machine translation, we build a second baseline system. We follow the work by Dou and Knight (2013) to decipher Malagasy into English, and build a translation lexicon $T_{decipher}$ from decipherment. To improve machine translation, we simply use $T_{decipher}$ as an additional parallel corpus. First, we filter $T_{decipher}$ by keeping only translation pairs (f, e), where f is observed in the Spanish part and e is observed in the English part of the parallel corpus. Then we append all the Spanish and English words in the filtered $T_{decipher}$ to the end of Spanish part and English part of the parallel corpus respectively. The training and tuning process is the same as the baseline machine translation system PBMT. We call this system **Decipher-Pipeline**.

4.4.2 Joint Word Alignment and Decipherment for Machine Translation

When deciphering Malagasy to English, we extract Malagasy dependency bigrams using all available Malagasy monolingual data plus the Malagasy part of the Global Voices parallel data, and extract English dependency bigrams using 834 million tokens from English Gigaword, and 396 million tokens from allAfrica news to build an English dependency language model. In the other direction, we extract English dependency bigrams from English part of the entire parallel corpus plus 9.7 million tokens from allAfrica news [4], and use 17.3 million tokens Malagasy monolingual data (15.3 million from the web and 2.0 million from Global Voices) to build a Malagasy dependency language model. We require that all dependency bigrams only contain words observed in the parallel data used to train the baseline MT system.

During learning, we run Model 1 without decipherment for 5 iterations. Then we perform joint word alignment and decipherment for another 5 iterations with Model 1 and 5 iterations with HMM. We tune decipherment weights (α) for Model 1 and HMM using grid search against B score on a development set. In the end, we only extract rules from one direction $P(English|Malagasy)$, where the decipherment weights for Model 1 and HMM are 0.5 and 0.005 respectively. We chose this because we did not find any benefits to tune the weights on each direction, and then use grow-diag-final-end heuristic to form final alignments. We call this system **Decipher-Joint**.

[4] We do not find further B gains by using more English monolingual data.

Parallel		
	Malagasy	English
Train (GV)	0.9 million	0.8 million
Tune (GV)	22.2k	20.2k
Test (GV)	23k	21k
Test (Web)	2.2k	2.1k
Non Parallel		
	Malagasy	English
Gigaword	N/A	834 million
Web	15.3 million	396 million

Table 3: Size of training, tuning, and testing data in number of tokens (GV: Global Voices)

4.5 Results

We tune each system three times with MERT and choose the best weights based on B scores on tuning set.

Table 4 shows that while using a translation lexicon learnt from decipherment does not improve the quality of machine translation significantly, the joint approach improves B score by 0.9 and 2.1 on Global Voices test set and web news test set respectively. The results show that the parsing quality correlates with gains in B scores. Scores in the brackets in the last row of the table are achieved using a dependency parser with 72.4% attachment accuracy, while scores outside the brackets are obtained using a dependency parser with 80.0% attachment accuracy.

We analyze the results and find the gain mainly comes from two parts. First, adding expected counts from non parallel data makes the distribution of translation probabilities sparser in word alignment models. The probabilities of translation pairs favored by both parallel data and decipherment becomes higher. This gain is consistent with previous observation where a sparse prior is applied to EM to help improve word alignment and machine translation (Vaswani et al., 2012). Second, expected counts from decipherment also help discover new translation pairs in the parallel data for low frequency words, where those words are either aligned to NULL or wrong translations in the baseline.

5 Conclusion and Future Work

We propose a new objective function for word alignment to combine the process of word alignment and decipherment into a single task. In, experiments, we find that the joint process performs better than previous pipeline approach, and observe B gains of 0.9 and 2.1 point on Global Voices and local web news test sets, respectively. Finally, our research leads to the release of 15.3 million tokens of monolingual Malagasy data from the web as well as a small Malagasy dependency tree bank containing 20k tokens.

Given the positive results we obtain by using the joint approach to improve word alignment, we are in-

Decipherment	System	Tune (GV)	Test (GV)	Test (Web)
None	PBMT (Baseline)	18.5	17.1	7.7
Separate	Decipher-Pipeline	18.5	17.4	7.7
Joint	Decipher-Joint	18.9 (18.7)	18.0 (17.7)	9.8 (8.5)

Table 4: Decipher-Pipeline does not show significant improvement over the baseline system. In contrast, Decipher-Joint using joint word alignment and decipherment approach achieves a B gain of 0.9 and 2.1 on the Global Voices test set and the web news test set, respectively. The results in brackets are obtained using a parser trained with only 120 sentences. (GV: Global Voices)

spired to apply this approach to help find translations for out of vocabulary words, and to explore other possible ways to improve machine translation with decipherment.

Acknowledgments

This work was supported by NSF Grant 0904684 and ARO grant W911NF-10-1-0533. The authors would like to thank David Chiang, Malte Nuhn, Victoria Fossum, Ashish Vaswani, Ulf Hermjakob, Yang Gao, and Hui Zhang (in no particular order) for their comments and suggestions.

References

Shane Bergsma and Benjamin Van Durme. 2011. Learning bilingual lexicons using the visual similarity of labeled web images. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Three*. AAAI Press.

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

Hal Daumé, III and Jagadeesh Jagarlamudi. 2011. Domain adaptation for machine translation by mining unseen words. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Arthur Dempster, Nan Laird, and Donald Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Computational Linguistics*, 39(4):1–38.

Qing Dou and Kevin Knight. 2012. Large scale decipherment for out-of-domain machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics.

Qing Dou and Kevin Knight. 2013. Dependency-based decipherment for resource-limited machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Pascale Fung and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*. Association for Computational Linguistics.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*. Association for Computational Linguistics.

Nikesh Garera, Chris Callison-Burch, and David Yarowsky. 2009. Improving translation lexicon induction from monolingual corpora via dependency contexts and part-of-speech equivalences. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of pos-taggers for low-resource languages. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Stuart Geman and Donald Geman. 1987. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. In *Readings in computer vision: issues, problems, principles, and paradigms*. Morgan Kaufmann Publishers Inc.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics.

Ann Irvine and Chris Callison-Burch. 2013a. Combining bilingual and comparable corpora for low resource machine translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, August.

Ann Irvine and Chris Callison-Burch. 2013b. Supervised bilingual lexicon induction with multiple monolingual signals. In *Proceedings of the 2013*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics.

Ann Irvine, Chris Quirk, and Hal Daume III. 2013. Monolingual marginal matching for translation model adaptation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008. A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics.

Alexandre Klementiev, Ann Irvine, Chris Callison-Burch, and David Yarowsky. 2012. Toward statistical machine translation without parallel corpora. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*. Association for Computational Linguistics.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of the ACL-02 Workshop on Unsupervised Lexical Acquisition*. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics.

Patrik Lambert, Adriá De Gispert, Rafael Banchs, and José B. Mariño. 2005. Guidelines for word alignment evaluation and manual alignment. *Language Resources and Evaluation*, 39(4):267–285.

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

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the Turbo: Fast third-order non-projective Turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics.

Radford Neal. 2000. Slice sampling. *Annals of Statistics*, 31.

Malte Nuhn, Arne Mauser, and Hermann Ney. 2012. Deciphering foreign language by combining language models and context vectors. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*. Association for Computational Linguistics.

Malte Nuhn, Julian Schamper, and Hermann Ney. 2013. Beam search for solving substitution ciphers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of Association for Computational Linguistics*. Association for Computational Linguistics.

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of the 33rd annual meeting of Association for Computational Linguistics*. Association for Computational Linguistics.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Sujith Ravi. 2013. Scalable decipherment for machine translation via hash sampling. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Ashish Vaswani, Liang Huang, and David Chiang. 2012. Smaller alignment models for better translations: Unsupervised word alignment with the l0-norm. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*. Association for Computational Linguistics.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*. Association for Computational Linguistics.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL-08: HLT*, Columbus, Ohio. Association for Computational Linguistics.

Latent Domain Phrase-based Models for Adaptation

Hoang Cuong and **Khalil Sima'an**
Institute for Logic, Language and Computation
University of Amsterdam
Science Park 107, 1098 XG Amsterdam, The Netherlands
{c.hoang,k.simaan}@uva.nl

Abstract

Phrase-based models directly trained on *mix-of-domain corpora* can be sub-optimal. In this paper we equip phrase-based models with a *latent domain* variable and present a novel method for adapting them to an in-domain task represented by a seed corpus. We derive an EM algorithm which alternates between inducing domain-focused phrase pair estimates, and weights for mix-domain sentence pairs reflecting their relevance for the in-domain task. By embedding our latent domain phrase model in a sentence-level model and training the two in tandem, we are able to adapt *all core* translation components together – phrase, lexical and reordering. We show experiments on weighing sentence pairs for relevance as well as adapting phrase-based models, showing significant performance improvement in both tasks.

1 Mix vs. Latent Domain Models

Domain adaptation is usually perceived as utilizing a small seed in-domain corpus to adapt an existing system trained on an *out-of-domain* corpus. Here we are interested in adapting an SMT system trained on a large *mix-domain* corpus C_{mix} to an *in-domain task* represented by a seed parallel corpus C_{in}. The mix-domain scenario is interesting because often a large corpus consists of sentence pairs representing diverse domains, e.g., news, politics, finance, sports, etc.

At the core of a standard state-of-the-art phrase-based system (Och and Ney, 2004) is a phrase table $\{\langle \tilde{e}, \tilde{f} \rangle\}$ extracted from the word-aligned training data together with estimates for $P_t(\tilde{e} \mid \tilde{f})$ and $P_t(\tilde{f} \mid \tilde{e})$. Because the translations of words often vary across domains, it is likely that in a mix-domain corpus C_{mix} the translation ambiguity will increase with the *domain diversity*. Furthermore, the statistics in C_{mix} will reflect translation preferences *averaged* over the diverse domains. In this sense, phrase-based models trained on C_{mix} can be considered *domain-confused*. This often leads to suboptimal performance (Gascó et al., 2012; Irvine et al., 2013).

Recent adaptation techniques can be seen as mixture models, where two or more phrase tables, estimated from in- and mix-domain corpora, are combined together by interpolation, fill-up, or multiple-decoding paths (Koehn and Schroeder, 2007; Bisazza et al., 2011; Sennrich, 2012; Razmara et al., 2012; Sennrich et al., 2013). Here we are interested in the specific question *how to induce* a phrase-based model *from* C_{mix} for *in-domain translation*? We view this as *in-domain focused training* on C_{mix}, a complementary adaptation step which might precede any further combination with other models, e.g., in-, mix- or general-domain.

The main challenge is how to induce from C_{mix} a phrase-based model for the in-domain task, given only C_{in} as evidence? We present an approach whereby the contrast between in-domain prior distributions and "out-domain" distributions is exploited for softly inviting (or recruiting) C_{mix} phrase pairs to either camp. To this end we in-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 566–576,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

troduce a *latent domain variable* D to signify in- (D_1) and out-domain (D_0) respectively.[1]

With the introduction of the latent variables, we extend the translation tables in phrase-based models from generic $P_t(\tilde{e} \mid \tilde{f})$ to domain-focused by conditioning them on D, i.e., $P_t(\tilde{e} \mid \tilde{f}, D)$ and decomposing them as follows:

$$P_t(\tilde{e} \mid \tilde{f}, D) = \frac{P_t(\tilde{e} \mid \tilde{f})P(D \mid \tilde{e}, \tilde{f})}{\sum_{\tilde{e}} P_t(\tilde{e} \mid \tilde{f})P(D \mid \tilde{e}, \tilde{f})}. \quad (1)$$

Where $P(D \mid \tilde{e}, \tilde{f})$ is viewed as the *latent phrase-relevance models*, i.e., the probability that a phrase pair is in- (D_1) or out-domain (D_0). In the end, our goal is to replace the domain-confused tables, $P_t(\tilde{e} \mid \tilde{f})$ and $P_t(\tilde{f} \mid \tilde{e})$, with the in-domain focused ones, $P_t(\tilde{e} \mid \tilde{f}, D_1)$ and $P_t(\tilde{f} \mid \tilde{e}, D_1)$.[2] Note how $P_t(\tilde{e} \mid \tilde{f}, D_1)$ and $P_t(\tilde{f} \mid \tilde{e}, D_1)$ contains $P_t(\tilde{e} \mid \tilde{f})$ and $P_t(\tilde{f} \mid \tilde{e})$ as special case.

Eq. 1 shows that the key to training the latent phrase-based translation models is to train the latent phrase-relevance models, $P(D \mid \tilde{e}, \tilde{f})$. Our approach is to embed $P(D \mid \tilde{e}, \tilde{f})$ in asymmetric sentence-level models $P(D \mid \mathbf{e}, \mathbf{f})$ and train them on C_{mix}. We devise an EM algorithm where at every iteration, in- or out-domain estimates provide full sentence pairs $\langle \mathbf{e}, \mathbf{f} \rangle$ with expectations $\{P(D \mid \mathbf{e}, \mathbf{f}) \mid D \in \{0, 1\}\}$. Once these expectation are in C_{mix}, we induce re-estimates for the latent phrase-relevance models, $P(D \mid \tilde{e}, \tilde{f})$. Metaphorically, during each EM iteration the current in- or out-domain phrase pairs compete on *inviting* C_{mix} sentence pairs to be in- or out-domain, which bring in new (weights for) in- and out-domain phrases. Using the same algorithm we also show how to adapt all core translation components in tandem, including also lexical weights and lexicalized reordering models.

Next we detail our model, the EM-based invitation training algorithm and provide technical solutions to a range of difficulties. We report experiments showing good instance weighting performance as well as significantly improved phrase-based translation performance.

[1] Crucially, the lack of explicit out-domain data in C_{mix} is a major technical difficulty. We follow (Cuong and Sima'an, 2014) and in the sequel present a relatively efficient solution based on a kind of "burn-in" procedure.

[2] It is common to use these domain-focused models as additional features besides the domain-confused features. However, here we are more interested in *replacing* the domain-confused features rather than complementing them. This distinguishes this work from other domain adaptation literature for MT.

2 Model and training by invitation

Eq. 1 shows that the key to training the latent phrase-based translation models is to train the latent phrase-relevance models, $P(D \mid \tilde{e}, \tilde{f})$. As mentioned, for training $P(D \mid \tilde{e}, \tilde{f})$ on parallel sentences in C_{mix} we embed them in two asymmetric sentence-level models $\{P(D \mid \mathbf{e}, \mathbf{f}) \mid D \in \{0, 1\}\}$.

2.1 Domain relevance sentence models

Intuitively, sentence models for domain relevance $P(D \mid \mathbf{e}, \mathbf{f})$ are somewhat related to *data selection* approaches (Moore and Lewis, 2010; Axelrod et al., 2011). The dominant approach to data selection uses the contrast between perplexities of in- and mix-domain language models.[3] In the translation context, however, often a source phrase has different senses/translations in different domains, which cannot be distinguished with monolingual language models (Cuong and Sima'an, 2014). Therefore, our proposed latent sentence-relevance model includes two major latent components - *monolingual domain-focused relevance models* and *domain-focused translation models* derives as follows:

$$P(D \mid \mathbf{e},\ \mathbf{f}) = \frac{P(\mathbf{e}, \mathbf{f}, D)}{\sum_{D \in \{D_1, D_0\}} P(\mathbf{e}, \mathbf{f}, D)}, \quad (2)$$

where $P(\mathbf{e}, \mathbf{f}, D)$ can be decomposed as:

$$P(\mathbf{f}, \mathbf{e}, D) = \frac{1}{2}\Big(P(D)P_{lm}(\mathbf{e} \mid D)P_t(\mathbf{f} \mid \mathbf{e},\ D) + P(D)P_{lm}(\mathbf{f} \mid D)P_t(\mathbf{e} \mid \mathbf{f},\ D)\Big). \quad (3)$$

Here

- $P_t(\mathbf{e}|\mathbf{f}, D)$ and similarly $P_t(\mathbf{f}|\mathbf{e}, D)$: the latent domain-focused translation models aim at capturing the faithfulness of translation with respect to different domains. We simplify this as

[3] Note that earlier work on data selection exploits the contrast between in- and mix-domain. In (Cuong and Sima'an, 2014), we present the idea of using the language and translation models derived separately from in- and out-domain data, and show how it helps for data selection.

"bag-of-possible-phrases" translation models:[4]

$$P_t(\mathbf{e}|\mathbf{f}, D) := \prod_{\langle\tilde{e},\tilde{f}\rangle \in \mathcal{A}(\mathbf{e},\mathbf{f})} P_t(\tilde{e}|\tilde{f}, D)^{c(\tilde{e},\tilde{f})}, \quad (4)$$

where $\mathcal{A}(\mathbf{e}, \mathbf{f})$ is the multiset of phrases in $\langle \mathbf{e}, \mathbf{f} \rangle$ and $c(\cdot)$ denotes their count. Sub-model $P_t(\tilde{e}|\tilde{f}, D)$ is given by Eq. 1.

- $P_{lm}(\mathbf{e}|D)$, $P_{lm}(\mathbf{f}|D)$: the latent monolingual domain-focused relevance models aim at capturing the relevance of $\mathbf{e}$ and $\mathbf{f}$ for identifying domain D but here we consider them language models (LMs).[5] As mentioned, the out-domain LMs differ from previous works, e.g., (Axelrod et al., 2011), which employ mix-domain LMs. Here, we stress the difficulty in finding data to train *out-domain LMs* and present a solution based on *identifying pseudo out-domain data*.

- $P(D)$: the domain priors aim at modeling the percentage of relevant data that the learning framework induces. It can be estimated via phrase-level parameters but here we prefer sentence-level parameters:[6]

$$P(D) := \frac{\sum_{\langle \mathbf{e},\mathbf{f}\rangle \in C_{mix}} P(D \mid \mathbf{e}, \mathbf{f})}{\sum_D \sum_{\langle \mathbf{e},\mathbf{f}\rangle \in C_{mix}} P(D \mid \mathbf{e}, \mathbf{f})} \quad (5)$$

2.2 Training by invitation

Generally, our model can be viewed to have latent parameters $\Theta = \{\Theta_{D_0}, \Theta_{D_1}\}$. The training procedure seeks Θ that maximize the log-likelihood of the observed sentence pairs $\langle \mathbf{e}, \mathbf{f} \rangle \in C_{mix}$:

$$\mathcal{L} = \sum_{\langle \mathbf{e},\mathbf{f}\rangle \in C_{mix}} \log \sum_D P_{\Theta_D}(D, \mathbf{e}, \mathbf{f}). \quad (6)$$

It is obvious that there does not exist a closed-form solution for Equation 6 because of the existence of the log-term $\log \sum$. The EM algorithm (Dempster et al., 1977) comes as an alternative solution to fit the model. It can be seen to maximize $\mathcal{L}$ via block-coordinate ascent on a lower bound $\mathcal{F}(q, \Theta)$ using an auxiliary distribution $q(D \mid \mathbf{e}, \mathbf{f})$

$$\mathcal{F}(q, \Theta) = \sum_{\langle \mathbf{e},\mathbf{f}\rangle} \sum_D q(D \mid \mathbf{e}, \mathbf{f}) \log \frac{P_{\Theta_D}(D, \mathbf{e}, \mathbf{f})}{q(D \mid \mathbf{e}, \mathbf{f})} \quad (7)$$

where the inequality results, i.e., $\mathcal{L} \geq \mathcal{F}(q, \Theta)$, derived from $\log$ being concave and Jensen's inequality. We rewrite the Free Energy $\mathcal{F}(q, \Theta)$ (Neal and Hinton, 1999) as follows:

$$\begin{aligned} \mathcal{F} = & \sum_{\langle \mathbf{e},\mathbf{f}\rangle} \sum_D q(D \mid \mathbf{e}, \mathbf{f}) \log \frac{P_{\Theta_D}(D \mid \mathbf{e}, \mathbf{f})}{q(D \mid \mathbf{e}, \mathbf{f})} \\ & + \sum_{\langle \mathbf{e},\mathbf{f}\rangle} \sum_D q(D \mid \mathbf{e}, \mathbf{f}) \log P_\Theta(\mathbf{e}, \mathbf{f}) \\ & = \sum_{\langle \mathbf{e},\mathbf{f}\rangle} \log P_\Theta(\mathbf{e}, \mathbf{f}) \\ & - KL[q(D \mid \mathbf{e}, \mathbf{f}) \mid\mid P_{\Theta_D}(D \mid \mathbf{e}, \mathbf{f})], \end{aligned} \quad (8)$$

where $KL[\cdot \mid\mid \cdot]$ is the KL-divergence.

With the introduction of the KL-divergence, the alternating E and M steps for our EM algorithm are easily derived as

$$\begin{aligned} & \textbf{E-}step : q^{t+1} \\ & \text{argmax}_{q(D \mid \mathbf{e},\mathbf{f})} \mathcal{F}(q, \Theta^t) \; = \\ & \text{argmin}_{q(D \mid \mathbf{e},\mathbf{f})} KL[q(D|\mathbf{e}, \mathbf{f}) \mid\mid P_{\Theta_D^t}(D|\mathbf{e}, \mathbf{f})] \\ & = P_{\Theta_D^t}(D \mid \mathbf{e}, \mathbf{f}) \end{aligned} \quad (9)$$

$$\begin{aligned} & \textbf{M-}step : \Theta^{t+1} \\ & \text{argmax}_\Theta \mathcal{F}(q^{t+1}, \Theta) \; = \\ & \text{argmax}_\Theta \sum_{\langle \mathbf{e},\mathbf{f}\rangle} \sum_D q(D \mid \mathbf{e}, \mathbf{f}) \log P_{\Theta_D}(D, \mathbf{e}, \mathbf{f}) \end{aligned} \quad (10)$$

The iterative procedure is illustrated in Figure 1.[7] At the E-step, a guess for $P(D \mid \tilde{e}, \tilde{f})$ can be used to update $P_t(\tilde{f} \mid \tilde{e}, D)$ and $P_t(\tilde{e} \mid \tilde{f}, D)$ (i.e., using Eq. 1) and consequently $P_t(\mathbf{f} \mid \mathbf{e}, D)$ and $P_t(\mathbf{e} \mid \mathbf{f}, D)$ (i.e., using Eq. 4). These resulting table estimates, together with the domain-focused LMs and the domain priors are served as *expected counts* to update $P(D \mid \mathbf{e}, \mathbf{f})$.[8] At the M-step,

[4] We design our latent domain translation models with efficiency as our main concern. Future extensions could include the lexical and reordering sub-models (as suggested by an anonymous reviewer.)

[5] Relevance for identification or retrieval could be different from frequency or fluency. We leave this extension for future work.

[6] It should be noted that in most phrase-based SMT systems bilingual phrase probabilities are estimated heuristically from word alignmened data which often leads to overfitting. Estimating $P(D)$ from sentence-level parameters rather than from phrase-level parameters helps us avoid the overfitting which often accompanies phrase extraction.

[7] For simplicity, we ignore the LMs and prior models in the illustration in Fig. 1.

[8] Since we only use the in-domain corpus as priors to initilize the EM parameters, in technical perspective we do not want $P(D \mid \mathbf{e}, \mathbf{f})$ parameters to go too far off from the initialization. We therefore prefer the averaged style in practice, i.e., at the iteration n we update the $P(D \mid \mathbf{e}, \mathbf{f})$ parameters, $P^{(n)}(D|\mathbf{e}, \mathbf{f})$ as $\frac{1}{n}(P^{(n)}(D \mid \mathbf{e}, \mathbf{f}) + \sum_{i=1}^{n-1} P^{(i)}(D \mid \mathbf{e}, \mathbf{f}))$.

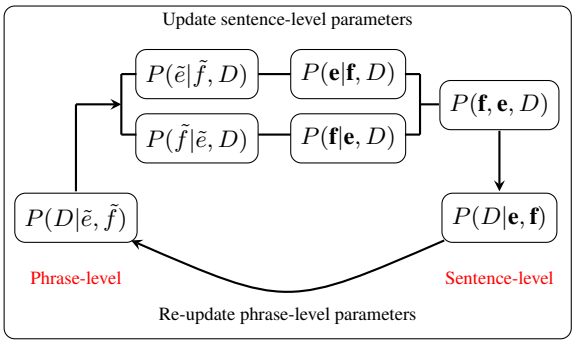


Figure 1: Our probabilistic invitation framework.

the new estimates for $P(D \mid \mathbf{e}, \mathbf{f})$ can be used to (softly) fill in the values of hidden variable D and estimate parameters $P(D \mid \tilde{e}, \tilde{f})$ and $P(D)$. The EM is guaranteed to converge to a local maximum of the likelihood under mild conditions (Neal and Hinton, 1999).

Before EM training starts we must provide a "reasonable" initial guess for $P(D \mid \tilde{e}, \tilde{f})$. We must also train the out-domain LMs, which needs the construction of *pseudo out-domain* data.[9] One simple way to do that is inspired by burn-in in sampling, under the guidance of an in-domain data set, C_{in} as prior. At the beginning, we train $P_t(\tilde{e} \mid \tilde{f}, D_1)$ and $P_t(\tilde{f} \mid \tilde{e}, D_1)$ for all phrases learned from C_{in}. We also train $P_t(\tilde{e} \mid \tilde{f})$ and $P_t(\tilde{f} \mid \tilde{e})$ for all phrases learned from C_{mix}. During burn-in we assume that the out-domain phrase-based models are the domain-confused phrase-based models, i.e., $P_t(\tilde{e} \mid \tilde{f}, D_0) \approx P_t(\tilde{e} \mid \tilde{f})$ and $P_t(\tilde{f} \mid \tilde{e}, D_0) \approx P_t(\tilde{f} \mid \tilde{e})$. We isolate all the LMs and the prior models from our model, and apply a single EM iteration to update $P(D \mid \mathbf{e}, \mathbf{f})$ based on those domain-focused models $P_t(\tilde{e} \mid \tilde{f}, D)$ and $P_t(\tilde{f} \mid \tilde{e}, D)$.

In the end, we use $P(D \mid \mathbf{e}, \mathbf{f})$ to fill in the values of hidden variable D in C_{mix}, so it provides us with an initialization for $P(D \mid \tilde{e}, \tilde{f})$. Subsequently, we also rank sentence pairs in $\mathcal{C}_{mix}$ with $P(D_1 \mid \mathbf{e}, \mathbf{f})$ and select a subset of smallest scoring pairs as a *pseudo out-domain subset* to train $P_{lm}(\mathbf{e} \mid D_0)$ and $P_{lm}(\mathbf{f} \mid D_0)$. Once the latent domain-focused LMs have been trained, the LM probabilities stay *fixed* during EM. Crucially, it is important to scale the probabilities of the four LMs to make them comparable: we normalize the probability that a LM assigns to a sentence by the total probability this LM assigns to all sentences in $\mathcal{C}_{mix}$.

[9]The in-domain LMs $P_{lm}(\mathbf{e} \mid D_1)$ and $P_{lm}(\mathbf{f} \mid D_1)$ can be simply trained on the source and target sides of $\mathcal{C}_{in}$ respectively.

3 Intrinsic evaluation

We evaluate the ability of our model to retrieve "hidden" in-domain data in a large mix-domain corpus, i.e., we hide some in-domain data in a large mix-domain corpus. We weigh sentence pairs under our model with $P(D_1 \mid \tilde{e}, \tilde{f})$ and $P(D_1 \mid \mathbf{e}, \mathbf{f})$ respectively. We report *pseudo*-precision/recall at the *sentence-level* using a range of cut-off criteria for selecting the top scoring instances in the mix-domain corpus. A good relevance model expects to score higher for the hidden in-domain data.

Baselines Two standard perplexity-based selection models in the literature have been implemented as the baselines: cross-entropy difference (Moore and Lewis, 2010) and bilingual cross-entropy difference (Axelrod et al., 2011), investigating their ability to retrieve the hiding data as well. Training them over the data to learn the sentences with their relevance, we then rank the sentences to select top of pairs to evaluate the *pseudo*-precision/recall at the *sentence-level*.

Results We use a mix-domain corpus $\mathcal{C}_g$ of 770K sentence pairs of different genres.[10] There is also a Legal corpus of 183K pairs that serves as the in-domain data. We create $\mathcal{C}_{mix}$ by selecting an arbitrary 83K pairs of in-domain pairs and adding them to $\mathcal{C}_g$ (the hidden in-domain data); we use the remaining 100k in-domain pairs as $\mathcal{C}_{in}$.

To train the baselines, we construct interpolated 4-gram Kneser-Ney LMs using BerkeleyLM (Pauls and Klein, 2011). Training our model on the data takes six EM-iterations to converge.[11]

[10]Count of *sentence pairs*: European Parliament (Koehn, 2005): 183,793; Pharmaceuticals: 190,443, Software: 196,168, Hardware: 196,501.

[11]After the fifth EM iteration we do not observe any significant increase in the likelihood of the data. Note that we use the same setting as for the baselines to train the latent domain-focused LMs for use in our model – interpolated 4-gram Kneser-Ney LMs using BerkeleyLM. This training setting is used for all experiments in this work.

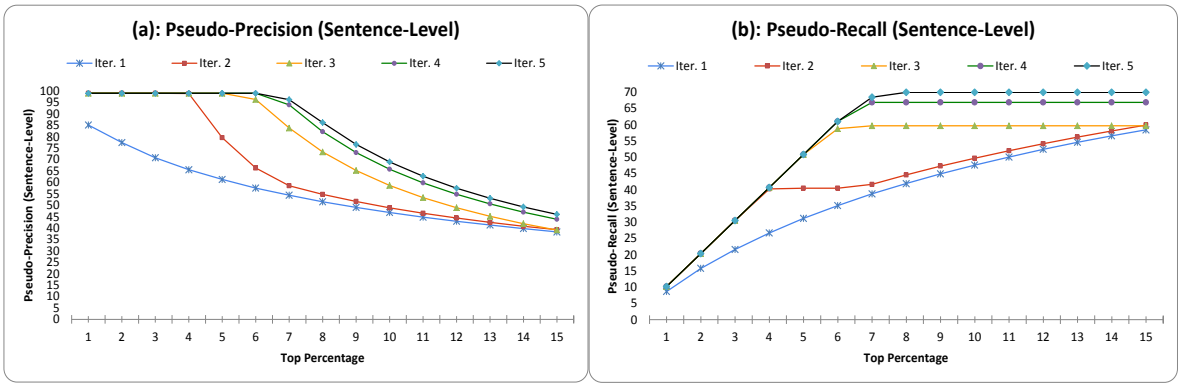


Figure 2: Intrinsic evaluation.

Fig. 2 helps us examine how the pseudo sentence invitation are done during each EM iteration. For later iterations we observe a better pseudo-precision and pseudo-recall at sentence-level (Fig. 2(a), Fig. 2(b)). Fig. 2 also reveals a good learning capacity of our learning framework. Nevertheless, we observe that the baselines do not work well for this task. This is not new, as pointed out in our previous work (Cuong and Sima'an, 2014).

Which component type contributes more to the performance, the latent domain language models or the latent domain translation models? Further experiments have been carried on to neutralize each component type in turn and build a selection system with the rest of our model parameters. It turns out that the latent domain translation models are crucial for performance for the learning framework, while the latent domain LMs make a far smaller yet substantial contribution. We refer readers to our previous work (Cuong and Sima'an, 2014), which provides detail analysis of the data selection problem.

4 Translation experiments: Setting

Data We use a mix-domain corpus consisting of $4M$ sentence pairs, collected from multiple resources including EuroParl (Koehn, 2005), Common Crawl Corpus, UN Corpus, News Commentary. As in-domain corpus we use "Consumer and Industrial Electronics" manually collected by Translation Automation Society (TAUS.com). The corpus statistics are summarized in Table 1.

System We train a standard state-of-the-art phrase-based system, using it as the baseline.[12] There are three main kinds of features for the translation model in the baseline - phrase-based translation features, lexical weights (Koehn et al., 2003) and lexicalized reordering features (Koehn et al., 2005).[13] Other features include the penalties for word, phrase and distance-based reordering.

			English	Spanish
	$\mathbf{C_{mix}}$	Sents	**4M**	
		Words	$113.7M$	127.1M
Domain: Electronics	$\mathbf{C_{in}}$	Sents	**109K**	
		Words	$1,485,558$	$1,685,716$
	Dev	Sents	984	
		Words	13130	$14,955$
	Test	Sents	982	
		Words	$13,493$	$15,392$

Table 1: The data preparation.

The mix-domain corpus is word-aligned using GIZA++ (Och and Ney, 2003) and symmetrized with *grow(-diag)-final-and* (Koehn et al., 2003). We limit phrase length to a maximum of seven words. The LMs are interpolated 4-grams with Kneser-Ney, trained on $2.2M$ English sentences from Europarl augmented with $248.8K$ sentences from News Commentary Corpus (WMT 2013). We tune the system using k-best batch MIRA (Cherry and Foster, 2012). Finally, we use Moses

[12] We use Stanford Phrasal - a standard state-of-the-art phrase-based translation system developed by Cer et al. (2010).

[13] The lexical weights and the lexical reordering features will be described in more detail in Section 6.

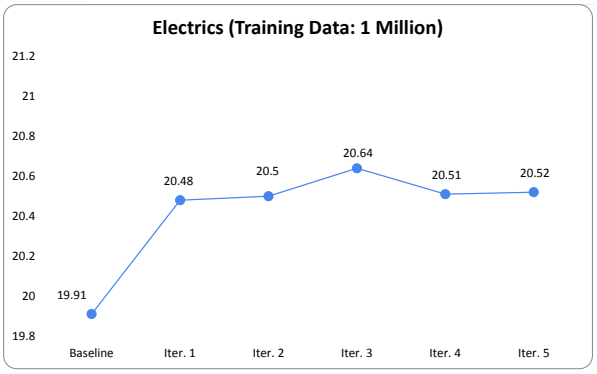


Figure 3: BLEU averaged over multiple runs.

(Koehn et al., 2007) as decoder.[14]

We report BLEU (Papineni et al., 2002), METEOR 1.4 (Denkowski and Lavie, 2011) and TER (Snover et al., 2006), with statistical significance at 95% confidence interval under paired bootstrap re-sampling (Press et al., 1992). For every system reported, we run the optimizer at least three times, before running MultEval (Clark et al., 2011) for resampling and significance testing.

Outlook In Section 5 we examine the effect of training only the latent domain-focused phrase table using our model. In Section 6 we proceed further to estimate also latent domain-focused lexical weights and lexicalized reordering models, examining how they incrementally improve the translation as well.

5 Adapting phrase table only

Here we investigate the effect of adapting the phrase table only; we will delay adapting the lexical weights and lexicalized reordering features to Section 6. We build a phrase-based system with the usual features as the baseline, including two bi-directional phrase-based models, plus the penalties for word, phrase and distortion. We also build a latent domain-focused phrase-based system with the two bi-directional latent phrase-based models, and the standard penalties described above.

We explore training data sizes $1M$, $2M$ and $4M$ sentence pairs. Three baselines are trained yielding $95.77M$, $176.29M$ and $323.88M$ phrases respectively. We run 5 EM iterations to train our learning framework. We use the parameter estimates for $P(D \mid \tilde{e}, \tilde{f})$ derived at each EM iteration to train our latent domain-focused phrase-based systems. Fig. 3 presents the results (in BLEU) at each iteration in detail for the case of $1M$ sentence pairs. Similar improvements are observed for METEOR and TER. Here, we consistently observe improvements at p-value $= 0.0001$ for all cases.

It should be noted that when doubling the training data to $2M$ and $4M$, we observe the similar results.

Finally, for all cases we report their best result in Table 2. Here, note how the improvement could be gained when doubling the training data.

Data	System	Avg	Δ	p-value
$1M$	Baseline	19.91	–	–
	Our System	20.64	**+0.73**	**0.0001**
$2M$	Baseline	20.54	–	–
	Our System	21.41	**+0.87**	**0.0001**
$4M$	Baseline	21.44	–	–
	Our System	22.62	**+1.18**	**0.0001**

Table 2: BLEU averaged over multiple runs.

It is also interesting to consider the average entropy of phrase table entries in the domain-confused systems, i.e.,

$$\frac{-\sum_{\langle\tilde{e},\tilde{f}\rangle} p_t(\tilde{e}|\tilde{f}) \log p_t(\tilde{e}|\tilde{f})}{\text{number of phrases}\langle\tilde{e},\tilde{f}\rangle}$$

against that in the domain-focused systems

$$\frac{-\sum_{\langle\tilde{e},\tilde{f}\rangle} p_t(\tilde{e}|\tilde{f}, D_1) \log p_t(\tilde{e}|\tilde{f}, D_1)}{\text{number of phrases}\langle\tilde{e},\tilde{f}\rangle}.$$

Following (Hasler et al., 2014) in Table 3 we also show that the entropy decreases significantly in

[14] While we implement the latent domain phrase-based models using Phrasal for some advantages, we prefer to use Moses for decoding.

the adapted tables in all cases, which indicates that the distributions over translations of phrases have become sharper.

Baseline	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5
0.210	0.187	0.186	0.185	0.185	0.184

Table 3: Average entropy of distributions.

In practice, the third iteration systems usually produce best translations. This is somewhat expected because as EM invites more pseudo in-domain pairs in later iterations, it sharpens the estimates of $P(D_1 \mid \tilde{e}, \tilde{f})$, making pseudo out-domain pairs tend to 0.0. Table 4 shows the percentage of entries with $P(D_1 \mid \tilde{e}, \tilde{f}) < 0.01$ at every iteration, e.g., 34.52% at the fifth iteration. This induced schism in C_{mix} diminishes the difference between the relevance scores for certain sentence pairs, limiting the ability of the latent phrase-based models to further discriminate in the gray zone.

Entries	$P(D_1 \mid \tilde{f}, \tilde{e}) < 0.01$
Iter. 1	22.82%
Iter. 2	27.06%
Iter. 3	30.07%
Iter. 4	32.47%
Iter. 5	34.52%

Table 4: Phrase analyses.

Finally, to give a sense of the improvement in translation, we (randomly) select cases where the systems produce different translations and present some of them in Table 5. These examples are indeed illuminating, e.g., "*can reproduce signs of audio*"/"*can play signals audio*", "*password teacher*"/"*password master*", revealing thoroughly the benefit derived from adapting the phrase models from being domain-confused to being domain-focused. Table 6 presents phrase table entries, i.e., $p_t(e \mid f)$ and $p_t(e \mid f, D_1)$, for the "*can reproduce signs of audio*"/"*can play signals audio*" example.

6 Fully adapted translation model

The preceding experiments reveal that adapting the phrase tables significantly improves translation performance. Now we also adapt the lexical and reordering components. The result is a fully adapted, domain-focused, phrase-based system.

	señales		**reproducir**	
Entries	**signals**	**signs**	**play**	**reproduce**
Baseline	0.29	0.36	0.15	0.20
Iter. 1	0.36	0.23	0.29	0.16
Iter. 2	0.37	0.19	0.32	0.17
Iter. 3	0.37	0.17	0.34	0.16
Iter. 4	0.37	0.16	0.36	0.16
Iter. 5	0.37	0.15	0.37	0.16

Table 6: Phrase entry examples.

Briefly, the lexical weights provide smooth estimates for the phrase pair based on word translation scores $P(e \mid f)$ between pairs of words $\langle e, f \rangle$, i.e., $P(e \mid f) = \frac{c(e,f)}{\sum_e c(e,f)}$ (Koehn et al., 2003). Our latent domain-focused lexical weights, on the other hand, are estimated according to $P(e \mid f,\ D_1)$, i.e., $P(e \mid f,\ D_1) = \frac{P(e \mid f)P(D_1 \mid e,\ f)}{\sum_f P(e \mid f)P(D_1 \mid e,\ f)}$.

The lexicalized reordering models with orientation variable O, $P(O \mid \tilde{e},\ \tilde{f})$, model how likely a phrase $\langle \tilde{e}, \tilde{f} \rangle$ directly follows a previous phrase ($monotone$), swaps positions with it ($swap$), or is not adjacent to it ($discontinous$) (Koehn et al., 2005). We make these domain-focused:

$$P(O \mid \tilde{e},\ \tilde{f}, D_1) = \frac{P(O \mid \tilde{e},\ \tilde{f})P(D_1 \mid O, \tilde{e},\ \tilde{f})}{\sum_O P(O \mid \tilde{e},\ \tilde{f})P(D_1 \mid O, \tilde{e},\ \tilde{f})} \quad (11)$$

Estimating $P(D_1 \mid O,\ \tilde{e},\ \tilde{f})$ and $P(D_1 \mid e,\ f)$ is similar to estimating $P(D_1 \mid \tilde{e},\ \tilde{f})$ and hinges on the estimates of $P(D_1 \mid \mathbf{e},\ \mathbf{f})$ during EM.

The baseline for the following experiments is a standard state-of-the-art phrase-based system, including two bi-directional phrase-based translation features, two bi-directional lexical weights, six lexicalized reordering features, as well as the penalties for word, phrase and distortion. We develop three kinds of domain-adapted systems that are different at their adaptation level to fit the task. The first (**Sys. 1**) adapts only the phrase-based models, using the same lexical weights, lexicalized reordering models and other penalties as the baseline. The second (**Sys. 2**) adapts also the lexical weights, fixing all other features as the baseline. The third (**Sys. 3**) adapts both the phrase-based models, lexical weights and lexicalized re-

	Translation Examples
Input	*El reproductor puede reproducir señales de audio grabadas en mix-mode cd, cd-g, cd-extra y cd text.*
Reference	*The player **can play back audio signals** recorded in mix-mode cd, cd-g, cd-extra and cd text.*
Baseline	*The player **can reproduce signs of audio** recorded in mix-mode cd, cd-g, cd-extra and cd text.*
Our System	*The player **can play signals audio** recorded in mix-mode cd, cd-g, cd-extra and cd text.*
Input	*Se puede crear un archivo autodescodificable cuando el archivo codificado se abre con la contraseña maestra.*
Reference	*A self-decrypting file can be created when the encrypted file is **opened with** the **master password**.*
Baseline	*To create an file autodescodificable when the file codified **commenced with** the **password teacher**.*
Our System	*You can create an archive autodescodificable when the file codified **opens with** the **password master**.*
Input	*Repite todas las pistas (únicamente cds de vídeo sin pbc)*
Reference	*Repeat **all tracks** (non-pbc video cds only)*
Baseline	*Repeated **all avenues** (only cds video without pbc)*
Our System	*Repeated **all the tracks** (only cds video without pbc)*

Table 5: Translation examples yielded by a domain-confused phrase-based system (**the baseline**) and a domain-focused phrase-based system (**our system**).

ordering models[15], fixing other penalties as the baseline.

Metric	System	Avg	Δ	p-value
Consumer and Industrial Electronics				
(In-domain: $109K$ pairs; Dev: 982 pairs; Test: 984 pairs)				
BLEU	Baseline	22.9	–	–
	Sys. 1	23.4	+0.5	0.008
	Sys. 2	23.9	+1.0	0.0001
	Sys. 3	**24.0**	**+1.1**	**0.0001**
METEOR	Baseline	30.0	–	–
	Sys. 1	30.4	+0.4	0.0001
	Sys. 2	30.8	+0.8	0.0001
	Sys. 3	**30.9**	**+0.9**	**0.0001**
TER	Baseline	59.5	–	–
	Sys. 1	58.8	-0.7	0.0001
	Sys. 2	58.0	-1.5	0.0001
	Sys. 3	**57.9**	**-1.6**	**0.0001**

Table 7: Metric scores for the systems, which are averages over multiple runs.

Table 7 presents results for training data size of $4M$ parallel sentences. It shows that the fully domain-focused system (**Sys. 3**) significantly improves over the baseline. The table also shows that the latent domain-focused phrase-based models and lexical weights are crucial for the improved performance, whereas adapting the reordering models makes a far smaller contribution.

Finally we also apply our approach to other tasks where the relation between their in-domain data and the mix-domain data varies substantially. Table 8 presents their in-domain, tuning and test data in detail, as well as the translation results over them. It shows that the fully domain-focused systems consistently and significantly improve the translation accuracy for all the tasks.

7 Combining multiple models

Finally, we proceed further to test our latent domain-focused phrase-based translation model on standard domain adaptation. We conduct experiments on the task "Professional & Business Services" as an example.[16] For standard adaptation we follow (Koehn and Schroeder, 2007) where we pass multiple phrase tables directly to the Moses decoder and tune them together. For baseline we combine the standard phrase-based system trained on C_{mix} with the one trained on the in-domain data C_{in}. We also combine our latent domain-focused phrase-based system with the one trained on C_{in}. Table 9 presents the results showing that combining our domain-focused system adapted from C_{mix} with the in-domain model outperforms the baseline.

[15]We run three EM iterations to train our invitation framework, and then use the parameter estimates for $P(D_1 \mid \tilde{e}, \tilde{f})$, $P(D_1 \mid e, f)$ and $P(D_1 \mid O, \tilde{e}, \tilde{f})$ to train these domain-focused features. We adopt this training setting for all other different tasks in the sequel.

[16]We choose this task for additional experiments because it has very small in-domain data ($23K$). This is supposed to make adaptation difficult because of the robust large-scale systems trained on C_{mix}.

Metric	System	Avg	Δ	p-value
Professional & Business Services				
(In-domain: $23K$ pairs; Dev: $1,000$ pairs; Test: 998 pairs)				
BLEU	Baseline	22.0	–	–
	Our System	23.1	**+1.1**	**0.0001**
METEOR	Baseline	30.8	–	–
	Our System	31.4	**+0.6**	**0.0001**
TER	Baseline	58.0	–	–
	Our System	56.6	**-1.4**	**0.0001**
Financials				
(In-domain: $31K$ pairs; Dev: $1,000$ pairs; Test: $1,000$ pairs)				
BLEU	Baseline	31.1	–	–
	Our System	31.8	**+0.7**	**0.0001**
METEOR	Baseline	36.3	–	–
	Our System	36.6	**+0.3**	**0.0001**
TER	Baseline	48.8	–	–
	Our System	48.3	**-0.5**	**0.0001**
Computer Hardware				
(In-domain: $52K$ pairs; Dev: $1,021$ pairs; Test: $1,054$ pairs)				
BLEU	Baseline	24.6	–	–
	Our System	25.3	**+0.7**	**0.0001**
METEOR	Baseline	32.4	–	–
	Our System	33.1	**+0.7**	**0.0001**
TER	Baseline	56.4	–	–
	Our System	55.0	**-1.4**	**0.0001**
Computer Software				
(In-domain: $65K$ pairs; Dev: $1,100$ pairs; Test: $1,000$ pairs)				
BLEU	Baseline	27.4	–	–
	Our System	28.3	**+0.9**	**0.0001**
METEOR	Baseline	34.0	–	–
	Our System	34.7	**+0.7**	**0.0001**
TER	Baseline	51.7	–	–
	Our System	50.6	**-1.1**	**0.0001**
Pharmaceuticals & Biotechnology				
(In-domain: $85K$ pairs; Dev: 920 pairs; Test: $1,000$ pairs)				
BLEU	Baseline	31.6	–	–
	Our System	32.4	**+0.8**	**0.0001**
METEOR	Baseline	34.0	–	–
	Our System	34.4	**+0.4**	**0.0001**
TER	Baseline	51.4	–	–
	Our System	50.6	**-0.8**	**0.0001**

Table 8: Metric scores for the systems, which are averages over multiple runs.

Metric	System	Avg	Δ	p-value
Professional & Business Services				
(In-domain: $23K$ pairs; Dev: $1,000$ pairs; Test: 998 pairs)				
BLEU	In-domain	46.5	–	–
	+ Mix-domain	46.6	–	–
	+ Our system	47.9	**+1.3**	**0.0001**
METEOR	In-domain	39.8	–	–
	+ Mix-domain	40.1	–	–
	+ Our System	41.1	**+1.0**	**0.0001**
TER	In-domain	38.2	–	–
	+ Mix-domain	38.0	–	–
	+ Our System	36.9	**-1.1**	**0.0001**

Table 9: Domain adaptation experiments. Metric scores for the systems, which are averages over multiple runs.

8 Related work

A distantly related, but clearly complementary, line of research focuses on the role of document topics (Eidelman et al., 2012; Zhang et al., 2014; Hasler et al., 2014). An off-the-shelf Latent Dirichlet Allocation tool is usually used to infer document-topic distributions. On one hand, this setting may not require in-domain data as prior. On the other hand, it requires meta-information (e.g., document information).

Part of this work (the latent sentence-relevance models) relates to data selection (Moore and Lewis, 2010; Axelrod et al., 2011), where sentence-relevance weights are used for hard-filtering rather than weighting. The idea of using sentence-relevance estimates for phrase-relevance estimates relates to Matsoukas et al. (2009) who estimate the former using meta-information over documents as main features. In contrast, our work overcomes the mutual dependence of sentence and phrase estimates on one another by training both models in tandem.

Adaptation using small in-domain data has a different but complementary goal to another line of research aiming at combining a domain-adapted system with the another trained on the in-domain data (Koehn and Schroeder, 2007; Bisazza et al., 2011; Sennrich, 2012; Razmara et al., 2012; Sennrich et al., 2013). Our work is somewhat related to, but markedly different from, phrase pair weighting (Foster et al., 2010). Finally, our latent domain-focused phrase-based models and invitation training paradigm can be seen to shift attention from adaptation to making explicit the role of domain-focused models in SMT.

9 Conclusion

We present a novel approach for in-domain focused training of a phrase-based system on a mix-of-domain corpus by using prior distributions from a small in-domain corpus. We derive an EM training algorithm for learning latent domain relevance models for the phrase- and sentence-levels in tandem. We also show how to overcome the difficulty of lack of explicit out-domain data by bootstrapping pseudo out-domain data.

In future work, we plan to explore generative Bayesian models as well as discriminative learning approaches with different ways for estimat-

ing the latent domain relevance models. We hypothesize that bilingual, but also monolingual, relevance models can be key to improved performance.

Acknowledgements

We thank Ivan Titov for stimulating discussions, and three anonymous reviewers for their comments on earlier versions. The first author is supported by the EXPERT (EXPloiting Empirical appRoaches to Translation) Initial Training Network (ITN) of the European Union's Seventh Framework Programme. The second author is supported by VICI grant nr. 277-89-002 from the Netherlands Organization for Scientific Research (NWO). We thank TAUS for providing us with suitable data.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 355–362, Stroudsburg, PA, USA. Association for Computational Linguistics.

Arianna Bisazza, Nick Ruiz, and Marcello Federico. 2011. Fill-up versus interpolation methods for phrase-based smt adaptation. In *IWSLT*, pages 136–143.

Daniel Cer, Michel Galley, Daniel Jurafsky, and Christopher D. Manning. 2010. Phrasal: A toolkit for statistical machine translation with facilities for extraction and incorporation of arbitrary model features. In *Proceedings of the NAACL HLT 2010 Demonstration Session*, HLT-DEMO '10, pages 9–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 427–436, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 176–181, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hoang Cuong and Khalil Sima'an. 2014. Latent domain translation models in mix-of-domains haystack. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1928–1939, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 85–91, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vladimir Eidelman, Jordan Boyd-Graber, and Philip Resnik. 2012. Topic models for dynamic translation model adaptation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 115–119, Stroudsburg, PA, USA. Association for Computational Linguistics.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 451–459, Stroudsburg, PA, USA. Association for Computational Linguistics.

Guillem Gascó, Martha-Alicia Rocha, Germán Sanchis-Trilles, Jesús Andrés-Ferrer, and Francisco Casacuberta. 2012. Does more data always yield better translations? In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 152–161, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eva Hasler, Phil Blunsom, Philipp Koehn, and Barry Haddow. 2014. Dynamic topic adaptation for phrase-based mt. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 328–337, Gothenburg, Sweden, April. Association for Computational Linguistics.

Ann Irvine, John Morgan, Marine Carpuat, Daume Hal III, and Dragos Munteanu. 2013. Measuring machine translation errors in new domains. pages 429–440.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 224–227, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *International Workshop on Spoken Language Translation*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Spyros Matsoukas, Antti-Veikko I. Rosti, and Bing Zhang. 2009. Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 708–717, Stroudsburg, PA, USA. Association for Computational Linguistics.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 220–224, Stroudsburg, PA, USA. Association for Computational Linguistics.

Radford M. Neal and Geoffrey E. Hinton. 1999. Learning in graphical models. chapter A View of the EM Algorithm That Justifies Incremental, Sparse, and Other Variants, pages 355–368. MIT Press, Cambridge, MA, USA.

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

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Comput. Linguist.*, 30(4):417–449, December.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Adam Pauls and Dan Klein. 2011. Faster and smaller n-gram language models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 258–267, Stroudsburg, PA, USA. Association for Computational Linguistics.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 1992. *Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA.

Majid Razmara, George Foster, Baskaran Sankaran, and Anoop Sarkar. 2012. Mixing multiple translation models in statistical machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 940–949, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rico Sennrich, Holger Schwenk, and Walid Aransa. 2013. A multi-domain translation model framework for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 832–840, Sofia, Bulgaria, August. Association for Computational Linguistics.

Rico Sennrich. 2012. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 539–549, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Min Zhang, Xinyan Xiao, Deyi Xiong, and Qun Liu. 2014. Topic-based dissimilarity and sensitivity models for translation rule selection. *Journal of Artificial Intelligence Research*, 50(1):1–30.

Translation Rules with Right-Hand Side Lattices

Fabien Cromières
Japan Science and Technology Agency
Kawaguchi-shi
Saitama 332-0012
fabien@pa.jst.jp

Sadao Kurohashi
Graduate School of Informatics
Kyoto University
Kyoto 606-8501
kuro@i.kyoto-u.ac.jp

Abstract

In Corpus-Based Machine Translation, the search space of the translation candidates for a given input sentence is often defined by a set of (cycle-free) context-free grammar rules. This happens naturally in Syntax-Based Machine Translation and Hierarchical Phrase-Based Machine Translation (where the representation will be the set of the target-side half of the synchronous rules used to parse the input sentence). But it is also possible to describe Phrase-Based Machine Translation in this framework. We propose a natural extension to this representation by using lattice-rules that allow to easily encode an exponential number of variations of each rules. We also demonstrate how the representation of the search space has an impact on decoding efficiency, and how it is possible to optimize this representation.

1 Introduction

A popular approach to modern Machine Translation is to decompose the translation problem into a modeling step and a search step. The modeling step will consist in defining implicitly a set of possible translations $\mathcal{T}$ for each input sentence. Each translation in $\mathcal{T}$ being associated with a real-valued *model score*. The search step will then consist in finding the translation in $\mathcal{T}$ with the highest model score. The search is non-trivial because it is usually impossible to enumerate all members of $\mathcal{T}$ (its cardinality being typically exponentially dependent on the size of the sentence to be translated).

Since at least (Chiang, 2007), a common way of representing $\mathcal{T}$ has been through a cycle-free context-free grammar. In such a grammar, $\mathcal{T}$ is represented as a set of context-free rules such as can be seen on figure 1. These rules themselves can be generated by the modeling step through the use of phrase tables, synchronous parsing, tree-to-string rules, etc. If the model score of each translation is taken to be the sum of *rule scores* independently given to each rule, the search for the optimal translation is easy with some classic dynamic programming techniques.

However, if the model score is going to take into account informations such as the language model score of each sentence, it cannot be expressed in such a way. Since the language model score has proven empirically to be a very good source of information, (Chiang, 2007) proposed an approximate search algorithm called *cube pruning*.

We propose here to represent $\mathcal{T}$ using context-free lattice-rules such as shown in figure 2. This allows us to compactly encode a large number of rules. One benefit is that it adds flexibility to the modeling step, making it easier: many choices such as whether or not a function word should be included, the relative position of words and non-terminal in the translation, as well as morphological variations can be delegated to the search step by encoding them in the lattice rules. While it is true that the same could be achieved by an explicit enumeration, lattice rules make this easier and more efficient.

In particular, we show that a decoding algorithm working with such lattice rules can be more efficient than one working directly on the enumeration of the rules encoded in the lattice.

A distinct but related idea of this paper is to consider how transforming the structure of the rules defining $\mathcal{T}$ can lead to improvements

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 577–588,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

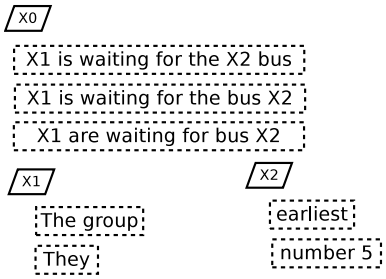


Figure 1: A simple cycle-free context grammar describing a set of possible translations.

in the speed/memory performances of the decoding. In particular, we propose a method to merge and reduce the size of the lattice rules and show that it translates into better performances at decoding time.

In this paper, we will first define more precisely our concept of lattice-rules, then try to give some motivation for them in the context of a tree-to-tree MT system (section 3). In section 4, we then propose an algorithm for preprocessing a representation given in a lattice-rule form that allows for more efficient search. In section 5, we describe a decoding algorithm specially designed for handling lattice-rules. In section 6, we perform some experiments demonstrating the merit of our approach.

2 Notations and Terminology

Here, we define semi-formally the terms we will use in this paper. We assume knowledge of the classic terminology of graph theory and context-free grammar.

2.1 Expansion rules

A *flat expansion rule* is the association of a non-terminal and a "flat" right hand side that we note *RHS*. A *flat RHS* is a sequence of words and non-terminal. See figure 1 for an example of a set of flat expansion rules.

A set of expansion rules is often produced in Hierarchical or Syntax-Based MT, by parsing with synchronous grammars or otherwise. In such a case, the set of rules define a representation of the (weighted) set of possible translations $\mathcal{T}$ of an input sentence.

2.2 Lattice

In the general sense, a lattice can be described as a labeled directed acyclic graph. More precisely, the type of lattice that we consider in this work is such that:

- Edges are labeled by either a word, a non-terminal or an epsilon (ie. an empty string).
- Vertices are only labeled by a unique id by which they can be designated.

Additionally, edges can also be labeled by a real-valued *edge score* and some real-valued *edge features*. Alternatively, a lattice could also be seen as an acyclic Finite State Automaton, with vertices and edges corresponding to states and transitions in the FSA terminology.

For simplicity, we also set the constraint that each lattice has a unique "start" vertex labeled v_S from which each vertex can be reached and a unique "end" vertex v_E that can be reached from each vertex. Each path from v_S to v_E define thus a flat RHS, with score and features obtained by summing the score and features of each edge of the path.

A *lattice expansion rule* is similar to a flat expansion rule, but with the RHS being a lattice. Thus a set of lattice expansion rules can also define a set of possible translations $\mathcal{T}$ of an input sentence.

For a given lattice $\mathcal{L}$, we will often note $v \in \mathcal{L}$ a vertex of $\mathcal{L}$ and $e : v_1 \to v_2 \in \mathcal{L}$ an edge of $\mathcal{L}$ going from vertex v_1 to vertex v_2.

Figures 2 and 3 show examples of such lattices.

2.3 Translation set and Representations

We note $\mathcal{T}$ a set of weighted sentences. $\mathcal{T}$ is intended as representing the set of scored translation candidates generated by a MT system for a given input sentence. As is customary in Corpus-Based MT literature, we will call *decoding* the process of searching for the translation with highest score in $\mathcal{T}$.

A representation of $\mathcal{T}$, noted $\mathcal{R}_T$ is a set of rules in a given formalism that implicitly define $\mathcal{T}$. As we mentioned earlier, in MT, $\mathcal{R}_T$ is often a set of cycle-free context-free grammar rules.

In this paper, we consider representations $\mathcal{R}_T$ consisting in a set of lattice expansion rules. With normal context-free grammar, it is usually necessary that a non-terminal is the

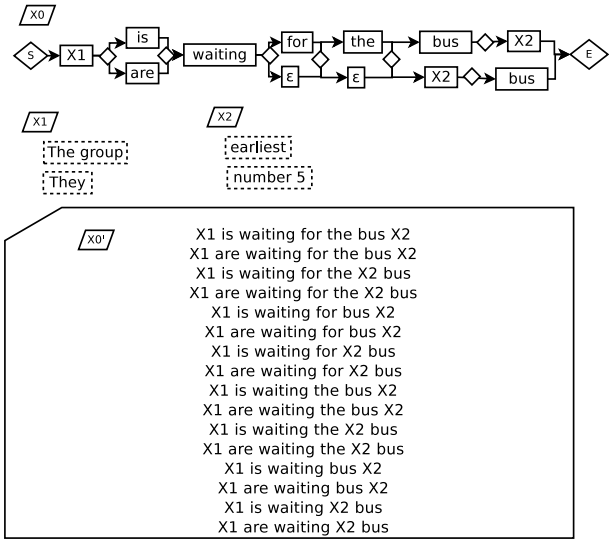


Figure 2: A simple example of lattice rule for non-terminal X0. The lower part list the set of "flat" rules that would be equivalent to the ones expressed by the lattice.

left-hand side of several rules. Using lattice expansion rules, however, it is not necessary, as one lattice RHS can encode an arbitrary number of flat rules (see for example the RHS of X0 in figure 3). Therefore, we set the constraint that there is only one lattice expansion rule for each left-hand non-terminal. And we will note unambiguously $RHS(X)$ the lattice that is the right hand side of this rule.

3 Motivation

3.1 Setting

This work was developed mainly in the context of a syntactic-dependency-based tree-to-tree translation system described in (Richardson et al., 2014). Although it is a tree-to-tree system, we simplify the decoding step by "flattening" the target-side tree translation rules into string expansion rules (keeping track of the dependency structure in state features). Thus our setting is actually quite similar to that of many tree-to-string and string-to-string systems. Aiming at simplicity and generality, we will set aside the question of target-side syntactic information and only describe our algorithms in a "tree-to-string" setting. We will also consider a n-gram language model score as our only stateful non-local feature.

However, this tree-to-tree original setting should be kept in mind, in particular when we describe the issue of the relative position of heads and dependents in section 3.2.2, as such issues do not appear as commonly in "X-to-string" settings.

3.2 Rule ambiguities

Expansion rules are typically created by matching part of the input sentence with some aligned example bilingual sentence. The alignment (and the linguistic structure of the phrase in the case of Syntax-Based Machine Translation) is then used to produce the target-side rule. However, it is often the case that it is difficult to fully specify a rule from an example. Such cases often come from two main reasons:

- Imperfect knowledge (eg. it is unclear whether a given unaligned word should belong to the translation)
- Context dependency (eg. the question of whether "to be" should be in plural form or not, depending on its subject in the constructed translation).

In both situation, it seems like it would be better to delay the full specification of the rule until decoding time, when the decoder can have access to the surrounding context of the rule and make a more informed choice. In particular, we can expect features such as language model or governor-dependent features (in the case of tree-to-tree Machine translation) to help remove the ambiguities.

We detail some cases for which we encode variations as lattice-rule.

3.2.1 Non-aligned words

When rules are extracted from aligned examples, we often find some target words which are not aligned to any source-side word and for which it is difficult to decide whether or not they should be included in the rule. Such words are often function words that do not have an equivalent in the source language. In Japanese-English translations, for example, articles such as "a" and "the" do not typically have equivalent in the Japanese side, and their necessity in the final sentence will often be a matter of context. We can make these edges

optionals by doubling them with an epsilon-edge. Different weights and features can be given to the epsilon edges to balance the tendency of the decoder to skip edges. In figure 2, this is illustrated by the epsilon edges allowing to skip "for" and "the"

3.2.2 Non-terminal positions

In the context of our tree-to-tree translation system, we often find that we know which target word should be the governor of a given non-terminal, but that we are unsure of the order of the words and non-terminals sharing a common governor. It can be convenient to represent such ambiguities in a lattice format as shown in figure 2. In this figure, one can see that the RHS of $X0$ encode two possible ordering for the word "bus" and the non-terminal $X2$.

3.2.3 Word variations

Linguistics phenomenons such as morphological variations can naturally create many minor problems in the setting of Corpus-Based Translation. Especially if the variations in the target language have no equivalence in the source language. An example of this in Japanese-English translation is the fact that verbs in Japanese are "plural-independent", while the verb "to be" in English is not. Therefore, a RHS that is a candidate for translating a large part of a Japanese input sentence can easily use one of the variant of "to be" that is not consistent with the full sentence. To solve this, for each edge corresponding to the words "is" or "are", we add an alternative edge with the same start and end vertices as the other word. The decoder will then be able to choose the edge that gives the best language model score. The same can be done, for example, for the article "a/an". Figure 2 provides an example of this, with two edges "is" and "are" in the RHS of X0.

Alternative edges can be labeled with different weights and features to tune the tendency of the decoder to choose a morphological variation.

While such variations could be fixed in a post-processing step, we feel it is a better option to let the decoder be aware of the possible options, lest it would discard rules due to language model considerations when these rules could actually have been useful with a simple change.

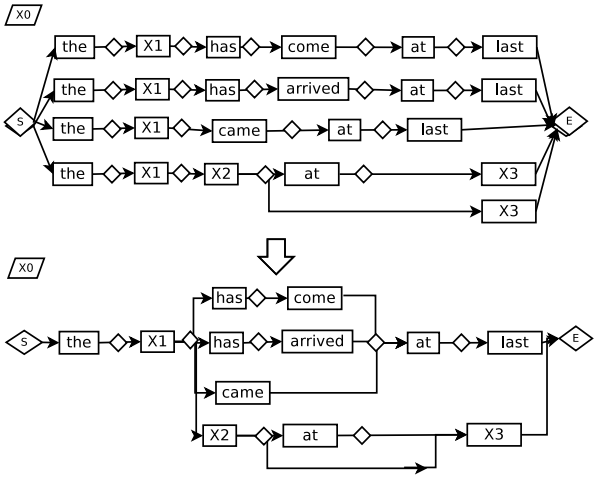


Figure 3: The lattice RHS(X0) optimized with the algorithm described in section 4

4 Representation optimisation

4.1 Goal

Given a description as a set of rule and scores $\mathcal{R}_T^1$ of $\mathcal{T}$, it is often possible to find another description $\mathcal{R}_T^2$ of $\mathcal{T}$ having the same formalism but a different set of rules. Although the $\mathcal{T}$ that is described remains the same, the same search algorithm applied to $\mathcal{R}_T^1$ or $\mathcal{R}_T^2$ might make approximations in a different way, be faster or use less memory.

It is an interesting question to try to transform an initial representation $\mathcal{R}_T^1$ into a representation $\mathcal{R}_T^2$ that will make the search step faster. This is especially interesting if one is going to search the same $\mathcal{T}$ several times, as is often done when one is fine-tuning the parameters of a model, as this representation optimisation needs only be done once.

The optimisation we propose is a natural fit to our framework of lattice rules. As lattice are a special case of Finite-State Automata (FSA), it is easy to adapt existing algorithms for FSA minimization. We describe a procedure in algorithm 1, which is essentially a simplification and adaptation to our case of the more general algorithm of (Hopcroft, 1971) for FSA. The central parts of the algorithm are the two sub-procedures *backward vertex merging* and *forward vertex merging*. An example of the result of an optimisation is given on figure 3.

Data: Representation $\mathcal{R}_T$
Result: Optimized Representation
1 **for** *non-terminal* $X \in \mathcal{R}_T$ **do**
2 | Apply backward vertex merging to $RHS(X)$;
3 | Apply forward vertex merging to $RHS(X)$;
4 **end**
Algorithm 1: Representation optimisation

4.2 Forward and backward merging

We describe the *forward vertex merging* in algorithm 2. This merging will merge vertices and suppress redundant edges, proceeding from left to right. The end result is a lattice with a reduced number of vertices and edges, but encoding the same paths as the initial one.

The basic idea here is to check the vertices from left to right and merge the ones that have identical incoming edges. After having been processed by the algorithm, a vertex is put in the set $\mathcal{P}$ (line 9). At each iteration, the candidate set $\mathcal{C}$ contains the set of vertices that can potentially be merged together. It is updated at each iteration to contain the set of not-yet-processed vertices for which all incoming edges come from processed vertices (done by marking edges at line 6 and then updating $\mathcal{C}$ at line 10). At each iteration, the merging process consists in:

1. Eliminating *duplicate* edges from the processed vertices to the candidate vertices (line 5). These duplicate edges could have been introduced by the merging of previously processed vertices.

2. Merging vertices whose set of incoming edges is *identical*. Here, merging two vertices v_1 and v_2 means that we create a third vertex v_3 such that incoming(v_3) = incoming(v_1) = incoming(v_2), and outgoing(v_3) = outgoing($v_3$1) $\bigcup$ outgoing(v_2), then remove v_1 and v_2.

The backward vertex merging is defined similarly to the forward merging, but with going right to left and inverting the role of the incoming and outgoing edges.

Data: Lattice RHS $\mathcal{L}$
Result: Optimized Lattice RHS
1 $\mathcal{P} \leftarrow \emptyset$ //processed vertices;
2 $\mathcal{C} \leftarrow \{v_S\}$ //candidate set ;
3 **while** $|\mathcal{C}| > 0$ **do**
4 | **for** $v \in \mathcal{C}$ **do**
5 | | Eliminate duplicate edges in incoming(v);
6 | | Mark edges in outgoing(v);
7 | **end**
8 | Merge all vertices $v_1, v_2 \in \mathcal{C}$ such that incoming(v_1) = incoming(v_2);
9 | $\mathcal{P} \leftarrow \mathcal{P} \bigcup \mathcal{C}$;
10 | $\mathcal{C} \leftarrow \{v \in \mathcal{L} \setminus \mathcal{P}$ s.t. all edges in incoming(v) are marked$\}$;
11 **end**
Algorithm 2: Forward Vertex Merging

4.3 Optimizing the whole representation

Algorithm 1 describe the global optimisation procedure. For each lattice RHS, we just perform first a backward merge and then a forward merge.

We have set the constraint in section 2.3 that each non-terminal should have only one lattice RHS. Note here that if there are several RHS for a given non-terminal, we can first merge them by merging their start vertex and end vertex, then apply this optimisation algorithm to obtain a representation with one optimised RHS per non-terminal.

This optimisation could be seen as doing some form of *hypothesis recombination*, but offline.

In term of *rule optimisations*, we only consider here transformations that do not modify the number of non-terminals. But it is worthwhile to note that there are some sequence appearing in the middle of some rules that cannot be merged through a lattice representation, but could be factored as sub-rules appearing in different non-terminals. Indeed, a lattice rule could actually be encoded as a set of "flat" rules by introducing a sufficient number of non-terminals, but this could possibly be less efficient from the search algorithm point of view. We plan to investigate the effects of this type of rule optimisations in conjunction with the described lattice-type opti-

misations in the future.

4.4 Handling of Edge Features

In the context of parameter tuning, we usually want the decoder to output not only the translations, but also a list of features characterizing the way the translation was constructed. Such features are, for example, the number of rules used, the language model of the translation, etc. In out context, some features will be dependent on the specific edges used in a rule. For example, the epsilon edge used to optionally skip non-aligned words (see section 3.2.1) is labeled with a feature "nb-words-skipped" set to 1, so that we can obtain the number of words skipped in a given translation and tune a score penalty for skipping such words. Similar features also exist for picking a word variation (section 3.2.3).

In the description of the merging process of section 4.2, one should thus be aware that two edges are to be considered identical only if both their associated word and their set of feature values are identical. This can sometimes prevent useful merging of states to take place. A solution to this could be to follow (de Gispert et al., 2010) and to discard all these features information during the decoding. The features values are then re-estimated afterward by aligning the translation and the input with a constrained version of the decoder.

We prefer to actually keep track of the features values, even if it can reduce the efficiency of vertex merging. In that setting, we can also adapt the so-called Weight Pushing algorithm (Mohri, 2004) to a multivalues case in order to improve the "mergeability" of vertices. The results of section 6.1 shows that it is still possible to strongly reduce the size of the lattices even when keeping track of the features values.

5 Decoding algorithm

In order to make an optimal use of these lattice-rule representations, we developed a decoding algorithm for translation candidate sets represented as a set of lattice-rules. For the most part, this algorithm re-use many of the techniques previously developed for decoding translation search spaces, but adapt them to our setting.

5.1 Overview

The outline of the decoding algorithm is described by algorithm 3. For simplicity, the description only compute the optimal model score over the translations in the candidate set. It is however trivial to adapt the description to keep track of which sentence correspond to this optimal score and output it instead of the score. Likewise, using the technique described in (Huang and Chiang, 2005), one can easily output k-best lists of translations. For simplicity again, we consider that a n-gram language model score is the only stateful non-local feature used for computing the model score, although in a tree-to-tree setting, other features (local in a tree representation but not in a string representation) could be used. The model score of a translation t has therefore the shape:

$$score(t) = \lambda \cdot lm(t) + \sum_{e} score(e)$$

where λ is the weight of the language model, $lm(t)$ is the language model log-probability of t and the sum is over all edges e crossed to obtain t.

5.2 Scored language model states

Conceptually, in a lattice $\mathcal{L}$, at each vertex v, we can consider the partial translations obtained by starting at v_S and concatenating the words labeling each edge not labeled by a non-terminal until v. If an edge is labeled by a non-terminal X, we first traverse the corresponding lattice $RHS(X)$ following the same process. Such a partial translation can be reduced compactly to a *scored language model state* (l, r, s), where l represent the first n words[1] of the partial translation, r its last n words and s its partial score. It is clear that if two partial translations have the same l and r parts but different score, we can discard the one with the lowest score, as it cannot be a part of the optimal translation.

Further, using the state reduction techniques described in (Li and Khudanpur, 2008) and (Heafield et al., 2011), we can often reduce the size of l and r to less than n, allowing further opportunities for discarding sub-optimal

[1] n being the order of the language mode

partial translations. For better behavior during the cube-pruning step of the algorithm (see later), the partial score s of a partial translation includes rest-costs estimates (Heafield et al., 2012).

We define the concatenation operation on scored language model states to be: $(l_1, r_1, s_1) \oplus (l_2, r_2, s_2) = (l_3, r_3, s_3)$, where $s_3 = s_1 + s_2 + \lambda lm(r_1, l_2)$, with $lm(r_1, l_2)$ being the language model probability of l_2 given r_1 with rest-costs adjustments. r_3 and l_3 are the resulting minimized states. Similarly, if an edge e is labeled by a word, we define the concatenation of a scored state with an edge to be $(l_1, r_1, s_1) \oplus e = (l_2, r_2, s_2)$ where $s_2 = s_1 + score(e) + \lambda lm(word(e)|r_1)$.

Conveniently for us, the KenLM[2] open-source library (Heafield, 2011) provides functionalities for easily computing such concatenation operations.

5.3 Algorithm

Having defined these operations, we can now more easily describe algorithm 3. Each vertex v has a list best[v] of the scored states of the best partial translations found to be ending at v. On line 1, we initialize best[v_S] with $(.,.,0)$, where "." represent an empty language model state. We then traverse the vertices of the lattice in topological order.

For each edge $e : v_1 \rightarrow v_2$, we compute new scored states for best[v_2] as follow:

- if e is labeled by a word or an epsilon, we create a state $st_2 = st_1 \oplus e$ for each st_1 in best[v_1] (line 10).
- if e is labeled by a non-terminal X, we recursively call the decoding algorithm on the lattice $RHS(X)$. The value returned by the line 15 will be a set of states corresponding to optimal partial translations traversing $RHS(X)$. We can concatenate these states with the ones in best[v_1] to obtain states corresponding to partial translations ending at v_2 (line 6).

Results of the calls decode(X) are memoized, as the same non-terminal is likely to appear in several edges of a RHS and in several RHS.

Lines 5 and 6 are the "cube-pruning-like" part of the algorithm. The function $prune_K$ returns the K best combinations of states in best[v] and decode($RHS(X)$), where best means "whose sum of partial score is highest". It can be implemented efficiently through the algorithms proposed in (Huang and Chiang, 2005) or (Chiang, 2007).

The $L \leftarrow_{max} st$ operation on lines 6 and 10 has the following meaning: L is a list of scored language model state and st is a scored language model state. $L \leftarrow_{max} st$ means that, if L already contains a state st_2 with same left and right state as st, L is updated to contain only the scored state with the maximum score. If L do not contain a state similar to st, st in simply inserted into L. This is the "hypothesis recombination" part of the algorithm. The function $trunc_{K'}$ truncate the list best[v] to its K' highest-scored elements.

The final result is obtained by calling decode(X_0), where X_0 is the "top-level" non-terminal. The result of decode(X_0) will contain only one scored state of the form (BOS, EOS, s), with s being the optimal score.

The search procedure of algorithm 3 could be described as "breadth-first", since we systematically visit each edge of the lattice. An alternative would be to use a "best-first" search with an A*-like procedure. We have tried this, but either because of optimisation issues or heuristics of insufficient qualities, we did not obtain better results than with the algorithm we describe here.

6 Evaluation

We now describe a set of experiments aimed at evaluating our approach.

We use the Japanese-English data from the NTCIR-10 Patent MT task[3] (Goto et al., 2013). The training data contains 3 millions parallel sentences for Japanese-English.

6.1 Effect of Lattice Representation and Optimisation

We first evaluate the impact of the lattice representation on the performances of our decoding algorithm. This will allow us to measure

[2] http://kheafield.com/code/kenlm/

[3] http://ntcir.nii.ac.jp/PatentMT-2/

Data: Lattice RHS $\mathcal{L}$
Result: Sorted list of best states
1 best$[v_E] = \{(.,.,0.0)\}$;
2 **for** *vertex* $v \in \mathcal{L}$ *in topological order* **do**
3 **for** *edge* $e : v \rightarrow v_2 \in$ *outgoing(v)* **do**
4 **if** *label(e)* $= X$ **then**
5 **for** $st_1, st_2 \in prune_K$*(best[v], decode(RHS(X))* **do**
6 best$[v_2] \leftarrow_{max} st_1 \oplus st_2$;
7 **end**
8 **else**
9 **for** $st \in trunc_{K'}$*(best[v])* **do**
10 best$[v_2] \leftarrow_{max} st \oplus e$;
11 **end**
12 **end**
13 **end**
14 **end**
15 **return** best$[v_E]$;

Algorithm 3: Lattice-rule decoding. See body for detailed explanations.

the benefits of our compact lattice representation of rules, as well as the benefits of the representation optimisation algorithm of section 4.

We use our Syntactic-dependency system to generate a lattice-rule representation of the possible translations of the 1800 sentences of the development set of the NTCIR-10 Patent MT task. We then produce two additional representations:

1. An optimized lattice-rule representation using the method described in section 4.

2. An expanded representation, that unfold the original lattice-rule representation into "flat rules" enumerating each path in the original lattice-rule representation (like the list $X0'$ enumerate the lattice $X0$ in figure 2).

Table 1 shows 3 columns. One for each of these 3 representations. We can see that, as expected, the performances in term of average search time or peak memory used are directly related to the number of vertices and edges in the representation. We can also see that our representation optimisation step is quite efficient, since it is able to divide by two the number of vertices in the representation, on average. This leads to a 2-fold speed improvement in the decoding step, as well as a large reduction of memory usage.

6.2 Decoding performances

In order to further evaluate the merit of our approach, we now compare the results obtained by using our decoder with lattice-rules with using a state-of-the-art decoder on the set of flat expanded rules equivalent to these lattice rules.

We use the decoder described in (Heafield et al., 2013), which is available under an open-source license[4] (henceforth called K-decoder). In this experience, we expanded the lattice rules generated by our MT system for 1800 sentences into files having the required format for the K-decoder. This basically mean we computed an equivalent of the expanded representation of section 6.1. This process generated files ranging in size from 20MB to 17GB depending on the sentence. We then ran the K-decoder on these files and compared the results with our own. We used a beam-width of 10000 for the K-decoder. Experiments were run in single thread mode. Partly to obtain more consistent results, and partly because the K-decoder was risking using too much memory for our system.

The results on table 3 show that, as the K-decoder do not have access to a more compact representation of the rules, it end up needing a much larger amount of memory for decoding the same sentences.

In term of model score obtained, the performances are quite similar, with the lattice-rule decoder providing slightly better model score.

It is interesting to note that, on "fair-ground" comparison, that is if our decoder do not have the benefit of a more compact lattice-rule representation, it actually perform quite worse as we can see by comparing with the third column of table 1 (at least in term of decoding time and memory usage, while it would still have a very slight edge in term of model score with the selected settings). On the other hand, the K-decoder is a rather strong baseline, shown to perform several times faster than a previous state-of-the-art implementation in (Heafield et al., 2013). It is well opti-

[4] http://kheafield.com/code/search/

Representation:	**Original**	**Optimized**	Expanded
Peak memory used	39 GB	**16GB**	85GB
Average search time	6.13s	**3.31s**	9.95s
#vertices (avg/max)	65K (1300K)	**32K (446K)**	263K (5421K)
#edges (avg/max)	92K (1512K)	**83K (541K)**	263K (5421K)

Table 1: Impact of the lattice representation on performances.

System	**JA–EN**
Lattice	**29.43**
No-variations	28.91
Moses (for scale)	28.86

Table 2: Impact on BLEU of using flexible lattice rules.

mized and makes use of advanced techniques with the language model (as the one described in (Heafield et al., 2013)) for which we do not have implemented an equivalent yet. Therefore, we are hopeful we can further improve our decoder in the future.

Also, note that, for practical reason, while we only measured the decoding time for our decoder [5], the K-decoder time include the time taken for loading the rule files.

6.3 Translation quality

Finally, we evaluate the advantages of extracting lattice rules such as proposed in section 3. That is, we consider rules for which null-aligned words are bypassable by epsilon-edges, for which Non-terminal are allowed to take several alternative positions around the word that is thought to be their governor, and for which we consider alternative morphologies of a few words ("is/are", "a/an"). We compare this approach with heuristically selecting only one possibility for each variation present in the lattice rule extracted from a single example.

Results shown on figure 2 show that we do obtain a significant improvement in translation quality. Note that the Moses score (Koehn et al., 2007), taken from the official results of NTCIR-10 is only here "for scale", as our MT system uses a quite different pipeline.

[5] in particular, we factored out the representation optimisation time, which is reasonable if we are in the setting of a parameter tuning step in which the same sentences are translated repeatedly

7 Related work

Searching for the most optimal translation in an implicitly defined set has been the focus of a lot of research in Machine Translation and it would be difficult to cover all of it. Among the most influential approaches, (Koehn et al., 2003) was using a form of stack based decoding for Phrase-Based Machine Translation. (Chiang, 2007) introduced the cube-pruning approach, which has been further improved in the previously mentioned (Heafield et al., 2013). (Rush and Collins, 2011) recently proposed an algorithm promising to find the optimal solution, but that is rather slow in practice.

Weighted Finite State Machines have seen a variety of use in NLP (Mohri, 1997). More specifically, some other previous work on Machine Translation have used lattices (or more generally Weighted Finite State Machines). In the context of Corpus-Based Machine Translation, (Knight and Al-Onaizan, 1998) was already proposing to use Weighted Transducers to decode the "IBM" models of translation (Brown et al., 1993). (Casacuberta and Vidal, 2004) and (Kumar et al., 2006) also propose to directly model the translation process with Finite State Transducers. (Graehl and Knight, 2004) propose to use Tree Transducers for modeling Syntactic Machine Translation. These approaches are however based on different paradigm, typically trying to directly learn a transducer rather than extracting SCFG-like rules.

Closer to our context, (de Gispert et al., 2010) propose to use Finite-State Transducers in the context of Hierarchical Phrase Based Translation. Their method is to iteratively construct and minimize the full "top-level lattice" representing the whole set of translations bottom-up. It is an approach more focused on the Finite State Machine aspect than our,

System	K-decoder	Lattice-rule decoder
Peak memory used	52G	**16G**
Average search time	3.47s	**3.31s**
Average model score	-107.55	**-107.39**
Nb wins	401	**579**

Table 3: Evaluation of the performances of our lattice-rule decoder compared with a state-of-the-art decoder using an expanded flat representation of the lattice rules. "Nb wins" is the number of times one of the decoder found a strictly better model score than the other one, out of 1800 search.

which is more of an hybrid approach that stays closer to the paradigm of cube-pruning. The merit of their approach is that they can apply minimization globally, allowing for more possibilities for vertex merging. On the other hand, for large grammars, the "top-level lattice" will be huge, creating the need to prune vertices during the construction. Furthermore, the composition of the "top-level lattice" with a language model will imply redundant computations (as lower-level lattices will potentially be expanded several times in the top-level lattice). As we do not construct the global lattice explicitly, we do not need to prune vertices (we only prune language model states). And each edge of each lattice rule is crossed only once during our decoding.

Very recently, (Heafield et al., 2014) also considered using the redundancy of translation hypotheses to optimize phrase-based stack decoding. To do so, they group the partial hypotheses in a trie structure.

We are not aware of other work proposing "lattice rules" as a native format for expressing translational equivalences. Work like (de Gispert et al., 2010) rely on SCFG rules created along the (Chiang, 2007) approach, while work like (Casacuberta and Vidal, 2004) adopt a pure Finite State Transducer paradigm (thus without explicit SCFG-like rules).

8 Conclusion

This work proposes to use a lattice-rule representation of the translation search space with two main goals:

- Easily represent the translation ambiguities that arise either due to lack of context or imperfect knowledge.
- Have a method for optimizing the representation of a search space to make this search more efficient.

We demonstrate that many types of ambiguities arising when extracting translation rules can easily be expressed in this framework, and that making these ambiguities explicit and solvable at compile time through lattice-rules leads to improvement in translation quality.

We also demonstrate that making a direct use of the lattice-rules representation allows a decoder to perform better than if working on the expanded set of corresponding "flat rules". And we propose an algorithm for computing more efficient representations of a translation candidate set.

We believe that the the link between the representation of a candidate set and the decoding efficiency is an interesting issue and we intend to explore further the possibilities of optimizing representations both in the contexts we considered in this paper and in others such as Phrase-Based Machine Translation.

The code of the decoder we implemented for this paper is to be released under a GPL license[6].

Acknowledgements

This work is supported by the Japanese Science and Technology Agency. We want to thank the anonymous reviewers for many very useful comments.

References

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation:

[6] http://nlp.ist.i.kyoto-u.ac.jp/kyotoebmt/

Parameter estimation. *Computational linguistics*, 19(2):263–311.

Francisco Casacuberta and Enrique Vidal. 2004. Machine translation with inferred stochastic finite-state transducers. *Computational Linguistics*, 30(2):205–225.

David Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

Adrià de Gispert, Gonzalo Iglesias, Graeme Blackwood, Eduardo R Banga, and William Byrne. 2010. Hierarchical phrase-based translation with weighted finite-state transducers and shallow-n grammars. *Computational linguistics*, 36(3):505–533.

Isao Goto, Ka Po Chow, Bin Lu, Eiichiro Sumita, and Benjamin K Tsou. 2013. Overview of the patent machine translation task at the ntcir-10 workshop. In *Proceedings of the 10th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access, NTCIR-10*.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proceedings of HLT-NAACL*.

Kenneth Heafield, Hieu Hoang, Philipp Koehn, Tetsuo Kiso, and Marcello Federico. 2011. Left language model state for syntactic machine translation. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 183–190, San Francisco, California, USA, December.

Kenneth Heafield, Philipp Koehn, and Alon Lavie. 2012. Language model rest costs and space-efficient storage. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1169–1178, Jeju Island, Korea, July.

Kenneth Heafield, Philipp Koehn, and Alon Lavie. 2013. Grouping language model boundary words to speed k-best extraction from hypergraphs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 958–968, Atlanta, Georgia, USA, June.

Kenneth Heafield, Michael Kayser, and Christopher D. Manning. 2014. Faster Phrase-Based decoding by refining feature state. In *Proceedings of the Association for Computational Linguistics*, Baltimore, MD, USA, June.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.

John Hopcroft. 1971. An n log n algorithm for minimizing states in a finite automaton. *Theory of Machines and Computations*, pages 189–196.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64. Association for Computational Linguistics.

Kevin Knight and Yaser Al-Onaizan. 1998. Translation with finite-state devices. In *Machine translation and the information soup*, pages 421–437. Springer.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

Shankar Kumar, Yonggang Deng, and William Byrne. 2006. A weighted finite state transducer translation template model for statistical machine translation. *Natural Language Engineering*, 12(01):35–75.

Zhifei Li and Sanjeev Khudanpur. 2008. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proceedings of the Second Workshop on Syntax and Structure in Statistical Translation*, pages 10–18. Association for Computational Linguistics.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311.

Mehryar Mohri. 2004. Weighted finite-state transducer algorithms. an overview. In *Formal Languages and Applications*, pages 551–563. Springer.

John Richardson, Fabien Cromières, Toshiaki Nakazawa, and Sadao Kurohashi. 2014. Kyotoebmt: An example-based dependency-to-dependency translation framework. In *Proceedings of ACL (System Demonstration)*, Baltimore, MD, USA, June.

Alexander M Rush and Michael Collins. 2011. Exact decoding of syntactic translation models through lagrangian relaxation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 72–82. Association for Computational Linguistics.

Learning to Translate: A Query-Specific Combination Approach for Cross-Lingual Information Retrieval

Ferhan Ture
Raytheon BBN Technologies
10 Moulton St
Cambridge, MA, 02138 USA
fture@bbn.com

Elizabeth Boschee
Raytheon BBN Technologies
10 Moulton St
Cambridge, MA, 02138 USA
eboschee@bbn.com

Abstract

When documents and queries are presented in different languages, the common approach is to translate the query into the document language. While there are a variety of query translation approaches, recent research suggests that combining multiple methods into a single "structured query" is the most effective. In this paper, we introduce a novel approach for producing a unique combination recipe for each query, as it has also been shown that the optimal combination weights differ substantially across queries and other task specifics. Our query-specific combination method generates statistically significant improvements over other combination strategies presented in the literature, such as uniform and task-specific weighting. An in-depth empirical analysis presents insights about the effect of data size, domain differences, labeling and tuning on the end performance of our approach.

1 Introduction

Cross-lingual information retrieval (CLIR) is a special case of information retrieval (IR) in which documents and queries are presented in different languages. In order to overcome the language barrier, the most commonly adopted method is to translate queries into the document language. Many methods have been introduced for translating queries for CLIR, ranging from word-by-word dictionary lookups (Xu and Weischedel, 2005; Darwish and Oard, 2003) to sophisticated use of machine translation (MT) systems (Magdy and Jones, 2011; Ma et al., 2012). Previous research has shown that combining evidence from different translation approaches is superior to any single query translation method (Braschler, 2004; Herbert et al., 2011). While there are numerous combination-of-evidence techniques for both mono-lingual and cross-lingual IR, recent work suggests that there is no one-size-fits-all solution. In fact, the optimal *combination weights* (i.e., weights assigned to each piece of evidence in a linear combination) differ greatly across queries, tasks, languages, and other variants (Ture et al., 2012; Berger and Savoy, 2007).

In this paper, we introduce a novel method for *learning optimal combination weights* when building a linear combination of existing query translation approaches. From standard query-document relevance judgments we train a set of classifiers, which produce a unique combination recipe for each query, based on a large set of features extracted from the query and collection. Experimental results show that the effectiveness of our method is significantly higher than state-of-the-art query translation methods and other combination strategies.

2 Related Work

The earliest approaches to query translation for CLIR used machine-readable bilingual dictionaries (Hull and Grefenstette, 1996; Ballesteros and Croft, 1996), achieving around up to 60% of monolingual IR effectiveness. Xu and Weischedel (2005) showed that effectiveness can be increased to around 80% by weighting each translation proportional to its rank in the dictionary. The practice of weighting translation candidates was later formulated as a "structured query", in which each query term is represented by a probability distribution over its translations in the document language (Pirkola, 1998; Kwok, 1999; Darwish and Oard, 2003). Our approach is based on the structured query formulation.

Some of the earliest studies in IR discovered that with different underlying models, the retrieved document set would vary substantially, al-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 589–599,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

though the effectiveness was similar (McGill et al., 1979). Later studies showed that combining different representations of the query and/or document often produced superior output (Rajashekar and Croft, 1995; Turtle and Croft, 1990; Fox, 1983). This intuitive idea was supported theoretically by Pearl (1988), concluding that multiple pieces of evidence estimates relevance more accurately, but that the benefit strongly depends on the *quality* and *independence* of each piece. Experiments by Belkin et al. (1995) indicated the need to properly weight each representation with respect to its effectiveness. These so-called "combination-of-evidence" techniques became more powerful with the introduction of `Indri`, a probabilistic retrieval framework specifically designed for combining multiple query and document representations (Metzler and Croft, 2005). Croft (2000) provides a detailed summary of earlier query combination approaches in IR, while Peters et al. (2012) cites more recent related work.

The benefits of combination-of-evidence transfer to the cross-lingual case especially well, since the inherent ambiguity of translation readily provides a diverse set of representations. Most CLIR approaches implement a post-retrieval *merging* of ranked lists, each generated from different query (Hiemstra et al., 2001; Savoy, 2001; Gey et al., 2001; Chen and Gey, 2004) or document (Lopez and Romary, 2009) representations, also called "data fusion". In contrast, we focus on a pre-retrieval combination at the modeling stage, so that a single complex query is used in retrieval, instead of multiple simpler ones. Two advantages of the former are easier implementation (since the approach requires no changes to the modeling side) and the possibly greater diversity that can be achieved by having separate retrieval runs. However, each ranked list needs to be limited in size, which might cause some potentially useful documents not to be considered in the combination at all. Since the focus of this paper is on the modeling end of retrieval, pre-retrieval combination was a more suitable choice, though we think that the two approaches have complementary benefits.

The idea of combining query translations before retrieval has been explored previously. Braschler (2004) combines three translation approaches: output of an MT system, a novel translation approach based on a similarity thesaurus built automatically from a comparable corpus, and a dictionary-based translation. The main reason that this combination does not provide much benefit is due to the lower coverage of the thesaurus-based and dictionary-based translation methods. A similar approach by Herbert et al. (2011) uses Wikipedia to provide translations of certain phrases and entities, and combining that with the `Google Translate` MT system yields statistically significant improvements in English-to-German retrieval. More recently, Ture et al. (2012) presented a more sophisticated translation approach using the internal representation of an MT system, and reported statistically significant improvements when a pre-retrieval combination was performed.

All of the previously cited approaches either use uniform weights for combination, or select weights based on collection-level information. However, as stated previously, numerous studies suggest that certain methods work better on certain queries, collections, languages. In fact, when weights are optimized separately on each collection, they differ substantially across different collections (Ture et al., 2012). For monolingual retrieval, there has been a series of learning-to-rank (LTR) papers that determine weights for query concepts (Bendersky et al., 2011), such that retrieval effectiveness is maximized. A recent study extends this idea to the cross-lingual case, by learning how to weight each *translated word* for English-Persian CLIR (Azarbonyad et al., 2013). In contrast, we extract translated word weights from diverse and sophisticated translation methods, then learn how to weight each *translated structured query*, We call this "learning-to-translate" (LTT), which can be formulated as a simpler learning problem. In CLIR, both LTR and LTT are under-explored problems, with a common goal of applying machine learning techniques to improve query translation, yet with complementary benefits.

To our knowledge, there has been one prior LTT approach: a classifier was trained to predict effectiveness of each query translation, using features based on statistics of the query terms (Berger and Savoy, 2007). Instead of weighting, the translations with highest classifier scores were concatenated, yielding statistically significant improvements over using the single-best translation method. However, the translation methods explored in this paper are all based on one-best MT

systems, making it difficult to draw strong conclusions.

3 Query Translation

The primary contribution of this paper is to show how a diverse set of query translation (QT) methods can be combined effectively into a single weighted structured query, with improved retrieval effectiveness. While our approach can applied to any set of translation methods, we focus on three methods that have complementary strengths and that have shown promise in CLIR: word-based probabilistic translation, one-best MT, and n-best probabilistic MT. We briefly present our implementation of each method; more details can be found in earlier work (Darwish and Oard, 2003; Ture et al., 2012).

Each QT method generates a representation of the query in the document language. In the case of word-based and n-best MT approaches, the representation is a structured query itself, where each query word is represented by a probability distribution over translation alternatives. For one-best MT, the query is represented by a bag of translated words.

3.1 One-Best MT

A query translation approach that has become more popular recently is to simply run the query through an MT system, and use the best output as the query:

$$t_1 t_2 \dots t_l = \mathbf{MT}(s_1 s_2 \dots s_k) \quad (1)$$

where $s = s_1 s_2 \dots s_k$ is the query and $t = t_1 t_2 \dots t_l$ is the translated query.

Since modern statistical MT systems generate high-quality translations for many language pairs, this one-best strategy works reasonably well for retrieval and provides a competitive baseline. A practical advantage of this approach is the ease of implementation – one can simply use any MT interface (e.g., `Google Translate`) as a black box in their CLIR system.

3.2 Probabilistic n-best MT

The top translation might sometimes be incorrect, or might lack some of the alternative representations that are very useful in retrieval. Therefore, considering the n highest scored translations (also referred to as the n-best list in MT literature) has become increasingly popular in CLIR approaches.

In order to benefit from the diversity amongst the n-best translations, one can simply concatenate them together, forming a large list of query terms. However, statistical MT systems also assign probabilities to each translation, which can be incorporated into the query representation for better effectiveness, as suggested by Ture et al. (2012).

In this approach, each of the top n translation candidates from the MT system are processed one by one. For each translation candidate, the MT system provides a translation probability, and alignments between words in the query and its translation. As we process each of the n translations, for each query word s_i, we accumulate probabilities on each translated word t_{ij} aligned to s_i. Finally, we normalize the translation probabilities to get $\mathbf{Pr}_{\text{nbest}}(t_{ij}|s_i)$.

3.3 Word-based

One of the most widely used approaches in CLIR is based on translating each query word s_i independently, with probabilities assigned to each translation candidate t_{ij}. Translations are derived automatically from a bilingual corpus using statistical word alignment techniques, which are used as part of the training of statistical MT systems (Brown et al., 1993). These probabilities can be exploited for retrieval based on the technique of Darwish and Oard (2003) for "projecting" text into the document language. After cleaning up the automatically learned translation probabilities (details omitted for space considerations), we end up with the translation probabilities $\mathbf{Pr}_{\text{word}}(t_{ij}|s_i)$.

4 Combination of Evidence

Once we have multiple ways to represent the query q in the document language ($\text{QT}_i(q), i = 1 \dots m$), it is possible to combine these "pieces of evidence" into a single representation as follows:

$$\text{QT}(q) = \sum_{i=1}^{m} w_i(q)\text{QT}_i(q)$$

and each combination-of-evidence approach differs by how the combination weights w_i are computed:

Uniform In this baseline method, we ignore any information we have about the collection or query and assign equal weights to each method (i.e., $w_i(q) = 1/m$). In our case, this means a weight

of 33.3% to each of the one-best, probabilistic n-best, and word-based QT methods.

Task-specific We can optimize the combination weights by overall effectiveness on a specific retrieval task. Given a query set and collection, we perform a grid search on combination weights (with a step interval of 0.1) and select the weights that maximize retrieval effectiveness. The training is performed in a leave-one-out manner: weights for test query q are optimized on all queries except for q.

Query-specific We propose a novel method to compute combination weights specifically for each query, resulting in a more customized optimization that can take into account how effectiveness of each translation method varies across queries.

In the remainder of this section, we describe the details of our novel query-specific combination method.

4.1 Overview of Query-Specific Combination

We present a novel approach for determining query-specific combination weights by training a classifier for each QT method. Prior to training the classifier, we first run retrieval using each QT method, and evaluate the effectiveness of the retrieved documents. The effectiveness of the i^{th} method on query q (i.e., $f_i(q)$) is then converted into a binary label (further described in Section 4.2). Treating each query as a separate instance, a classifier is trained for each method, generating classifiers $C_1, \ldots, C_m$. During retrieval (i.e., at test time), for each query q, each trained classifier C_i is applied to the query, resulting in a predicted label $l_i(q)$ and the classifier's confidence in a positive label, $C_i(q)$.[1] These values are then used to determine combination weights $w_1(q), \ldots, w_m(q)$ that are custom-fit for the query.

4.2 Labeling

First of all, we discard queries in which the difference between the best and worst performing methods is small (specifically, the worst performing method scores at least $k_1\%$ of the best performing one). For such queries, generating fair training labels is more difficult and therefore more likely to introduce noise into the process.[2] Moreover, these are exactly the queries where choosing optimal combination weights is less important (since all methods perform relatively similarly), so it is reasonable to exclude them from training. In fact, a high number of such queries would indicate lower potential for combination-of-evidence approaches.

For each QT method i, we create training instances per query, per retrieval task. Since our goal is to select the best among existing methods, the training label should reflect the effectiveness of method i relative to other methods. A strategy that we call *best-by-measure* assigns a label of 1 if the effectiveness of the i^{th} method (i.e., $f_i(q)$) is at least $k_2\%$ of the maximum effectiveness for that query, and 0 otherwise. While this directly correlates with retrieval effectiveness, labels might be distributed in an unbalanced manner, which might affect the training process negatively. A balanced labeling requires sorting all training instances by how much better the i^{th} method is than other methods ($\max_{i' \neq i}(f_{i'}(q)/f_i(q))$), and then assigning a label of 1 to the lower half and 0 to the higher half. This strategy is called *best-by-rank*.

4.3 Features

We introduce a diverse set of features, in order to train a robust classifier for predicting when each QT method performs better and worse than others. We split the feature set into four meaningful categories, so that we can measure the impact of each subset separately:

Surface features These features do not require a deep analysis of the query: (a) Number of words in query and the translated query, (b) Type of query that we automatically classify based on predefined templates (e.g., fact question, cause-effect, etc.), and (c) Number of stop words in the query and the translated query.

Parse-based features These features are extracted from a deeper syntactic analysis of the query text: (a) Number of related names found in a named entity database, and (b) Existence of syntactic constituents in query and its translation (e.g., "is there a VVB in the query parse tree").

[1] The confidence in a negative label is $1 - C_i(q)$.

[2] We also experimented with including these queries with a third label (e.g., "same") and train a ternary classifier. Having more labels requires more training data, which is not easy to obtain for this task. Also, obtaining a balanced label distribution becomes even more difficult with three labels.

Translation-based features These features consist of statistics computed from the query and its translation: (a) Number of query words that were unaligned in at least half of the n-best query translations, (b) Number of query words that were aligned to multiple target words in at least half of the n-best query translations, (c) Number of query words that were self-aligned (i.e., target word is exactly same string) in at least half of the n-best query translations, (d) Average / Standard deviation / Maximum / Minimum of entropy of $\mathbf{Pr}_{\text{nbest}}$ of each query word, and (e) Average / Standard deviation / Maximum / Minimum of entropy of $\mathbf{Pr}_{\text{word}}$ of each query word.

Index-based features These features are based on frequency statistics from a representative collection:[3] (a) Average / Standard deviation / Maximum / Minimum of document frequency (df) of query words and their translations, (b) Average / Standard deviation / Maximum / Minimum of term frequency averaged across query words and their translations, and (c) Sum / Maximum / Minimum of total probability assigned to words that do not appear in the collection (df $= 0$).

Additionally, the target language is a default feature in all of our experiments. For each classification task, we train a separate classifier on each subset of these four feature categories, so that there are 16 different sets (including the empty set). After we select which categories to pull features from, we optionally perform feature selection to reduce the number of features by a predefined percentage.

In our experimentation, we observed that collection-based features were most useful for classifying the one-best method, whereas parse-based features were most discriminative for probabilistic 10-best. For the word-based QT method, the translation-based features were most effective in our experiments. We further analyze the effect of various features in Section 5.

4.4 Training and Tuning Classifiers

The `scikit-learn` package was used for the training pipeline (Pedregosa et al., 2012). Using an established toolkit allowed us to experiment with many options for classification, such as the learner type (support vector machine, maximum entropy, decision tree), feature set (16 subsets of the four categories described earlier) and two feature selection methods (recursive elimination or selection based on univariate statistical tests). In the end, we get 96 different parameter combinations while training a classifier for a particular QT method, resulting in the need for tuning — picking the parameters that produce highest accuracy on a representative *tuning set*.

Given that we have a set of queries for testing purposes, there are few strategies for selecting a training and tuning set. One approach is to apply a leave-one-out strategy, so that a classifier is trained and tuned on all but one of the test queries, and then applied on the remaining query to predict its label. We call this the *fully-open* setting.

In a more realistic scenario, there will not be relevance judgments for the test queries, yet there might be a small amount of labeled data similar to the test task (e.g., different queries on same collection) that can be utilized for tuning purposes, and a larger set of training queries from different collections. We call this the *half-blind* setting.

If testing in a new domain, queries of similar type are not available for training and tuning purposes. This is a more challenging scenario than the previous two, yet it is important for real-world applications. In order to demonstrate the effectiveness of the training pipeline in this case, we hold out test queries entirely, then train and tune on queries from a completely different task (i.e., different queries *and* collection). We call this the *fully-blind* setting.

4.5 Retrieval

Once we have classifiers trained for all QT methods, we can apply them to a given query on-the-fly, and compute query-specific combination weights. One approach is *hard weighting*, putting all weight onto a single method — when there are more than one methods classified with label 1, we can either pick one randomly or use the classifier confidence value as a tie-breaker. An alternative is *soft weighting*, where the weight of the i^{th} method can be computed either using classifier confidence C_i (i.e., how confident the model is that the i^{th} method will perform well), precision on tuning set precision_i (i.e., how precise the model is at its pre-

[3] We used the BOLT collection in our experiments.

dictions for the i^{th} method), or both:

$$\begin{aligned} w_i^{s1}(q) =& C_i(q) \\ w_i^{s2}(q) =& \text{precision}_i(1) \times l_i(q) \\ &+(1-\text{precision}_i(0)) \times (1 - l_i(q)) \\ w_i^{s3}(q) =& \text{precision}_i(1) \times C_i(q) \\ &+(1-\text{precision}_i(0)) \times (1 - C_i(q)) \end{aligned}$$

The intuition behind all of these weighting schemes is to produce a weight for each QT method, by taking into account the confidence of the classifier, and/or the precision of the classifier on tuning instances.

The computed weights are normalized before constructing the final query for retrieval:

$$w_i^{\text{final}}(q) = w_i(q) / \textstyle\sum_{j=1}^{m} w_j(q)$$

When compared empirically, we noticed that soft weighting is more effective than hard weighting, as the latter is more sensitive to classifier errors. Among the three soft weighting functions, differences were mostly negligible in our experiments. Hence, we decided to use the simplest weighting function w^{s1}.

4.6 Analytical Model

It is time-consuming to implement various combination-of-evidence approaches and run retrieval experiments. Therefore, it is useful to have an analytical model of the process that can provide a rough estimate of how fruitful it would be to spend this effort, given certain details about the task. The model we present in this section estimates the effectiveness of combining QT methods $1 \ldots m$ on a query set Q, given (1) the effectiveness of each method on Q and (2) error rate of binary classifiers $C_1 \ldots C_m$ on Q. Using this formulation, one can assess the benefit of combination without running retrieval, based only on error rates — this saves precious time during development. Moreover, even without trained classifiers, this model can be used to estimate potential benefits by plugging in *hypothetical* error values. In other words, one can ask the question "If I had classifiers with x% error on this query set, what would be the benefit of using these classifiers to combine QT methods?" before developing any combination approach at all.

The analytical model considers a special case of weighted combination: for each query q, we pick a single QT method $i = 1 \ldots m$, for which the classifier predicts a label of 1. If there are more than one such method, one of them is picked randomly. This simplified version allows us to compute expected effectiveness for q as follows:

$$E[f(q)] = \sum_{\text{method } i} \mathbf{Pr}(\text{pick } i|q) f_i(q)$$

While $f_i(q)$ is an observed value (the effectiveness of the i^{th} method on query q), $\mathbf{Pr}(\text{pick } i|q)$ needs to be estimated (the probability of selecting the i^{th} method). Since this depends on the predicted labels, we consider all possible scenarios $l = l_1 l_2 \ldots l_m$, where each value is the prediction of a classifier. For instance, "l=010" means that classifiers C_1 and C_3 predicted a label of 0, while C_2 predicted a positive label. Marginalizing over the 2^m possible scenarios gives us the following estimate:

$$\begin{aligned} &\mathbf{Pr}(\text{pick } i|q) \\ &= \left(\sum_{l_1=0}^{1} \ldots \sum_{l_m=0}^{1} \mathbf{Pr}(l|q) \right) \times \mathbf{Pr}(\text{pick } i|l, q) \\ &= \left(\sum_{l_1=0}^{1} \ldots \sum_{l_m=0}^{1} \prod_{i=1}^{m} \mathbf{Pr}(l_i|q) \right) \times \mathbf{Pr}(i|l, q) \end{aligned}$$

In the final step, we assumed that classifiers make predictions independent of each other, which is a desired property for successful combination. $\mathbf{Pr}(l_i|q)$ can be estimated using classifier error statistics:

$$\mathbf{Pr}(l_i|q) \sim \frac{\text{count}(\text{predicted} = l_i, \text{true} = l_q)}{\text{count}(\text{true} = l_q)}$$

where l_q is the true label of q. If $l_i = l_q$, this expression becomes the true positive or true negative rate, depending on the value. Similarly, if $l_i \neq l_q$, it is either the false positive or false negative rate.

Finally, the probability that the i^{th} method is selected in a particular scenario depends solely on the predicted labels, since it is a random selection: $\mathbf{Pr}(\text{pick } i|l) = l_i / \sum_{j=1}^{m} l_j$

This concludes the derivation of the analytical model of query evidence combination, which we use in Section 5.1 to evaluate the effectiveness of labeling approaches.

5 Evaluation

We evaluated our approach on four different CLIR tasks: TREC 2002 English-Arabic CLIR, NTCIR-8 English-Chinese Advanced Cross-Lingual Infor-

mation Access (ACLIA), and two forum post retrieval tasks as part of the DARPA Broad Operational Language Technologies (BOLT) program: English-Arabic (BOLT_{ar}) and English-Chinese (BOLT_{ch}). The query language is English in all cases, and we preprocess the queries using BBN's information extraction toolkit SERIF (Ramshaw et al., 2011). State-of-the-art English-Arabic (En-Ar) and English-Chinese (En-Ch) MT systems were trained on parallel corpora released in NIST OpenMT 2012, in addition to parallel forum data collected as part of the BOLT program (10m En-Ar words; 30m En-Ch words). From these data, word alignments were learned with GIZA++ (Och and Ney, 2003), using five iterations of each of IBM Models 1–4 and HMM.

3-gram Chinese and 5-gram Arabic Kneser-Ney language models were trained from the Gigaword corpus (1b words each) and non-English side of the training corpus. Chinese and English parallel text were preprocessed through the Treebank Tokenizer,[4] while no special treatment was performed on Arabic.

For retrieval, we used `Indri`, a state-of-the-art probabilistic relevance model that supports weighted query representations through operators `#combine` and `#weight` (Metzler and Croft, 2005). A character-based index was built for Chinese collections, whereas Arabic text was stemmed using `Lucene` before indexing.[5] English text was preprocessed by `Indri`'s implementation of the Porter stemmer (Porter, 1997). Statistics for each collection and query set are summarized in Table 1.

Before performing any combination, we first ran the three baseline QT methods individually and evaluated the retrieved documents. Mean average precision (MAP) was used to measure retrieval effectiveness, which is a widely used and stable metric, estimating the area under the precision-recall curve. We set $n = 10$ for the n-best probabilistic translation method. Baseline scores are reported in Table 2. The average precision (AP) of each query in these tasks was used to label the query and construct training data accordingly.

In subsequent sections, we evaluate the effect of several variants in the training pipeline.

[4] `http://www.cis.upenn.edu/~treebank`

[5] `http://lucene.apache.org`

5.1 Effect of Labeling

In Section 4.2, we introduced two ways to label instances. In our evaluation, we set the free parameters $k_1 = k_2 = 90$, which filters out 33% of queries from the training set of the BOLT_{ar} task; this percentage is 29% in BOLT_{ch}, 44% in TREC, and 27% in NTCIR.

Labeling determines which query translation method is considered effective or not, which consequently determines what the "learning problem" is (since the objective of the classifier is to separate differently labeled instances). As a result, there are two dimensions to consider when comparing labeling strategies. One is the accuracy of the classifiers on held-out data, and the other is how well the trained classifier reflects this accuracy when used in retrieval. To clarify the distinction, consider a case where every instance is labeled 1. This generates a trivial learning problem with no test errors, yet this does not entail that using these classifiers in retrieval will be more effective than other labeling strategies. If, even with high classifier accuracy, the retrieval effectiveness is low, that indicates a bad choice for labeling.

We can *theoretically analyze* how suitable each labeling method is by applying the analytical model to each CLIR task, setting parameters based on a perfect classifier: true positive/negative rate of 1 and false positive/negative rate of 0 (see Section 4.6). Table 2 shows these results in the "Perfect" column, since these scores represent what *could be* achieved if classifiers were trained to predict labels *perfectly* (no training or retrieval is actually performed). There are two values in each row of the "Perfect" column, one for each labeling strategy. In each row, we found these two values to be statistically significantly higher than any of the baseline scores. This shows that both labeling approaches have the *potential* to improve effectiveness significantly.

We also made an empirical comparison of the two labeling approaches by actually training classifiers with each labeling, and then using the classifiers to combine query translations in retrieval. The "Trained" column in Table 2 shows the MAP we get on each CLIR task (and average classifier accuracies), using either labeling.[6]

Based on these results, we conclude that best-

[6] For a fair comparison, we fixed the train-tune setting to fully-open, trained classifiers on the test collection and reported leave-one-out accuracies.

Lang	Collection		Topics	MT Training data	
	Source	Size (docs)		Source (domain)	Size (words)
Arabic	TREC-02	383,872	50	OpenMT-12 (news/web) BOLT (forum)	10m
Arabic	BOLT	12,258,904	45		
Chinese	NTCIR-8	388,589	100	OpenMT-12 (news/web) BOLT (forum)	30m
Chinese	BOLT	6,693,951	45		

Table 1: Summary of the CLIR tasks in our evaluations.

Task	Baseline			Perfect		Trained	
	one-best	ten-best	word	measure	rank	measure	rank
BOLT_{ar}	0.296	0.311	0.318	*0.341*	*0.341*	0.342 (74)	0.330 (72)
BOLT_{ch}	0.370	0.406	0.407	*0.458*	*0.462*	0.438 (68)	0.426 (60)
TREC	0.292	0.298	0.301	*0.327*	*0.330*	0.305 (59)	0.316 (59)
NTCIR	0.146	0.152	0.141	*0.180*	*0.177*	0.163 (56)	0.162 (61)

Table 2: Retrieval effectiveness of baseline QT methods is presented on the left side, and a comparison of labeling strategies is provided on the right side. All numbers represent MAP values, except for classifier accuracy shown in percentage values (in parantheses). Analytically computed values are shown in italics.

by-measure labeling is more useful in practice, supported by typically higher accuracy and effectiveness. Best-by-rank yields better results only on TREC, but a closer look reveals that the increase in MAP is due to only two outlier queries. For BOLT_{ar}, on the other hand, retrieval with best-by-measure labeling is more effective (statistically significant) than best-by-rank; hence, the former is used in remaining parts of our evaluation.

5.2 Effect of Train-Tune Setting

In Section 4.4, we introduced three major train-tune settings: fully-open, half-blind, and fully-blind. In order to implement these settings, we treat each of the three query sets (BOLT, TREC, NTCIR) as a separate training dataset and experiment with a variety of combinations.

For simplicity, let us demonstrate the variety of experiments assuming the test collection is BOLT. For the fully-open case, the default training data is all of the BOLT queries (this training set is referred to as *b*). Additionally, one can include queries from TREC (referred to as *t*) and NTCIR (referred to as *n*) into the training data. This gives us four different training datasets for the fully-open case: b, $b + n$, $b + t$, $b + t + n$. Similarly, each of the half-blind and fully-blind settings can be applied to three different training sets: For BOLT, these are t, n, $t + n$.[7] This results in ten different experiments run for each task — in each experiment, we train a classifier for each QT method, select the best meta-parameters on the tuning set, and then compute combination weights for retrieval using the classifiers.

Each cell on the left side of Table 3 (under column "Query-specific Combination") shows the results of the most effective experiment for a particular task and train-tune setting. Accuracy values for classifiers varied widely across these experiments. Still, even when accuracies dropped close to or below 50% (i.e. random baseline), combined retrieval was always more effective than any single QT approach, which emphasizes the robustness of our approach. For instance, in the fully-blind setting for the NTCIR task, the individual classifiers had accuracies of only 56%, 49%, and 44% but MAP was 0.163, which is higher than the MAP of any individual method for that collection (0.146, 0.152, or 0.141).

Another key observation in Table 3 is that the domain effect (i.e., training and/or tuning on queries similar to test queries) is only noticeable on the two BOLT tasks. For NTCIR and TREC, we do not observe a boost in MAP when queries from the same task are included in training (i.e., fully-open setting). This can be explained by the BOLT-centric nature of our system components: the text analysis tool and MT systems are tuned mainly for forum data, and the collection-based features are extracted from BOLT. Due to this bias, BOLT queries were most useful in our experiments, supported by the fact that BOLT is always

[7] In the case of half-blind, b is split into two: 20% is used for tuning and the remainder is used for testing.

Task	Query-specific Combination						Uniform	Task-specific	Max
	fully-open		half-blind		fully-blind				
$BOLT_{ar}$	**0.342***†	b	0.330	t+n	0.329	t+n	0.324^{12}	0.329^{1}	0.346
$BOLT_{ch}$	**0.438***†	b	0.428	n	0.426	t+n	0.422^{1}	0.431^{1}	0.466
TREC	0.321	b+t	**0.324***†	b+n	0.321	b+n	0.314^{1}	0.318^{1}	0.332
NTCIR	**0.164***	b+n	0.163	b+t	0.163	b	0.162^{13}	0.162^{13}	0.182

Table 3: A comparison of query combination approaches. For query-specific combination, MAP and training data are shown for the most effective experiment of each train-tune setting. For each task, the highest MAP achieved with our approach is shown in bold. Superscripts 1, 2, and 3 indicate statistically significant improvements over baseline methods one-best, probabilistic 10-best, and word-based, whereas * indicates improvements over all three. Superscript † indicates results significantly better than uniform and task-specific combination methods.

included in the train set when testing on TREC or NTCIR (see lowest two rows in Table 3). Also, when there is no domain effect (i.e., half-blind and fully-blind), more data yields higher effectiveness in 6 out of 8 cases (see two right columns on the left side of Table 3).

5.3 Retrieval Effectiveness

In this section, we compare our novel query-specific combination-of-evidence approach to the baseline CLIR approaches, as well as comparable combination methods (uniform and task-specific combination) in terms of retrieval effectiveness. Based on a randomized significance test (Smucker et al., 2007), the best query-specific combination method (shown in boldface in Table 3) outperforms all baseline QT methods in all tasks with 95% confidence (indicated by superscript * in Table 3). This is not the case for uniform or task-specific query combination, which are statistically indistinguishable from at least one of the QT methods, depending on the task (indicated by superscripts 1, 2, and 3 for one-best, probabilistic 10-best, and word-based QT methods, respectively). When we directly compare our query-specific combination approach to other combination methods, the differences are statistically significant for all tasks but NTCIR (indicated by superscript †).

For reference, we also computed effectiveness for a hypothetical system (denoted by "Max" in Table 3) that could select the best QT method for each query and use only that for retrieval. This is not a strict upper bound, since correctly weighting each method can produce better results, but it is still a reasonable target for effectiveness. In our experiments, Arabic retrieval runs were very close to this target with our combination approach, while the gap for Chinese is still substantial, which is worth further exploration.

6 Conclusions and Future Work

In this paper, we introduced a novel combination-of-evidence approach for CLIR, which *learns a custom combination recipe* for each query. We formulate this as a set of binary classification problems, and show that trained classifiers can be used to produce query-specific combination weights effectively. Our deep exploration of many variants (e.g., labeling, training-tuning, weight computation, analytical formulation) and extensive empirical analysis on four different tasks provide insights for future research on the under-studied problem of combining translations for CLIR.

Our approach advances the state of the art of CLIR, yielding higher effectiveness than three advanced query translation approaches, all based on state-of-the-art MT systems. Furthermore, on three of the four tasks, our combination strategy is statistically significantly better than two comparable combination techniques. Experimental results also suggest that even a uniform combination of query translations is consistently better than any individual method. While it is known that combining translations helps CLIR, we confirm this on a set of modern CLIR tasks, including two target languages and a variety of text domains.

Having a simple linear learning problem allows us to train robust models with relatively simpler features. Nevertheless, we are interested in experimenting with more sophisticated learning approaches. In terms of non-linear classifiers, our experience with decision trees in this paper indicated a higher tendency to overfit. In terms of

combining queries in a non-linear fashion, our future plans include integrating our approach into a LTR framework, and directly optimize MAP. This will also allow us to explore more complex features extracted from query *and* document text, as well as external sources.

Another possible future endeavor is to extend these ideas to (i) other query translation approaches and (ii) document translation. While the exact same problem can be formulated for learning to translate documents effectively, a more complicated infrastructure and longer running times are two challenges that need to be considered.

Finally, we hope this to be a significant step towards more context-dependent and robust CLIR models, by taking advantage of modern translation technologies, as well as machine learning techniques.

Acknowledgments

This work was supported by DARPA/I2O Contract No. HR0011-12-C-0014 under the BOLT program (Approved for Public Release, Distribution Unlimited). The views, opinions, and/or findings contained in this article are those of the author and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

References

Hosein Azarbonyad, Azadeh Shakery, and Heshaam Faili. 2013. Exploiting multiple translation resources for english-persian cross language information retrieval. In *Proceedings of the Cross-Language Evaluation Forum on Cross-Language Information Retrieval and Evaluation*, CLEF '13, pages 93–99.

Lisa Ballesteros and W. Bruce Croft. 1996. Dictionary methods for cross-lingual information retrieval. In *Proceedings of the 7th International DEXA Conference on Database and Expert Systems Applications*, pages 791–801.

Nicholas J. Belkin, Paul Kantor, Edward A. Fox, and Joseph A. Shaw. 1995. Combining the evidence of multiple query representations for information retrieval. *Information Processing & Management*, 31(3):431–448, May.

Michael Bendersky, Donald Metzler, and W. Bruce Croft. 2011. Parameterized concept weighting in verbose queries. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 605–614, New York, NY, USA. ACM.

Pierre-Yves Berger and Jacques Savoy. 2007. Selecting automatically the best query translations. In *Large Scale Semantic Access to Content (Text, Image, Video, and Sound)*, RIAO '07, pages 287–300, Paris, France, France. Le Centre de Hautes Etudes Internationales D'Informatique Documentaire.

Martin Braschler. 2004. Combination approaches for multilingual text retrieval. *Information Retrieval*, 7(1-2):183–204, January.

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

Aitao Chen and Fredric C. Gey. 2004. Multilingual information retrieval using machine translation, relevance feedback and decompounding. *Inf. Retr.*, 7(1-2):149–182, January.

W. Bruce Croft. 2000. Combining approaches to information retrieval. In W. Bruce Croft, editor, *Advances in Information Retrieval*, volume 7 of *The Information Retrieval Series*, pages 1–36. Springer.

Kareem Darwish and Douglas W. Oard. 2003. Probabilistic structured query methods. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 338–344.

Edward A. Fox. 1983. *Extending the Boolean and Vector Space Models of Information Retrieval with P-norm Queries and Multiple Concept Types*. Ph.D. thesis, Cornell University, Ithaca, NY, USA. AAI8328584.

Fredric C. Gey, Hailing Jiang, Vivien Petras, and Aitao Chen. 2001. Cross-language retrieval for the clef collections - comparing multiple methods of retrieval. In *Revised Papers from the Workshop of Cross-Language Evaluation Forum on Cross-Language Information Retrieval and Evaluation*, CLEF '00, pages 116–128, London, UK, UK. Springer-Verlag.

Benjamin Herbert, György Szarvas, and Iryna Gurevych. 2011. Combining query translation techniques to improve cross-language information retrieval. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval*, ECIR'11, pages 712–715, Berlin, Heidelberg. Springer-Verlag.

Djoerd Hiemstra, Wessel Kraaij, Renée Pohlmann, and Thijs Westerveld. 2001. Translation resources, merging strategies, and relevance feedback for cross-language information retrieval. In *Revised Papers from the Workshop of Cross-Language Evaluation Forum on Cross-Language Information Retrieval and Evaluation*, CLEF '00, pages 102–115, London, UK, UK. Springer-Verlag.

David A. Hull and Gregory Grefenstette. 1996. Querying across languages: a dictionary-based approach to multilingual information retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, pages 49–57.

Kui-Lam Kwok. 1999. English-Chinese cross-language retrieval based on a translation package. In *Workshop on Machine Translation for Cross Language Information Retrieval, Machine Translation Summit VII*, pages 8–13.

Patrice Lopez and Laurent Romary. 2009. Patatras: Retrieval model combination and regression models for prior art search. In *Proceedings of the 10th Cross-language Evaluation Forum Conference on Multilingual Information Access Evaluation: Text Retrieval Experiments*, CLEF'09, pages 430–437, Berlin, Heidelberg. Springer-Verlag.

Yanjun Ma, Jian-Yun Nie, Hua Wu, and Haifeng Wang. 2012. Opening machine translation black box for cross-language information retrieval. In *Information Retrieval Technology*, pages 467–476. Springer.

Walid Magdy and Gareth J. F. Jones. 2011. Should MT systems be used as black boxes in CLIR? In *Proceedings of the 33rd European Conference on Information Retrieval*, ECIR '11, pages 683–686.

Michael McGill, Matthew Koll, and Terry Noreault. 1979. *An Evaluation of Factors Affecting Document Ranking by Information Retrieval Systems*. ERIC reports. School of Information Studies, Syracuse University.

Donald Metzler and W. Bruce Croft. 2005. A Markov random field model for term dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 472–479.

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

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2012. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490.

Carol Peters, Martin Braschler, and Paul Clough. 2012. *Multilingual Information Retrieval - From Research To Practice*. Springer.

Ari Pirkola. 1998. The effects of query structure and dictionary-setups in dictionary-based cross-language information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 55–63.

M. F. Porter. 1997. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

T. B. Rajashekar and W. Bruce Croft. 1995. Combining automatic and manual index representations in probabilistic retrieval. *J. Am. Soc. Inf. Sci.*, 46(4):272–283, May.

Lance Ramshaw, Elizabeth Boschee, Marjorie Freedman, Jessica MacBride, Ralph Weischedel, and Alex Zamanian. 2011. Serif language processing — effective trainable language understanding. In J. Olive et al., editor, *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*, pages 626–631. Springer.

Jacques Savoy. 2001. Report on CLEF-2001 experiments: Effective combined query-translation approach. In *CLEF*, pages 27–43.

Mark D. Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the 16th ACM conference on Conference on Information and Knowledge Management*, CIKM '07, pages 623–632.

Ferhan Ture, Jimmy Lin, and Douglas W. Oard. 2012. Combining statistical translation techniques for cross-language information retrieval. In *Proceedings of the 24th International Conference on Computational Linguistics*, COLING '12, pages 2685–2702.

Howard Turtle and W. Bruce Croft. 1990. Inference networks for document retrieval. In *Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '90, pages 1–24, New York, NY, USA. ACM.

Jinxi Xu and Ralph Weischedel. 2005. Empirical studies on the impact of lexical resources on CLIR performance. *Information Processing & Management*, 41(3):475–487, May.

Semantic-Based Multilingual Document Clustering via Tensor Modeling

Salvatore Romeo, Andrea Tagarelli
DIMES, University of Calabria
Arcavacata di Rende, Italy
sromeo@dimes.unical.it
tagarelli@dimes.unical.it

Dino Ienco
IRSTEA, UMR TETIS
Montpellier, France
LIRMM
Montpellier, France
dino.ienco@irstea.fr

Abstract

A major challenge in document clustering research arises from the growing amount of text data written in different languages. Previous approaches depend on language-specific solutions (e.g., bilingual dictionaries, sequential machine translation) to evaluate document similarities, and the required transformations may alter the original document semantics. To cope with this issue we propose a new document clustering approach for multilingual corpora that (i) exploits a large-scale multilingual knowledge base, (ii) takes advantage of the multi-topic nature of the text documents, and (iii) employs a tensor-based model to deal with high dimensionality and sparseness. Results have shown the significance of our approach and its better performance w.r.t. classic document clustering approaches, in both a balanced and an unbalanced corpus evaluation.

1 Introduction

Document clustering research was initially focused on the development of general purpose strategies to group unstructured text data. Recent studies have started developing new methodologies and algorithms that take into account both linguistic and topical characteristics, where the former include the size of the text and the type of language, and the latter focus on the communicative function and targets of the documents.

A major challenge in document clustering research arises from the growing amount of text data that are written in different languages, also due to the increased popularity of a number of tools for collaboratively editing through contributors across the world. *Multilingual document clustering* (MDC) aims to detect clusters in a collection of texts written in different languages. This can aid a variety of applications in cross-lingual information retrieval, including statistical machine translation and corpora alignment.

Existing approaches to MDC can be divided in two broad categories, depending on whether a parallel corpus rather than a comparable corpus is used (Kumar et al., 2011c). A *parallel corpus* is typically comprised of documents with their related translations (Kim et al., 2010). These translations are usually obtained through machine translation techniques based on a selected anchor language. Conversely, a *comparable corpus* is a collection of multilingual documents written over the same set of classes (Ni et al., 2011; Yogatama and Tanaka-Ishii, 2009) without any restriction about translation or perfect correspondence between documents. To mine this kind of corpus, external knowledge is employed to map concepts or terms from a language to another (Kumar et al., 2011c; Kumar et al., 2011a), which enables the extraction of cross-lingual document correlations. In this case, a major issue lies in the definition of a cross-lingual similarity measure that can fit the extracted cross-lingual correlations. Also, from a semi-supervised perspective, other works attempt to define must-link constraints to detect cross-lingual clusters (Yogatama and Tanaka-Ishii, 2009). This implies that, for each different dataset, the set of constraints needs to be redefined; in general, the final results can be negatively affected by the quantity and the quality of involved constraints (Davidson et al., 2006).

To the best of our knowledge, existing clustering approaches for comparable corpora are customized for a small set (two or three) of languages (Montalvo et al., 2007). Most of them are not generalizable to many languages as they employ bilingual dictionaries and the translation is performed sequentially considering only pairs of languages. Therefore, the order in which this process is done can seriously impact the results. Another common drawback concerns the way most of the recent approaches perform their analysis: the various languages are analyzed independently of each other (possibly by exploiting external knowledge like Wikipedia to enrich documents (Kumar et al., 2011c; Kumar et al., 2011a)), and then the language-specific results are merged. This two-step analysis however may fail in profitably exploiting cross-language information from the multilingual corpus.

Contributions. We address the problem of MDC by proposing a framework that features three key elements, namely: (1) to model documents over a unified conceptual space, with the support of a large-scale multilingual knowledge base; (2) to decompose the multilingual documents into topically-cohesive segments; and (3) to describe the multilingual corpus under a multi-dimensional data structure.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 600–609,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

The first key element prevents loss of information due to the translation of documents from different languages to a target one. It enables a conceptual representation of the documents in a language-independent way preserving the content semantics. BabelNet (Navigli and Ponzetto, 2012a) is used as multilingual knowledge base. To the extent of our knowledge, this is the first work in MDC that exploits BabelNet.

The second key element, document segmentation, enables us to simplify the document representation according to their multi-topic nature. Previous research has demonstrated that a segment-based approach can significantly improve document clustering performance (Tagarelli and Karypis, 2013). Moreover, the conceptual representation of the document segments enables the grouping of linguistically different (portions of) documents into topically coherent clusters.

The latter aspect is leveraged by the third key element of our proposal, which relies on a tensor-based model (Kolda and Bader, 2009) to effectively handle the high dimensionality and sparseness in text. Tensors are considered as a multi-linear generalization of matrix factorizations, since all dimensions or modes are retained thanks to multi-linear structures which can produce meaningful components. The applicability of tensor analysis has recently attracted growing attention in information retrieval and data mining, including document clustering (e.g., (Liu et al., 2011; Romeo et al., 2013)) and cross-lingual information retrieval (e.g., (Chew et al., 2007)).

The rest of the paper is organized as follows. Section 2 provides an overview of BabelNet and basic notions on tensors. We describe our proposal in Section 3. Data and experimental settings are described in Section 4, while results are presented in Section 5. We summarize our main findings in Section 6, finally Section 7 concludes the paper.

2 Background

2.1 BabelNet

BabelNet (Navigli and Ponzetto, 2012a) is a multilingual semantic network obtained by linking Wikipedia with WordNet, that is, the largest multilingual Web encyclopedia and the most popular computational lexicon. The linking of the two knowledge bases was performed through an automatic mapping of WordNet synsets and Wikipages, harvesting multilingual lexicalization of the available concepts through human-generated translations provided by the Wikipedia interlanguage links or through machine translation techniques. The result is an encyclopedic dictionary containing concepts and named entities lexicalized in 50 different languages.

Multilingual knowledge in BabelNet is represented as a labeled directed graph in which nodes are concepts or named entities and edges connect pairs of nodes through a semantic relation. Each edge is labeled with a relation type (is-a, part-of, etc.), while each node corresponds to a *BabelNet synset*, i.e., a set of lexicalizations of a concept in different languages.

BabelNet can be accessed and easily integrated into applications by means of a Java API provided by the toolkit described in (Navigli and Ponzetto, 2012b). The toolkit also provides functionalities for *graph-based WSD in a multilingual context*. Given an input set of words, a semantic graph is built by looking for related synset paths and by merging all them in a unique graph. Once the semantic graph is built, the graph nodes can be scored with a variety of algorithms. Finally, this graph with scored nodes is used to rank the input word senses by a graph-based approach.

2.2 Tensor model representation

A tensor is a multi-dimensional array $\boldsymbol{\mathcal{T}} \in \Re^{I_1 \times I_2 \times \cdots \times I_M}$. The number of dimensions M, also known as *ways* or *modes*, is called *order* of the tensor, so that a tensor with order M is also said a M-way or M-order tensor. A *higher-order* tensor (i.e., a tensor with order three or higher) is denoted by boldface calligraphic letters, e.g., $\boldsymbol{\mathcal{T}}$; a matrix (2-way tensor) is denoted by boldface capital letters, e.g., $\mathbf{U}$; a vector (1-way tensor) is denoted by boldface lowercase letters, e.g., $\mathbf{v}$. The generic entry (i_1, i_2, i_3) of a third-order tensor $\boldsymbol{\mathcal{T}}$ is denoted by $t_{i_1 i_2 i_3}$, with $i_1 \in [1..I_1], i_2 \in [1..I_2], i_3 \in [1..I_3]$.

A one-dimensional fragment of tensor, defined by varying one index and keeping the others fixed, is a 1-way tensor called *fiber*. A third-order tensor has column, row and tube fibers. Analogously, a two-dimensional fragment of tensor, defined by varying two indices and keeping the rest fixed, is a 2-way tensor called *slice*. A third-order tensor has horizontal, lateral and frontal slices.

The *mode-m matricization* of a tensor $\boldsymbol{\mathcal{T}}$, denoted by $\mathbf{T}_{(\mathrm{m})}$, is obtained by arranging the mode-m fibers as columns of a matrix. A third-order tensor $\boldsymbol{\mathcal{T}} \in \Re^{I_1 \times I_2 \times I_3}$ is *all-orthogonal* if $\sum_{i_1 i_2} t_{i_1 i_2 \alpha} t_{i_1 i_2 \beta} = \sum_{i_1 i_3} t_{i_1 \alpha i_3} t_{i_1 \beta i_3} = \sum_{i_2 i_3} t_{\alpha i_2 i_3} t_{\beta i_2 i_3} = 0$ whenever $\alpha \neq \beta$. The *mode-m product* of a tensor $\boldsymbol{\mathcal{T}} \in \Re^{I_1 \times I_2 \times \cdots \times I_M}$ with a matrix $\mathbf{U} \in \Re^{J \times I_m}$, denoted by $\boldsymbol{\mathcal{T}} \times_m \mathbf{U}$, is a tensor of dimension $I_1 \times \ldots I_{m-1} \times J \times I_{m+1} \times \cdots \times I_M$ and can be expressed in terms of matrix product as $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{T}} \times_m \mathbf{U}$, whose mode-$m$ matricization is $\mathbf{Y}_{(\mathrm{m})} = \mathbf{U}\mathbf{T}_{(\mathrm{m})}$.

3 Our Proposal

3.1 Multilingual Document Clustering framework

We are given a collection of multilingual documents $\mathcal{D} = \bigcup_{l=1}^{L} \mathcal{D}_l$, where each $\mathcal{D}_l = \{d_{l_i}\}_{i=1}^{N_l}$ represents a subset of documents written in the same language, with $N = \sum_{l=1}^{L} N_l = |\mathcal{D}|$. Our framework can be applied to any multilingual document collection regardless of the languages, and can deal with balanced as well as

Algorithm 1 *SeMDocT* (**Se**gment-based **M**ultiLingual **Doc**ument Clustering via **T**ensor Modeling)

Input: A collection of multilingual documents $\mathcal{D}$, the number k of segment clusters, the number of tensorial components r.
Output: A document clustering solution $\mathcal{C}$ over $\mathcal{D}$.

1: Apply a text segmentation algorithm over each of the documents in $\mathcal{D}$ to produce a collection of document segments $\mathcal{S}$. */* Section 3.1.1 */*
2: Represent $\mathcal{S}$ in either a bag-of-words (BoW) or a bag-of-synsets (BoS) space. */* Section 3.1.2 */*
3: Apply any document clustering algorithm on $\mathcal{S}$ to obtain a segment clustering $\mathcal{C}^{\mathcal{S}} = \{C_i^s\}_{i=1}^k$. */* Section 3.1.2 */*
4: Represent $\mathcal{C}^{\mathcal{S}}$ in either a bag-of-words (BoW) or a bag-of-synsets (BoS) space. */* Section 3.1.3 */*
5: Model $\mathcal{S}$ as a third-order tensor $\boldsymbol{\mathcal{T}} \in \Re^{I_1 \times I_2 \times I_3}$, with $I_1 = |\mathcal{D}|$, $I_2 = |\mathcal{F}|$, and $I_3 = k$. */* Section 3.1.4 */*
6: Decompose the tensor using a Truncated HOSVD. */* Section 3.1.4 */*
7: Apply a document clustering algorithm on the mode-1 factor matrix to obtain the final clusters of documents $\mathcal{C} = \{C_i\}_{i=1}^K$. */* Section 3.1.5 */*

unbalanced corpora. Therefore, no restriction is given on both the number L of languages and the distribution of documents over the languages (i.e., $N_i \lesseqgtr N_j$, with $i, j = 1..L, i \neq j$).

Real-world documents often span multiple topics. We assume that each document in $\mathcal{D}$ is relatively long to be comprised of smaller textual units, or *segments*, each of which can be considered cohesive w.r.t. a topic over the document. This represents a key aspect in our framework as it enables the use of a *tensor model* to conveniently address the multi-faceted nature of the documents.

Our overall framework, named *SeMDocT* (**Se**gment-based **M**ultiLingual **Doc**ument Clustering via **T**ensor Modeling), is shown in Algorithm 1. In the following, we shall describe in details each of the steps involved in *SeMDocT*.

3.1.1 Computing within-document segments

Text segmentation is concerned with the fragmentation of an input text into multi-paragraph, contiguous and disjoint blocks that represent subtopics. Regardless of the presence of logical structure clues in the document, linguistic criteria (Beeferman et al., 1999) and statistical similarity measures (Hearst, 1997; Choi et al., 2001; Cristianini et al., 2001) have been mainly used to detect subtopic boundaries between segments. A common assumption is that terms that discuss a subtopic tend to co-occur locally, and a switch to a new subtopic is detected by the ending of co-occurrence of a given set of terms and the beginning of the co-occurrence of another set of terms.

Our *SeMDocT* does not depend on a specific algorithmic choice to perform text segmentation; in this work, we refer to the classic *TextTiling* (Hearst, 1997), which is the exemplary similarity-block-based method for text segmentation.

3.1.2 Inducing document segment clusters

The result of the previous step is a collection of document segments, henceforth denoted as $\mathcal{S}$. Each segment in $\mathcal{S}$ is represented as a vector of feature occurrences, where a feature can be either *lexical* or *semantic*. This corresponds to two alternative representation models: the standard *bag-of-words* (henceforth *BoW*), whereby features correspond to lemmatized, non-stopword terms, and the obtained feature space results from the union of the vocabularies of the different languages; and *bag-of-synsets* (henceforth *BoS*), whereby features correspond to BabelNet synsets. We shall devote Section 3.2 to a detailed description of our proposed BoS representation.

The segment collection $\mathcal{S}$ is given in input to a document clustering algorithm to produce a clustering of the segments $\mathcal{C}^{\mathcal{S}} = \{C_i^s\}_{i=1}^k$. The obtained clusters of segments can be disjoint or overlapping. Again, our *SeMDocT* is parametric to the clustering algorithm as well; here, we resort to a state-of-the-art clustering algorithm, namely *Bisecting K-Means* (Steinbach et al., 2000), which is widely known to produce high-quality (hard) clustering solutions in high-dimensional, large datasets (Zhao and Karypis, 2004). Note however that it requires as input the number of clusters. To cope with this issue, we adopt the method described in (Salvador and Chan, 2004), which explores how the within-cluster cohesion changes by varying the number of clusters. The number of clusters for which the slope of the plot changes drastically is chosen as a suitable value for the clustering algorithm.

3.1.3 Segment-cluster based representation

Upon the segment clustering, each document is represented by its segments assigned to possibly multiple segment clusters. Therefore, we derive a document-feature matrix for each of the k segment clusters. The features correspond either to the BoW or BoS model, according to the choice made for the segment representation.

Let us denote with $\mathcal{F}$ the feature space for all segments in $\mathcal{S}$. Given a segment cluster C^s, the corresponding document-feature matrix is constructed as follows. The representation of each document $d \in \mathcal{D}$ w.r.t. C^s is a vector of length $|\mathcal{F}|$ that results from the sum of the feature vectors of the d's segments belonging to C^s. Moreover, in order to weight the appearance of a document in a cluster based on its segment-based portion covered in the cluster, the document vector of d w.r.t. C^s is finally obtained by multiplying the sum of the segment-vectors by a scalar representing the portion of d's features that appear in the segments belonging to C^s. The document-feature matrix of C^s resulting from the previous step is finally normalized by column.

3.1.4 Tensor model and decomposition

The document-feature matrices corresponding to the k segment-clusters are used to form a third-order tensor.

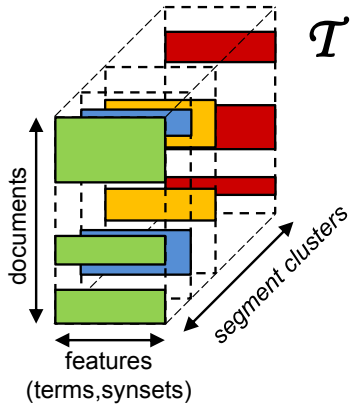


Figure 1: The third-order tensor model for the representation of a multilingual document collection based on segment clusters.

Our third-order tensor model is built by arranging as frontal slices the k segment-cluster matrices. The resulting tensor will be of the form $\boldsymbol{\mathcal{T}} \in \Re^{I_1 \times I_2 \times I_3}$, with $I_1 = |\mathcal{D}|$, $I_2 = |\mathcal{F}|$, and $I_3 = k$. The proposed tensor model is sketched in Fig. 1.

The resulting tensor is decomposed through a Truncated Higher Order SVD (T-HOSVD) (Lathauwer et al., 2000) in order to obtain a low-dimensional representation of the segment-cluster-based representation of the document collection. The T-HOSVD can be considered as an extension of the Truncated Singular Value Decomposition (T-SVD) to the case of three or more dimensions. For a third-order tensor $\boldsymbol{\mathcal{T}} \in \Re^{I_1 \times I_2 \times I_3}$ the T-HOSVD is expressed as

$$\boldsymbol{\mathcal{T}} \approx \boldsymbol{\mathcal{X}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}$$

where $\mathbf{U}^{(m)} = [\mathbf{u}_1^{(m)} \mathbf{u}_2^{(m)} \dots \mathbf{u}_{r_m}^{(m)}] \in \Re^{I_m \times r_m}$ $(m = 1, 2, 3)$ are orthogonal matrices, $r_m \ll I_m$, and the core tensor $\boldsymbol{\mathcal{X}} \in \Re^{r_1 \times r_2 \times r_3}$ is an all-orthogonal and ordered tensor. T-HOSVD can be computed in two steps:

1. For $m \in \{1, 2, 3\}$, compute the unfolded matrices $\mathbf{T}_{(\mathrm{m})}$ from $\boldsymbol{\mathcal{T}}$ and related standard SVD: $\mathbf{T}_{(\mathrm{m})} = \mathbf{U}^{(\mathrm{m})} \mathbf{S}^{(\mathrm{m})} \mathbf{V}^{(\mathrm{m})}$. The orthogonal matrix $\mathbf{U}^{(\mathrm{m})}$ contains the leading left singular vectors of $\mathbf{T}_{(\mathrm{m})}$.

2. Compute the core tensor $\boldsymbol{\mathcal{X}}$ using the inversion formula: $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{T}} \times_1 {\mathbf{U}^{(1)}}^T \times_2 {\mathbf{U}^{(2)}}^T \times_3 {\mathbf{U}^{(3)}}^T$.

Note that, since T-HOSVD is computed by means of 3 standard matrix T-SVDs, its computational cost can be reduced by using fast and efficient SVD algorithms. Moreover, the ability of T-HOSVD in effectively capturing the variation in each of the modes independently from the other ones, is particularly important to alleviate the problem of concentration of distances, thus making T-HOSVD well-suited to clustering purposes. In this work, in order to obtain a final clustering solution of the documents, we will consider the mode-1 factor matrix $\mathbf{U}^{(1)}$ of the T-HOSVD.

3.1.5 Document clustering

The mode-1 factor matrix is provided in input to a clustering method to obtain a final organization of the documents into K clusters, i.e., $\mathcal{C} = \{C_i\}_{i=1}^{K}$. Note that there is no principled relation between the number K of final document clusters and k. However, K is expected to reflect the number of topics of interest for the document collection. Also, possibly but not necessarily, the same clustering algorithm used for the segment clustering step (i.e., Bisecting K-Means) can be employed for this step.

3.2 Bag-of-synset representation

In the BoS model, our objective is to represent the document segments in a conceptual feature space instead of the traditional term space. Since we deal with multilingual documents, this task clearly relies on the multilingual lexical knowledge base functionalities of BabelNet. Conceptual features will hence correspond to BabelNet synsets.

The segment collection $\mathcal{S}$ is subject to a two-step processing phase. In the first step, each segment is broken down into a set of lemmatized and POS-tagged sentences, in which each word is replaced with related lemma and associated POS-tag. Let us denote with $\langle w, POS(w) \rangle$ a lemma and associated POS-tag occurring in any sentence sen of the segment. In the second step, a WSD method is applied to each pair $\langle w, POS(w) \rangle$ to detect the most appropriate BabelNet synset σ_w for $\langle w, POS(w) \rangle$ contextually to sen. The WSD algorithm is carried out in such a way that all words from all languages are disambiguated over the same concept inventory, producing a language-independent feature space for the whole multilingual corpus. Each segment is finally modeled as a $|\mathcal{BS}|$-dimensional vector of BabelNet synset frequencies, being $\mathcal{BS}$ the set of retrieved BabelNet synsets.

As previously discussed in Section 2.1, BabelNet provides WSD algorithms for multilingual corpora. More specifically, the authors in (Navigli and Ponzetto, 2012b) suggest to use the Degree algorithm (Navigli and Lapata, 2010), as it showed to yield highly competitive performance in a multilingual context as well. Note that the Degree algorithm, given a semantic graph for the input context, simply selects the sense of the target word with the highest vertex degree. Clearly, other graph-based methods for (unsupervised) WSD, particularly PageRank-style methods (e.g., (Mihalcea et al., 2004; Agirre and Soroa, 2009; Yeh et al., 2009; Tsatsaronis et al., 2010)), can be plugged in to address the multilingual WSD task based on BabelNet. An investigation of the performance of existing WSD algorithms for a multilingual context is however out of the scope of this paper.

4 Evaluation Methodology

In order to evaluate our proposal we need a multilingual comparable document collection with annotated

RCV2 Topics	*English*	*French*	*Italian*
Balanced Corpus			
C15 - PERFORMANCE	850	850	850
C18 - OWNERSHIP CHANGES	850	850	850
E11 - ECONOMIC PERFORMANCE	850	850	850
E12 - MONETARY/ECONOMIC	850	850	850
M11 - EQUITY MARKETS	850	850	850
M13 - MONEY MARKETS	850	850	850
Total	5 100	5 100	5 100
Unbalanced Corpus			
C15 - PERFORMANCE	850	850	0
C18 - OWNERSHIP CHANGES	850	850	0
E11 - ECONOMIC PERFORMANCE	0	850	850
E12 - MONETARY/ECONOMIC	850	0	850
M11 - EQUITY MARKETS	0	850	850
M13 - MONEY MARKETS	850	0	850
Total	3 400	3 400	3 400

Table 1: Number of documents for each topic and language.

Statistics	*Balanced Corpus*	*Unbalanced Corpus*
# of docs	15 300	10 200
# of terms	58 825	44 535
# of synsets	16 395	14 339
BoW Density	1.5E-3	2.0E-3
BoS Density	2.6E-3	3.1E-3

Table 2: Main characteristics of the corpora.

topics. For this reason, we used *Reuters Corpus Volume 2* (RCV2), a multilingual corpus containing news articles in thirteen language.[1] In the following, we present the corpus characteristics and competing methods used in our analysis.

4.1 Data preparation

We consider a subset of the RCV2 corpus corresponding to three languages: *English*, *French* and *Italian*. It covers six different topics, i.e., different labels of the RCV2 TOPICS field. Topics are chosen according with their coverage in the different languages. The language-specific documents were lemmatized and POS-tagged through the Freeling library (Padró and Stanilovsky, 2012) in order to obtain a suitable representation for the WSD process.

To assess the robustness of our proposal, we design two different scenarios. The first (*Balanced Corpus*) is characterized by a completely balanced dataset. Each language covers all topics and for each pair language/topic the same number of documents is selected. The second scenario corresponds to an *Unbalanced Corpus*. Starting from the balanced corpus, we removed for each topic all the documents belonging to one language. In this way, we obtained a corpus in which each topic is covered by only two of the three languages.

Main characteristics of both evaluation corpora are reported in Table 1 and Table 2. In the latter table, we report the number of documents, number of terms, number of synsets and the dataset density for both representations. To quantify the density of each corpus/representation combination, we counted the non-zero entries of the induced document-synset matrix (alternatively, document-term matrix) and we divided this value by the size of such matrix. This number provides an estimate about the density/sparseness of each dataset. Lower values indicate more sparse data. We can note that BoS model yields more dense datasets for both *Balanced Corpus* and *Unbalanced Corpus*.

[1] http://trec.nist.gov/data/reuters/reuters.html

RCV2 Topics	*English*	*French*	*Italian*
C15 - PERFORMANCE	3.41	3.67	3.27
C18 - OWNERSHIP CHANGES	3.20	3.32	2.40
E11 - ECONOMIC PERFORMANCE	4.89	3.17	2.07
E12 - MONETARY/ECONOMIC	5.22	3.69	2.05
M11 - EQUITY MARKETS	4.29	2.94	2.15
M13 - MONEY MARKETS	3.31	3.12	2.10

Table 3: Average number of document segments, for each topic and language.

	English		*French*		*Italian*	
RCV2 Topics	avg *BoS* seg. leng.	avg *BoW* seg. leng.	avg *BoS* seg. leng.	avg *BoW* seg. leng.	avg *BoS* seg. leng.	avg *BoW* seg. leng.
C15	21.76	36.32	11.54	34.92	10.58	37.75
C18	20.94	36.87	10.94	35.62	11.24	41.20
E11	22.90	37.24	11.47	34.73	11.96	38.60
E12	22.70	37.70	11.50	37.44	12.59	43.63
M11	22.04	36.83	10.91	32.76	11.57	42.39
M13	22.22	36.97	11.34	34.75	11.72	39.36

Table 4: Average length of document segment in the BoW and BoS spaces, for each topic and language.

As our proposal explicitly models document segments, we also report statistics, considering both topics and languages, related to the average number of segments per document (Table 3), and the average length of segments per document (Table 4). The latter statistic is computed separately for BoW and BoS representations. We made this distinction because a term cannot have a mapping to a synset, or it can be mapped to more than one synset in the BoS space during the WSD process (Section 3.2).

Looking at the average number of segments per document in Table 3, it can be noted that English documents contain, for all topics, a larger number of segments. This means that English documents are generally richer than the ones in the other languages. Italian language corresponds to the smallest documents, each of them containing between 2 and 3.2 segments on average. A sharper difference appears in the *MONETARY/ECONOMIC* topic for which English documents contain 5.2 segments, while the Italian ones are composed, on average, by only 2 segments.

Table 4 shows the average length of segments per document for both space representations. Generally, segments in the BoS representation are smaller than the corresponding segments in the BoW space. More in detail, if we consider the ratio between the segment length in BoS and the one in BoW, this ratio is around 2/3 for the English language, while for both French and Italian it varies between 1/4 and 1/3. This disequilibrium is induced by the multilingual concept coverage of BabelNet, as stated by its authors (Navigli and Ponzetto,

2012a), (Navigli and Ponzetto, 2012b). In particular, the WSD process tightly depends from the concept coverage supplied from the language-specific knowledge base.

4.2 Competing methods and settings

We compare our *SeMDocT* with two standard approaches, namely *Bisecting K-Means* (Steinbach et al., 2000), and *Latent Semantic Analysis (LSA)*-based document clustering (for short, *LSA*). Given a number K of desired clusters, *Bisecting K-Means* produces a K-way clustering solution by performing a sequence of K-1 repeated bisections based on standard K-Means algorithm. This process continues until the number K of clusters is found. *LSA* performs a decomposition of the document collection matrix through Singular Value Decomposition in order to extract a more concise and descriptive representation of the documents. After this step, *Bisecting K-Means* is applied over the new document space to get the final document clustering.

All the three methods, *SeMDocT*, *Bisecting K-Means* and *LSA* are coupled with either BoS or BoW representation models. The comparison between BoS and BoW representations allows us to evaluate the presumed benefits that can be derived by exploiting synsets instead of terms for the multilingual document clustering task.

Both *SeMDocT* and *LSA* require the number of components as input; as concerns specifically *SeMDocT*, we varied r_1 (cf. Section 3.1.4) from 2 to 30, with increments of 2. To determine the number of segment clusters k, we employed an automatic way as discussed in Section 3.1.2. By varying k from 2 to 40, for *Balanced Corpus* and *Unbalanced Corpus*, respectively, the values of k obtained were 22 and 23 under BoS, and 25 and 11 under BoW.

As concerns the step of text segmentation, TextTiling requires the setting of some interdependent parameters, particularly the size of the text unit to be compared and the number of words in a token sequence. We used the setting suggested in (Hearst, 1997) and also confirmed in (Tagarelli and Karypis, 2013), i.e., 10 for the text unit size and 20 for the token-sequence size.

4.3 Assessment criteria

Performance of the different methods are evaluated using two standard clustering validation criteria, namely *F-Measure* and *Rand Index*.

Given a document collection $\mathcal{D}$, let $\mathbf{\Gamma} = \{\Gamma_j\}_{j=1}^{H}$ and $\mathcal{C} = \{C_i\}_{i=1}^{K}$ denote a reference classification and a clustering solution for $\mathcal{D}$, respectively. The local precision and the local recall of a cluster C_i w.r.t. a class Γ_j are defined as $P_{ij} = |C_i \cap \Gamma_j|/|C_i|$ and $R_{ij} = |C_i \cap \Gamma_j|/|\Gamma_j|$, respectively. F-Measure (FM) is computed as follows (Steinbach et al., 2000):

$$F = \sum_{j=1}^{H} \frac{|\Gamma_j|}{|\mathcal{D}|} \max_{i=1\ldots K}\{F_{ij}\}$$

where $F_{ij} = 2P_{ij}R_{ij}/(P_{ij} + R_{ij})$.

Rand Index (RI) (Rand, 1971) measures the percentage of decisions that are correct, penalizing false positive and false negative decisions during clustering. It takes into account the following quantities: *TP*, i.e., the number of pairs of documents that are in the same cluster in $\mathcal{C}$ and in the same class in $\mathbf{\Gamma}$; *TN*, i.e., the number of pairs of documents that are in different clusters in $\mathcal{C}$ and in different classes in $\mathbf{\Gamma}$; *FN*, i.e., the number of pairs of documents that are in different clusters in $\mathcal{C}$ and in the same class in $\mathbf{\Gamma}$; and *FP*, i.e., the number of pairs of documents that are in the same cluster in $\mathcal{C}$ and in different classes in $\mathbf{\Gamma}$. Rand Index is hence defined as:

$$RI = \frac{TP + TN}{TP + TN + FP + FN}$$

Note that for each method, results were averaged over 30 runs and the number of final document clusters K was set equal to the number of topics in the document collection (i.e., 6).

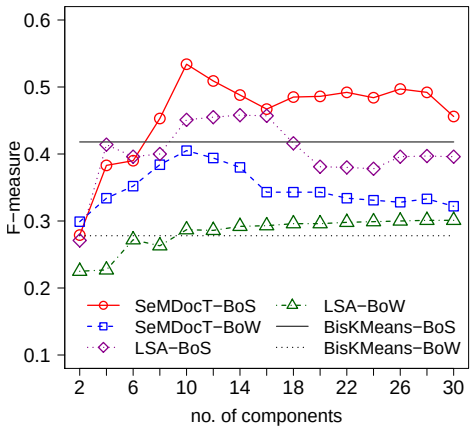
5 Results

We present here our main experimental results. We first provide a comparative evaluation of our *SeMDocT* with the competing methods, on both balanced and unbalanced corpus evaluation cases. Then we provide a per language analysis focusing on *SeMDocT*.

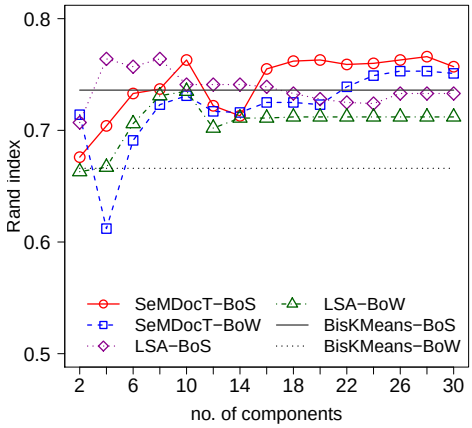
5.1 Evaluation with competing methods

Evaluation on balanced corpus. Figure 2 shows FM and RI results obtained by the various methods coupled with the two document representations on the *Balanced Corpus*. Several remarks stand out. First, the BoS space positively influences the performance of all the employed approaches. This is particularly evident for *Bisecting K-Means* and *LSA* that clearly benefit from this kind of representation. The former almost doubles its performance in terms of FM and significantly improves its result w.r.t. RI. *LSA* shows improvements in both cases. *SeMDocT*-BoS generally outperforms all the competitors for both FM and RI when the number of components is greater than 16. Note that, under the BoW model, *SeMDocT*-BoW still outperforms the other methods.

Evaluation on unbalanced corpus. Figure 3 reports results for the *Unbalanced Corpus*. Also in this evaluation, the best performances for all the methods are reached using the BoS representation. *SeMDocT*-BoS shows similar behavior according to the two measures. It always outperforms the competitors considering a number of components greater than or equal to 12. More precisely, *SeMDocT*-BoW obtains a gain of 0.047 and 0.103 in terms of FM and 0.006 and 0.058 in terms of RI, w.r.t. *LSA*-BoW and *Bisecting K-Means*-BoW, respectively. Similarly, *SeMDocT*-BoS obtains improvements of 0.05 in terms of FM w.r.t. both BoS

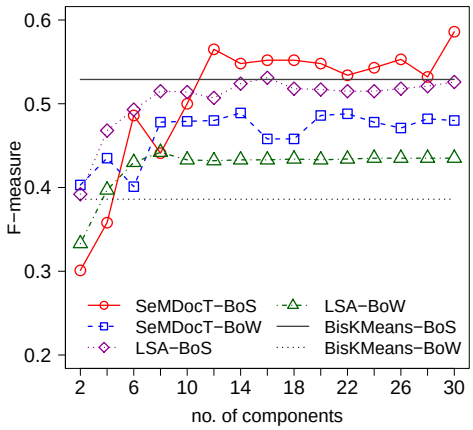


(a)

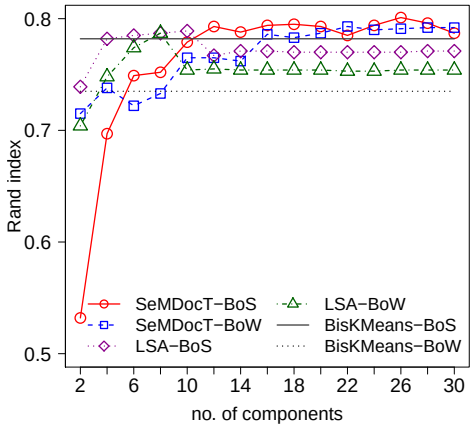


(b)

Figure 2: Average F-Measure (a) and Rand Index (b) on the *Balanced Corpus* using BoW and BoS document representation and varying the number of components for both *SeMDocT* and *LSA*.



(a)



(b)

Figure 3: Average F-Measure (a) and Rand Index (b) on the *Unbalanced Corpus* using BoW and BoS document representation and varying the number of components for both *SeMDocT* and *LSA*.

competitors, while in terms of RI the differences in performance are 0.012 and 0.019 for *LSA*-BoS and *Bisecting K-Means*-BoS, respectively.

5.2 Per language evaluation of *SeMDocT*-BoS

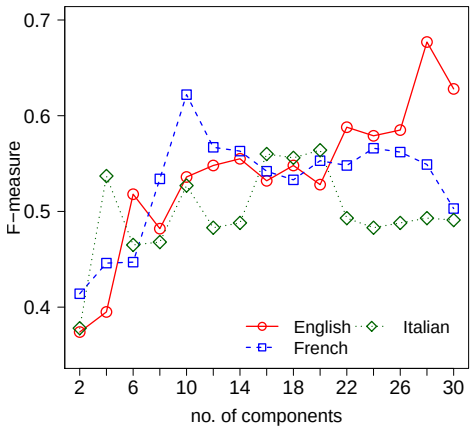
Starting from the clustering solutions produced by *SeMDocT*-BoS in both balanced and unbalanced cases, for each language we extracted a language-specific projection of the clustering. After that, we computed the clustering validation criteria according to language specific solutions to quantify how well the clustering result fits each specific language. The results of this experiment are reported in Fig. 4 and Fig. 5.

On the *Balanced Corpus*, *SeMDocT*-BoS shows comparable performance for English and French documents, while it behaves slightly worse for Italian texts. This trend is highlighted for both clustering evaluation criteria. Inspecting the results for the *Unbalanced Corpus*, we observe a different trend. Results obtained for the English texts are generally better than the results for the French and Italian documents. For this benchmark, *SeMDocT*-BoS obtains similar results for documents written in French and in Italian.

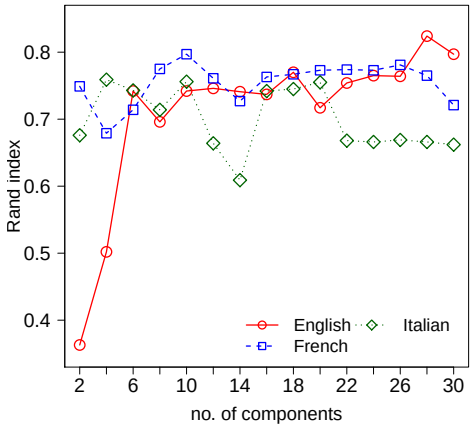
Dataset	*Language*	*BoW size*	*BoS size*	*avg # synsets per term* (β)
Balanced	English	29 999	12 065	0.4021
	French	17 826	5 310	0.2978
	Italian	16 951	4 471	0.2637
Unbalanced	English	19 432	10 387	0.5345
	French	14 439	4 431	0.3068
	Italian	14 743	4 012	0.2721

Table 5: Statistics by language.

We gained an insight into the above discussed performance behaviors by computing some additional statistics that we report in Table 5: for each language and each dataset, the size of the term and synset dictionaries and the average number of synsets per lemma (β) we retrieved through BabelNet according to the related corpus. More in detail, β is the ratio between the BoS and the BoW dictionaries. This quantity roughly evaluates how many synsets are produced per term during the multilingual WSD process (Section 3.2). As we can observe, this value is always smaller than one, which means that not all the terms have a corresponding mapping to a synset. The β ratio can explain the discrep-

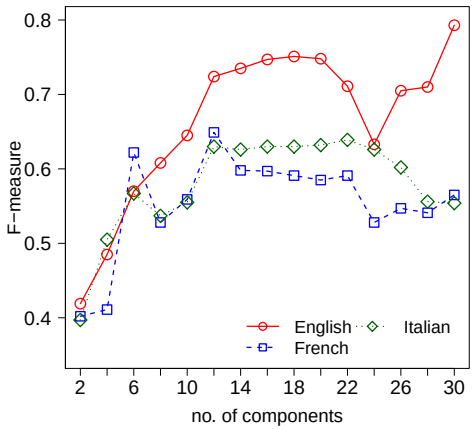


(a)

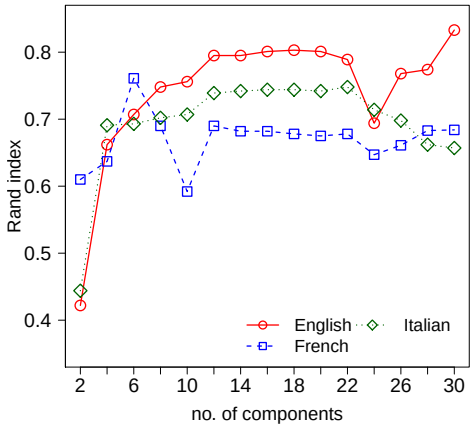


(b)

Figure 4: Average F-Measure (a) and Rand Index (b) for language specific solutions on the *Balanced Corpus* obtained by *SeMDocT*-BoS.



(a)



(b)

Figure 5: Average F-Measure (a) and Rand Index (b) for language specific solutions on the *Unbalanced Corpus* obtained by *SeMDocT*-BoS.

ancy in (language-specific) performances in the two scenarios. In particular, the difference in the β statistic between English and the other languages is more evident for the *Unbalanced Corpus* (i.e., 0.23 between English and French), while it is lower for the *Balanced Corpus* (around 0.1). The relatively large gap in β between the first and the second language (respectively, English and French) for the *Unbalanced Corpus* reduces the relative gap between the second and the third languages (respectively, French and Italian) while this trend is less marked for the *Balanced Corpus* as β range is narrower. In summary, we can state that our framework works well if BabelNet knowledge base provides a good coverage of the terms in the analyzed language. Experimental evidence shows that, if this condition is met, *SeMDocT*-BoS provides better clustering results w.r.t. the competing approaches.

5.3 Runtime of tensor decomposition

As previously discussed, T-HOSVD of a third-order tensor can be computed through three standard SVDs. Furthermore, for clustering purposes, we considered only the mode-1 factor matrix of the decomposition. To compute the SVD, we used the *svds()* function of MATLAB R2012b, which is based on an iterative algorithm.[2] Experiments were carried out on an Intel Core I7-3610QM platform with 16GB DDR RAM.

Figure 6 shows the execution time of the SVD over the mode-1 matricization of our tensor for the *Balanced Corpus*, by varying the number of components, for both BoW and BoS representation models. As it can be observed, in both cases the runtime is linear in the number of components. However, the SVD computation in the BoS setting is one order of magnitude faster than time performance in the BoW setting. This is mainly due to a large difference in size between the feature spaces of BoW and BoS (cf. Table 2), since the selected number of segment clusters (k) was nearly the same (25 for BoW, and 22 for BoS). Therefore, by providing a more compact feature space, BoS clearly allows for a much less expensive SVD computation for our tensor decomposition.

[2] http://www.mathworks.it/it/help/matlab/ref/svds.html

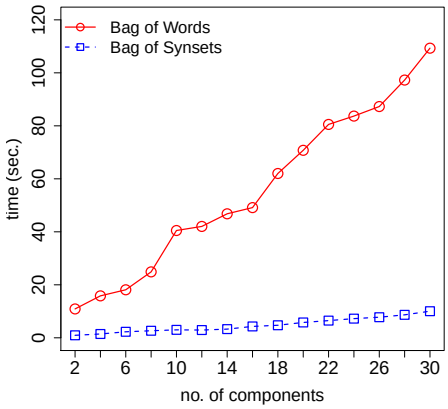


Figure 6: Time performance of SVD over the mode-1 matricization of the *Balanced Corpus* tensor.

6 Discussion

Our work paves the way for the use of a multilingual knowledge base to deal with the multilingual document clustering task. Here we sum up our main findings.

***SeMDocT* vs. *LSA*.** *LSA* achieved its best results for a number of components generally smaller than the one for which *SeMDocT* obtained its maximum. This is due to the initial information that the two methods summarize. *LSA* tries to capture the variation of the initial document-term (alternatively, document-synset) matrix representing the texts in a lower space, whereas *SeMDocT* does the same starting from a richer representation of the documents (i.e., a third-order tensor model). For this reason, *SeMDocT* tends to employ relatively more components in order to summarize the documents content; however, a number of components between 16 and 30 is generally enough to ensure good performance of *SeMDocT*. Moreover, in most cases, the highest performance results by *SeMDocT* are better than the highest performances of *LSA*.

BoS vs. BoW. Our results have highlighted the better quality in multilingual clustering supplied by synsets compared with the one provided by terms. BoS produces a smaller representation space over which documents are projected, but it is enough rich to well capture the documents content. In particular, BoS benefits from the WSD process that is able to discriminate the same term w.r.t. the context in which it appears.

BabelNet. BabelNet is a recent project that supports many different languages. As the intention of the authors is to enrich this resource, in the future our framework will benefit of this fact. Moreover, our framework can deal with documents written in many different languages as they are represented through the same space; the only constraint is related to the available language support in BabelNet. On the other hand, we point out that any other multilingual knowledge base and WSD tools can in principle be integrated in our framework.

7 Conclusion

In this paper we proposed a new approach for multilingual document clustering. Our key idea lies in the combination of a tensor-based model with a bag-of-synsets description, which enables a common space to project multilingual document collections. We evaluated our approach w.r.t. standard document clustering methods, using both term and synset representations. Results have shown the benefits deriving from the use of a multilingual knowledge base in the analysis of comparable corpora, and also shown the significance of our approach in both a balanced and an unbalanced corpus evaluation. Our tensor-based representation of topically-segmented multilingual documents can also be applied to cross-lingual information retrieval or multilingual document categorization.

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for Word Sense Disambiguation. In *Proc. of the International Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 33–41.

Doug Beeferman, Adam L. Berger, and John D. Lafferty. 1999. Statistical Models for Text Segmentation. *Machine Learning*, 34(1-3):177–210.

Peter A. Chew, Brett W. Bader, Tamara G. Kolda, and Ahmed Abdelali. 2007. Cross-language information retrieval using PARAFAC2. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 143–152.

Freddy Y. Y. Choi, Peter Wiemer-Hastings, and Johanna Moore. 2001. Latent Semantic Analysis for Text Segmentation. In *Proc. of the International Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 109–117.

Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. 2001. Latent Semantic Kernels. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 66–73.

Ian Davidson, Kiri Wagstaff, and Sugato Basu. 2006. Measuring constraint-set utility for partitional clustering algorithms. In *Proc. of the European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 115–126.

Dino Ienco, Céline Robardet, Ruggero G. Pensa, and Rosa Meo. 2013. Parameter-less co-clustering for star-structured heterogeneous data. *Data Mining and Knowledge Discovery*, 26(2):217–254.

Marti A. Hearst. 1997. TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages. *Computational Linguistics*, 23(1):33–64.

Young-Min Kim, Massih-Reza Amini, Cyril Goutte, and Patrick Gallinari. 2010. Multi-view clustering

of multilingual documents. In *Proc. of the ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 821–822.

Tamara G. Kolda and Brett W. Bader. 2009. Tensor Decompositions and Applications. *SIAM Review*, 51(3):455–500.

N. Kiran Kumar, G. S. K. Santosh, and Vasudeva Varma. 2011. A language-independent approach to identify the named entities in under-resourced languages and clustering multilingual documents. In *Proc. of the International Conference of the Cross-Language Evaluation Forum (CLEF)*, pages 74–82.

N. Kiran Kumar, G. S. K. Santosh, and Vasudeva Varma. 2011. Effectively mining Wikipedia for clustering multilingual documents. In *Proc. of the International Conference on Applications of Natural Language to Information Systems (NLDB)*, pages 254–257.

N. Kiran Kumar, G. S. K. Santosh, and Vasudeva Varma. 2011. Multilingual document clustering using Wikipedia as external knowledge. In *Proc. of the Information Retrieval Facility Conference (IRFC)*, pages 108–117.

Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. 2000. A Multilinear Singular Value Decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278.

Xinhai Liu, Wolfgang Glänzel, and Bart De Moor. 2011. Hybrid clustering of multi-view data via Tucker-2 model and its application. *Scientometrics*, 88(3):819–839.

Rada Mihalcea, Paul Tarau, and Elizabeth Figa. 2004. PageRank on Semantic Networks, with Application to Word Sense Disambiguation. In *Proc. of the International Conference on Computational Linguistics (COLING)*.

Soto Montalvo, Raquel Martínez-Unanue, Arantza Casillas, and Víctor Fresno. 2007. Multilingual news clustering: Feature translation vs. identification of cognate named entities. *Pattern Recognition Letters*, 28(16):2305–2311.

Roberto Navigli and Mirella Lapata. 2010. An Experimental Study of Graph Connectivity for Unsupervised Word Sense Disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):678–692.

Roberto Navigli and Simone P. Ponzetto. 2012. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. *Artificial Intelligence*, 193:217-250.

Roberto Navigli and Simone P. Ponzetto. 2012. Multilingual WSD with Just a Few Lines of Code: The BabelNet API. In *Proc. of the System Demonstrations of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 67–72.

Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. 2011. Cross lingual text classification by mining multilingual topics from Wikipedia. In *Proc. of the ACM International Conference on Web Search and Web Data Mining (WSDM)*, pages 375–384.

Lluís Padró and Evgeny Stanilovsky. 2012. FreeLing 3.0: Towards Wider Multilinguality. In *Proc. of the Language Resources and Evaluation Conference (LREC)*.

William M. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850.

Salvatore Romeo, Andrea Tagarelli, Francesco Gullo, and Sergio Greco. 2013. A Tensor-based Clustering Approach for Multiple Document Classifications. In *Proc. of the International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, pages 200–205.

Stan Salvador and Philip Chan. 2004. Determining the Number of Clusters/Segments in Hierarchical Clustering/Segmentation Algorithms. In *Proc. of the International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 576–584.

Michael Steinbach, George Karypis, and Vipin Kumar. 2000. A Comparison of Document Clustering Techniques. In *Proc. of the KDD Workshop on Text Mining*.

Andrea Tagarelli and George Karypis. 2013. A segment-based approach to clustering multi-topic documents. *Knowledge and Information Systems*, 34(3):563–595.

George Tsatsaronis, Iraklis Varlamis, and Kjetil Nørvåg. 2010. SemanticRank: Ranking Keywords and Sentences Using Semantic Graphs. In *Proc. of the International Conference on Computational Linguistics (COLING)*, pages 1074–1082.

Chih-Ping Wei, Christopher C. Yang, and Chia-Min Lin. 2008. A Latent Semantic Indexing-based Approach to Multilingual Document Clustering. *Decision Support Systems*, 45(3):606–620.

Eric Yeh, Daniel Ramage, Christopher D. Manning, Eneko Agirre, and Aitor Soroa. 2009. WikiWalk: Random walks on Wikipedia for Semantic Relatedness. In *Proc. of the ACL Workshop on Graph-based Methods for Natural Language Processing*, pages 41–49.

Dani Yogatama and Kumiko Tanaka-Ishii. 2009. Multilingual spectral clustering using document similarity propagation. In *Proc. of the International Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 871–879.

Ying Zhao and George Karypis. 2004. Empirical and Theoretical Comparison of Selected Criterion Functions for Document Clustering. *Machine Learning*, 55(3):311–331.

Lexical Substitution for the Medical Domain

Martin Riedl[1] **Michael R. Glass**[2] **Alfio Gliozzo**[2]
(1) FG Language Technology, CS Dept., TU Darmstadt, 64289 Darmstadt, Germany
(2) IBM T.J. Watson Research, Yorktown Heights, NY 10598, USA
riedl@cs.tu-darmstadt.de, {mrglass,gliozzo}@us.ibm.com

Abstract

In this paper we examine the lexical substitution task for the medical domain. We adapt the current best system from the open domain, which trains a single classifier for all instances using delexicalized features. We show significant improvements over a strong baseline coming from a distributional thesaurus (DT). Whereas in the open domain system, features derived from WordNet show only slight improvements, we show that its counterpart for the medical domain (UMLS) shows a significant additional benefit when used for feature generation.

1 Introduction

The task of lexical substitution (McCarthy and Navigli, 2009) deals with the substitution of a *target term* within a sentence with words having the same meaning. Thus, the task divides into two subtasks:

- Identification of *substitution candidates*, i.e. terms that are, for some contexts, substitutable for a given target term.
- Ranking the substitution candidates according to their context

Such a substitution system can help for semantic text similarity (Bär et al., 2012), textual entailment (Dagan et al., 2013) or plagiarism detection (Chong and Specia, 2011).

Datasets provided by McCarthy and Navigli (2009) and Biemann (2012) offer manually annotated substitutes for a given set of target words within a context (sentence). Contrary to these two datasets in Kremer et al. (2014) a dataset is offered where all words have are annotated with substitutes. All the datasets are suited for the open domain.

But a system performing lexical substitution is not only of interest for the open domain, but also for the medical domain. Such a system could then be applied to medical word sense disambiguation, entailment or question answering tasks. Here we introduce a new dataset and adapt the lexical substitution system, provided by Szarvas et al. (2013), to the medical domain. Additionally, we do not make use of WordNet (Miller, 1995) to provide similar terms, but rather employ a Distributional Thesaurus (DT), computed on medical texts.

2 Related Work

For the general domain, the lexical substitution task was initiated by a Semeval-2007 Task (McCarthy and Navigli, 2009). This task was won by an unsupervised method (Giuliano et al., 2007), which uses WordNet for the substitution candidate generation and then relies on the Google Web1T n-grams (Brants and Franz, 2006)[1] to rank the substitutes.

The currently best system, to our knowledge, is proposed by Szarvas et al. (2013). This is a supervised approach, where a single classifier is trained using delexicalized features for all substitutes and can thus be applied even to previously unseen substitutes. Although there have been many approaches for solving the task for the general domain, only slight effort has been done in adapting it to different domains.

3 Method

To perform lexical substitution, we follow the delexicalization framework of Szarvas et al. (2013). We automatically build Distributional Thesauri (DTs) for the medical domain and use features from the Unified Medical Language System (UMLS) ontology. The dataset for supervised lexical substitution consists of sentences, containing an annotated target word t. Considering the sentence being the context for the target word, the target word might have different meanings. Thus annotated substitute candidates $s_{g_1} \dots s_{g_n} \in s_g$, need to be provided for each context. The negative examples are substitute candidates that either are incorrect for the target word, do not fit into the context or both. We will refer to these substitutes as *false substitute candidates* $s_{f_1} \dots s_{f_m} \in s_f$ with $s_f \cap s_g = \emptyset$.

For the generation of substitute candidates we do not use WordNet, as done in previous works (Szarvas et al., 2013), but use only substitutes from a DT. To train a single classifier, features that distinguishing the meaning of words in different context need to be considered. Such features could be e.g. n-grams, features from distributional semantics or features which are extracted

[1] http://catalog.ldc.upenn.edu/LDC2006T13

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 610–614,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

relative to the target word, such as the ratio between frequencies of the substitute candidate and the target word. After training, we apply the algorithm to unseen substitute candidates and rank them according to their positive probabilities, given by the classifier. Contrary to Szarvas et al. (2013), we do not use any weighting in the training if a substitute has been supplied by many annotators, as we could not observe any improvements. Additionally, we use logistic regression (Fan et al., 2008) as classifier[2].

4 Resources

For the substitutes and for the generation of delexicalized features, we rely on DTs, the UMLS and Google Web1T.

4.1 Distributional thesauri (DTs)

We computed two different DTs using the framework proposed in Biemann and Riedl (2013)[3].

The first DT is computed based on Medline[4] abstracts. This thesaurus uses the left and the right word as context features. To include multi-word expressions, we allow the number of tokens that form a term to be up to the length of three.

The second DT is based on dependencies as context features from a English Slot Grammar (ESG) parser (McCord et al., 2012) modified to handle medical data. The ESG parser is also capable of finding multi-word expressions. As input data we use 3.3 GB of texts from medical textbooks, encyclopedias and clinical reference material as well as selected journals. This DT is also used for the generation of candidates supplied to annotators when creating the gold standard and therefore is the main resource to provide substitute candidates.

4.2 UMLS

The Unified Medical Language System (UMLS) is an ontology for the medical domain. In contrast to Szarvas et al. (2013), which uses WordNet (Miller, 1995) to generate substitute candidates and also for generating features, we use UMLS solely for feature generation.

4.3 Google Web1T

We use the Google Web1T to generate n-gram features as we expect this open domain resource to have considerable coverage for most specific domains as well. For accessing the resource, we use JWeb1T[5] (Giuliano et al., 2007).

[2]We use a Java port of LIBLINEAR (`http://www.csie.ntu.edu.tw/~cjlin/liblinear/`) available from `http://liblinear.bwaldvogel.de/`

[3]We use Lexicographer's Mutual Information (LMI) (Evert, 2005) as significance measure and consider only the top 1000 ($p = 1000$) features per term.

[4]`http://www.nlm.nih.gov/bsd/licensee/2014_stats/baseline_med_filecount.html`

[5]https://code.google.com/p/jweb1t/

5 Lexical Substitution dataset

Besides the lexical substitution data sets for the open domain (McCarthy and Navigli, 2009; Biemann, 2012; Kremer et al., 2014) there is no dataset available that can be used for the medical domain. Therefore, we constructed an annotation task for the medical domain using a medical corpus and domain experts.

In order to provide the annotators with a clear task, we presented a question, and a passage that contains the correct answer to the question. We restricted this to a subset of passages that were previously annotated as justifying the answer to the question. This is related to a textual entailment task, essentially the passage entails the question with the answer substituted for the focus of the question. We instructed the annotators to first identify the terms that were relevant for the entailment relation. For each relevant term we randomly extracted 10 terms from the ESG-based DT within the top 100 most similar terms. Using this list of distributionally similar terms, the annotators selected those terms that would preserve the entailment relation if substituted. This resulted in a dataset of 699 target terms with substitutes. On average from the 10 terms 0.846 are annotated as correct substitutes. Thus, the remaining terms can be used as false substitute candidates.

The agreement on this task by Fleiss Kappa was 0.551 indicating "moderate agreement" (Landis and Koch, 1977). On the metric of pairwise agreement, as defined in the SemEval lexical substitution task, we achieve 0.627. This number is not directly comparable to the pairwise agreement score of 0.277 for the SemEval lexical substitution task (McCarthy and Navigli, 2009) since in our task the candidates are given. However, it shows promise that subjectivity may be reduced by casting lexical substitution into a task of maintaining entailment.

6 Evaluation

For the evaluation we use a ten-fold cross validation and report P@1 (also called Average Precision (AP) at 1) and Mean Average Precision (MAP) (Buckley and Voorhees, 2004) scores. The P@1 score indicates how often the first substitute of the system matches the gold standard. The MAP score is the mean of all AP from 1 to the number of all substitutes.

- Google Web 1T:
  We use the same Google n-gram features, as used in Giuliano et al. (2007) and Szarvas et al. (2013). These are frequencies of n-grams formed by the substitute candidate s_i and the left and right words, taken from the context sentence, normalized by the frequency of the same context n-gram with the target term t. Additionally, we add the same features, normalized by the frequency sum of all n-grams of the substitute candidates. Another feature is generated using the frequencies where t and s are listed together using the words

and, *or* and *","* as separator and also add the left and right words of that phrase as context. Then we normalize this frequency by the frequency of the context occurring only with t.

- DT features:
  To characterize if t and s_i have similar words in common, and therefore are similar, we compute the percentage of words their thesauri entries share, considering the top n words in each entry with $n = 1, 5, 20, 50, 100, 200$. During the DT calculation we also calculate the significances between each word and its context features (see Section 4.1). Using this information, we compute if the words in the sentences also occur as context features for the substitute candidate. A third feature group relying on DTs is created by the overlapping context features for the top m entries of t and s_i with $m = 1, 5, 20, 50, 100, 1000$, which are ranked regarding their significance score. Whereas, the similarities between the trigram-based and the ESG-based DT are similar, the context features are different. Both feature types can be applied to the two DTs. Additionally, we extract the thesaurus entry for the target word t and generate a feature indicating whether the substitute s_i is within the top k entries with $k = 1, 5, 10, 20, 100$ entries[6].

- Part-of-speech n-grams:
  To identify the context of the word we use the POS-tag (only the first letter) of s_i and t as feature and POS-tag combinations of up to three neighboring words.

- UMLS:
  Considering UMLS we look up all concept unique identifiers (CUIs) for s_i and t. The first two features are the number of CUIs for s_i and t. The next features compute the number of CUIs that s_i and t share, starting from the minimal to the maximum number of CUIs. Additionally, we use a feature indicating that s_i and t do not share any CUI.

6.1 Substitute candidates

The candidates for the substitution are taken from the ESG based DT. For each target term we use the gold substitute candidates as correct instances and add all possible substitutes for the same target term occurring in a different context and do not have been annotated as valid in the present context as false instances.

7 Results

Running the experiment, we get the results as shown in Table 1. As baseline system we use the ranking of the ESG-based DT. As can be seen, the baseline is already quite high, which can be attributed to the fact that this resource was used to generate substitutes und thus contains all positive instances. Using the supervised approach, we can beat the baseline by 0.10 for the MAP score and by 0.176 for the P@1 score, which is a significant improvement ($p < 0.0001$, using a two tailed permutation test). To get insights of the contribution of individual feature types, we perform an ablation test. We observe that the most prominent features are coming from the two DTs as we only achieve results below the baseline, when removing DT features. We still obtain significant improvements over the baseline when removing other feature groups. The second most important feature comes from the UMLS. Features coming from the Google n-grams improve the system only slightly. The lowest improvement is derived from the part-of-speech features. This leads us to summarize that a hybrid approach for feature generation using manually created resources (UMLS) and unsupervised features (DTs) leads to the best result for lexical substitution for the medical domain.

System	MAP	P@1
Baseline	0.6408	0.5365
ALL	0.7048	0.6366
w/o DT	0.5798	0.4835
w/o UMLS	0.6618	0.5651
w/o Ngrams	0.7009	0.6252
w/o POS	0.7027	0.6323

Table 1: Results for the evaluation using substitute candidates from the DT.

8 Analysis

For a better insight into the lexical substitution we analyzed how often we outperform the baseline, get equal results or get decreased scores. According to Table 2 in around 26% of the cases we observe a decreased MAP score, which is on average 0.16 smaller then the scores achieved with the baseline. On the other hand, we see improvements in around 39% of the cases: an average improvements of 0.26, which is much higher then the loss. For the remaining 25% of cases we observe the same score.

performance	# of instances	Avg. Δ MAP
decline	180	-0.16
equal	244	0
improvements	275	0.26

Table 2: Error analysis for the task respectively to the MAP score.

Looking inside the data, the largest error class is caused by antonyms. A sub-class of this error are multi-word expressions having an adjective modifier. This problems might be solved by additional features using the UMLS resource. An example is shown in Figure 1.

[6]Whereas in Szarvas et al. (2013) only $k = 100$ is used, we gained an improvement in performance when also adding smaller values of k.

Sentence: he most common cause of thrombocytopenia during pregnancy is gestational thrombocytopenia, which is a **mild thrombocytopenia** with platelet levels remaining greater than 70,000/mL.

Gold: decreased platelet=1

Baseline: decreased platelet:17.0, severe thrombocytopenia: 16.0, macrothrombocytopenia:16.0, prolonged aptt:16.0, normal platelet count:16.0, hepatosplenomegaly:15.0, hypoxaemia:13.0, short finger:12.0, lymphadenopathy:11.0, mild symptom:11.0

System: severe thrombocytopenia:0.272, normal platelet count:0.204, macrothrombocytopenia:0.190, hepatosplenomegaly:0.174, prolonged aptt:0.168, decreased platelet:0.156, mild symptom:0.113, lymphadenopathy:0.085, hypoxaemia:0.067, short finger:0.053

Figure 1: Example sentence for the target term *mild thrombocytopenia*. The system returns a wrong ranking, as the adjective changes the meaning and turns the first ranked term into an antonym.

For feature generation, we currently lookup multi-word expressions as one term, both in the DT and the UMLS resource and do not split them into their single tokens. This error also suggests considering the single words inside the multi-word expression, especially adjectives, and looking them up in a resource (e.g. UMLS) to detect synonymy and antonymy.

Figure 2 shows the case, where the ranking is performed correctly, but the precise substitute is not annotated as a correct one. The term *nail plate* might be even more precise in the context as the manual annotated term *nail bed*. Due to the missing annotation the

Sentence: 7 Yellow nail syndrome is a rare disorder characterized by the triad of yellow and thickened **nails**, lymphedema and respiratory manifestation commonly pleural effusion and other complications like bronchiectasis and chronic sinusitis. Lymphedema in yellow nail syndrome is characteristically non-pitting and involves the lower extremities in symmetric fashion

Gold: finger nail=1, nail bed=1

Baseline: finger nail:58, nail bed:54.0, nail plate:49.0, scalp:47.0, hand:26.0, arms and legs:25.0, eye:23.0, cranial bone:19.0, palm:16.0, urinary organ:16.0

System: nail plate:0.676, finger nail:0.158, nail bed:0.144, scalp:0.106, hand:0.044, arms and legs:0.043, urinary organ:0.017, eye:0.016, palm:0.016, cranial bone:0.014

Figure 2: Example sentence for the target term *nails*. Here the ranking from the system is correct, but the first substitute from the system was not annotated as such.

baseline gets better scores then the result from the system.

9 Conclusion

In summary, we have examined the lexical substitution task for the medical domain and could show that a system for open domain text data can be applied to the medical domain. We can show that following a hybrid approach using features from UMLS and distributional semantics leads to the best results. In future work, we will work on integrating DTs using other context features, as we could see an impact of using two different DTs. Furthermore, we want to incorporate features using n-grams computed on a corpus from the domain and include co-occurrence features.

Acknowledgments

We thank Adam Lally, Eric Brown, Edward A. Epstein, Chris Biemann and Faisal Chowdhury for their helpful comments.

References

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: Computing Semantic Textual Similarity by Combining Multiple Content Similarity Measures. In *Proceedings of the 6th International Workshop on Semantic Evaluation, held in conjunction with the 1st Joint Conference on Lexical and Computational Semantics*, pages 435–440, Montreal, Canada.

Chris Biemann and Martin Riedl. 2013. Text: Now in 2D! A Framework for Lexical Expansion with Contextual Similarity. *Journal of Language Modelling*, 1(1):55–95.

Chris Biemann. 2012. Turk bootstrap word sense inventory 2.0: A large-scale resource for lexical substitution. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey.

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram corpus version 1. Technical report, Google Research.

Chris Buckley and Ellen M. Voorhees. 2004. Retrieval evaluation with incomplete information. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 25–32, Sheffield, United Kingdom.

Miranda Chong and Lucia Specia. 2011. Lexical generalisation for word-level matching in plagiarism detection. In *Recent Advances in Natural Language Processing*, pages 704–709, Hissar, Bulgaria.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio M. Zanzotto. 2013. Recognizing Textual Entailment: Models and Applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.

Stefan Evert. 2005. *The Statistics of Word Cooccurrences: Word Pairs and Collocations.* Ph.D. thesis, Institut für maschinelle Sprachverarbeitung, University of Stuttgart.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Claudio Giuliano, Alfio Gliozzo, and Carlo Strapparava. 2007. Fbk-irst: Lexical substitution task exploiting domain and syntagmatic coherence. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 145–148, Prague, Czech Republic.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What Substitutes Tell Us - Analysis of an "All-Words" Lexical Substitution Corpus. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*, pages 540–549, Gothenburg, Sweden.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33:159–174.

Diana McCarthy and Roberto Navigli. 2009. The English lexical substitution task. *Language Resources and Evaluation*, 43(2):139–159.

Michael C. McCord, J. William Murdock, and Branimir K. Boguraev. 2012. Deep Parsing in Watson. *IBM J. Res. Dev.*, 56(3):264–278.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38:39–41.

György Szarvas, Chris Biemann, and Iryna Gurevych. 2013. Supervised All-Words Lexical Substitution using Delexicalized Features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, pages 1131–1141, Atlanta, GA, USA.

Question Answering with Subgraph Embeddings

Antoine Bordes
Facebook AI Research
112 avenue de Wagram,
Paris, France
abordes@fb.com

Sumit Chopra
Facebook AI Research
770 Broadway,
New York, USA
spchopra@fb.com

Jason Weston
Facebook AI Research
770 Broadway,
New York, USA
jase@fb.com

Abstract

This paper presents a system which learns to answer questions on a broad range of topics from a knowledge base using few hand-crafted features. Our model learns low-dimensional embeddings of words and knowledge base constituents; these representations are used to score natural language questions against candidate answers. Training our system using pairs of questions and structured representations of their answers, and pairs of question paraphrases, yields competitive results on a recent benchmark of the literature.

1 Introduction

Teaching machines how to automatically answer questions asked in natural language on any topic or in any domain has always been a long standing goal in Artificial Intelligence. With the rise of large scale structured knowledge bases (KBs), this problem, known as open-domain question answering (or open QA), boils down to being able to query efficiently such databases with natural language. These KBs, such as FREEBASE (Bollacker et al., 2008) encompass huge ever growing amounts of information and ease open QA by organizing a great variety of answers in a structured format. However, the scale and the difficulty for machines to interpret natural language still makes this task a challenging problem.

The state-of-the-art techniques in open QA can be classified into two main classes, namely, information retrieval based and semantic parsing based. Information retrieval systems first retrieve a broad set of candidate answers by querying the search API of KBs with a transformation of the question into a valid query and then use fine-grained detection heuristics to identify the exact answer (Kolomiyets and Moens, 2011; Unger et al., 2012; Yao and Van Durme, 2014). On the other hand, semantic parsing methods focus on the correct interpretation of the meaning of a question by a semantic parsing system. A correct interpretation converts a question into the exact database query that returns the correct answer. Interestingly, recent works (Berant et al., 2013; Kwiatkowski et al., 2013; Berant and Liang, 2014; Fader et al., 2014) have shown that such systems can be efficiently trained under indirect and imperfect supervision and hence scale to large-scale regimes, while bypassing most of the annotation costs.

Yet, even if both kinds of system have shown the ability to handle large-scale KBs, they still require experts to hand-craft lexicons, grammars, and KB schema to be effective. This non-negligible human intervention might not be generic enough to conveniently scale up to new databases with other schema, broader vocabularies or languages other than English. In contrast, (Fader et al., 2013) proposed a framework for open QA requiring almost no human annotation. Despite being an interesting approach, this method is outperformed by other competing methods. (Bordes et al., 2014b) introduced an embedding model, which learns low-dimensional vector representations of words and symbols (such as KBs constituents) and can be trained with even less supervision than the system of (Fader et al., 2013) while being able to achieve better prediction performance. However, this approach is only compared with (Fader et al., 2013) which operates in a simplified setting and has not been applied in more realistic conditions nor evaluated against the best performing methods.

In this paper, we improve the model of (Bordes et al., 2014b) by providing the ability to answer more complicated questions. The main contributions of the paper are: (1) a more sophisticated inference procedure that is both efficient and can consider longer paths ((Bordes et al., 2014b) considered only answers directly connected to the

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 615–620,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

question in the graph); and (2) a richer representation of the answers which encodes the question-answer path and surrounding subgraph of the KB. Our approach is competitive with the current state-of-the-art on the recent benchmark WEBQUESTIONS (Berant et al., 2013) without using any lexicon, rules or additional system for part-of-speech tagging, syntactic or dependency parsing during training as most other systems do.

2 Task Definition

Our main motivation is to provide a system for open QA able to be trained as long as it has access to: (1) a training set of questions paired with answers and (2) a KB providing a structure among answers. We suppose that all potential answers are entities in the KB and that questions are sequences of words that include one identified KB entity. When this entity is not given, plain string matching is used to perform entity resolution. Smarter methods could be used but this is not our focus.

We use WEBQUESTIONS (Berant et al., 2013) as our evaluation bemchmark. Since it contains few training samples, it is impossible to learn on it alone, and this section describes the various data sources that were used for training. These are similar to those used in (Berant and Liang, 2014).

WebQuestions This dataset is built using FREEBASE as the KB and contains 5,810 question-answer pairs. It was created by crawling questions through the Google Suggest API, and then obtaining answers using Amazon Mechanical Turk. We used the original split (3,778 examples for training and 2,032 for testing), and isolated 1k questions from the training set for validation. WEBQUESTIONS is built on FREEBASE since all answers are defined as FREEBASE entities. In each question, we identified one FREEBASE entity using string matching between words of the question and entity names in FREEBASE. When the same string matches multiple entities, only the entity appearing in most triples, i.e. the most popular in FREEBASE, was kept. Example questions (answers) in the dataset include *"Where did Edgar Allan Poe died?"* (`baltimore`) or *"What degrees did Barack Obama get?"* (`bachelor_of_arts`, `juris_doctor`).

Freebase FREEBASE (Bollacker et al., 2008) is a huge and freely available database of general facts; data is organized as triplets (`subject`, `type1.type2.predicate`, `object`), where two entities `subject` and `object` (identified by `mids`) are connected by the relation type `type1.type2.predicate`. We used a subset, created by only keeping triples where one of the entities was appearing in either the WEBQUESTIONS training/validation set or in CLUEWEB extractions. We also removed all entities appearing less than 5 times and finally obtained a FREEBASE set containing 14M triples made of 2.2M entities and 7k relation types.[1] Since the format of triples does not correspond to any structure one could find in language, we decided to transform them into automatically generated questions. Hence, all triples were converted into questions "What is the `predicate` of the `type2` `subject`?" (using the `mid` of the subject) with the answer being `object`. An example is *"What is the* `nationality` *of the* `person` `barack_obama`*?"* (`united_states`). More examples and details are given in a longer version of this paper (Bordes et al., 2014a).

ClueWeb Extractions FREEBASE data allows to train our model on 14M questions but these have a fixed lexicon and vocabulary, which is not realistic. Following (Berant et al., 2013), we also created questions using CLUEWEB extractions provided by (Lin et al., 2012). Using string matching, we ended up with 2M extractions structured as (`subject`, "text string", `object`) with both `subject` and `object` linked to FREEBASE. We also converted these triples into questions by using simple patterns and FREEBASE types. An example of generated question is *"Where* `barack_obama` *was allegedly bear in?"* (`hawaii`).

Paraphrases The automatically generated questions that are useful to connect FREEBASE triples and natural language, do not provide a satisfactory modeling of natural language because of their semi-automatic wording and rigid syntax. To overcome this issue, we follow (Fader et al., 2013) and supplement our training data with an indirect supervision signal made of pairs of question paraphrases collected from the WIKIANSWERS website. On WIKIANSWERS, users can tag pairs of questions as rephrasings of each other: (Fader et al., 2013) harvested a set of 2M distinct questions from WIKIANSWERS, which were grouped into 350k paraphrase clusters.

[1] WEBQUESTIONS contains ∼2k entities, hence restricting FREEBASE to 2.2M entities does not ease the task for us.

3 Embedding Questions and Answers

Inspired by (Bordes et al., 2014b), our model works by learning low-dimensional vector embeddings of words appearing in questions and of entities and relation types of FREEBASE, so that representations of questions and of their corresponding answers are close to each other in the joint embedding space. Let q denote a question and a a candidate answer. Learning embeddings is achieved by learning a scoring function $S(q, a)$, so that S generates a high score if a is the correct answer to the question q, and a low score otherwise. Note that both q and a are represented as a combination of the embeddings of their individual words and/or symbols; hence, learning S essentially involves learning these embeddings. In our model, the form of the scoring function is:

$$S(q, a) = f(q)^\top g(a). \quad (1)$$

Let $\mathbf{W}$ be a matrix of $\mathbb{R}^{k \times N}$, where k is the dimension of the embedding space which is fixed a-priori, and N is the dictionary of embeddings to be learned. Let N_W denote the total number of words and N_S the total number of entities and relation types. With $N = N_W + N_S$, the i-th column of $\mathbf{W}$ is the embedding of the i-th element (word, entity or relation type) in the dictionary. The function $f(.)$, which maps the questions into the embedding space $\mathbb{R}^k$ is defined as $f(q) = \mathbf{W}\phi(q)$, where $\phi(q) \in \mathbb{N}^N$, is a sparse vector indicating the number of times each word appears in the question q (usually 0 or 1). Likewise the function $g(.)$ which maps the answer into the same embedding space $\mathbb{R}^k$ as the questions, is given by $g(a) = \mathbf{W}\psi(a)$. Here $\psi(a) \in \mathbb{N}^N$ is a sparse vector representation of the answer a, which we now detail.

3.1 Representing Candidate Answers

We now describe possible feature representations for a single candidate answer. (When there are multiple correct answers, we average these representations, see Section 3.4.) We consider three different types of representation, corresponding to different subgraphs of FREEBASE around it.

(i) *Single Entity*. The answer is represented as a single entity from FREEBASE: $\psi(a)$ is a 1-of-N_S coded vector with 1 corresponding to the entity of the answer, and 0 elsewhere.

(ii) *Path Representation*. The answer is represented as a path from the entity mentioned in the question to the answer entity. In our experiments, we considered 1- or 2-hops paths (i.e. with either 1 or 2 edges to traverse): (`barack_obama`, `people.person.place_of_birth`, `honolulu`) is a 1-hop path and (`barack_obama`, `people.person.place_of_birth`, `location.location.containedby`, `hawaii`) a 2-hops path. This results in a $\psi(a)$ which is a 3-of-N_S or 4-of-N_S coded vector, expressing the start and end entities of the path and the relation types (but not entities) in-between.

(iii) *Subgraph Representation*. We encode both the path representation from (ii), and the entire subgraph of entities connected to the candidate answer entity. That is, for each entity connected to the answer we include both the relation type and the entity itself in the representation $\psi(a)$. In order to represent the answer path differently to the surrounding subgraph (so the model can differentiate them), we double the dictionary size for entities, and use one embedding representation if they are in the path and another if they are in the subgraph. Thus we now learn a parameter matrix $\mathbb{R}^{k \times N}$ where $N = N_W + 2N_S$ (N_S is the total number of entities and relation types). If there are C connected entities with D relation types to the candidate answer, its representation is a $3+C+D$ or $4+C+D$-of-N_S coded vector, depending on the path length.

Our hypothesis is that including more information about the answer in its representation will lead to improved results. While it is possible that all required information could be encoded in the k dimensional embedding of the single entity *(i)*, it is unclear what dimension k should be to make this possible. For example the embedding of a country entity encoding all of its citizens seems unrealistic. Similarly, only having access to the path ignores all the other information we have about the answer entity, unless it is encoded in the embeddings of either the entity of the question, the answer or the relations linking them, which might be quite complicated as well. We thus adopt the subgraph approach. Figure 1 illustrates our model.

3.2 Training and Loss Function

As in (Weston et al., 2010), we train our model using a margin-based ranking loss function. Let $\mathcal{D} = \{(q_i, a_i) : i = 1, \ldots, |\mathcal{D}|\}$ be the training set

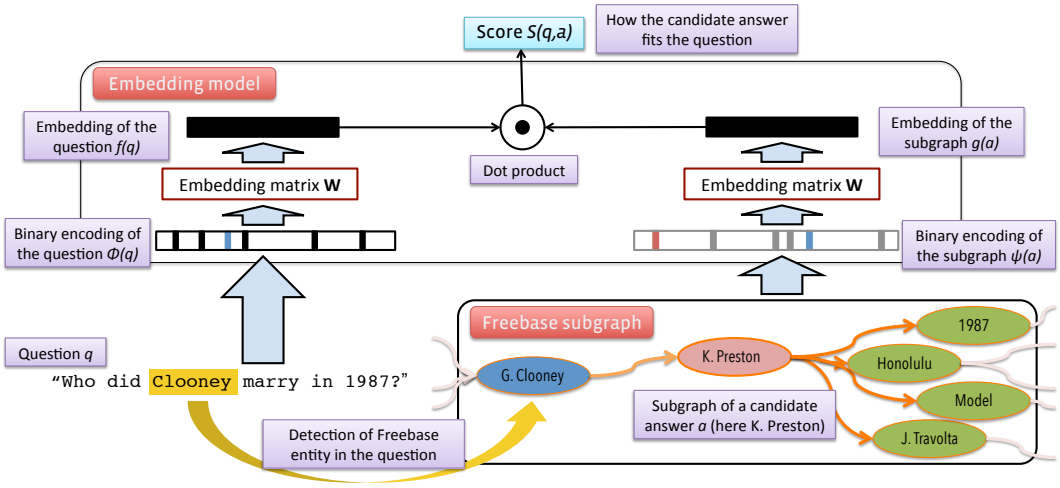


Figure 1: Illustration of the subgraph embedding model scoring a candidate answer: (i) locate entity in the question; (ii) compute path from entity to answer; (iii) represent answer as path plus all connected entities to the answer (the subgraph); (iv) embed both the question and the answer subgraph separately using the learnt embedding vectors, and score the match via their dot product.

of questions q_i paired with their correct answer a_i. The loss function we minimize is

$$\sum_{i=1}^{|\mathcal{D}|} \sum_{\bar{a} \in \bar{\mathcal{A}}(a_i)} \max\{0, m - S(q_i, a_i) + S(q_i, \bar{a})\}, \quad (2)$$

where m is the margin (fixed to 0.1). Minimizing Eq. (2) learns the embedding matrix $\mathbf{W}$ so that the score of a question paired with a correct answer is greater than with any incorrect answer $\bar{a}$ by at least m. $\bar{a}$ is sampled from a set of incorrect candidates $\bar{\mathcal{A}}$. This is achieved by sampling 50% of the time from the set of entities connected to the entity of the question (i.e. other candidate paths), and by replacing the answer entity by a random one otherwise. Optimization is accomplished using stochastic gradient descent, multi-threaded with Hogwild! (Recht et al., 2011), with the constraint that the columns w_i of $\mathbf{W}$ remain within the unit-ball, i.e., $\forall_i, ||w_i||_2 \leq 1$.

3.3 Multitask Training of Embeddings

Since a large number of questions in our training datasets are synthetically generated, they do not adequately cover the range of syntax used in natural language. Hence, we also multi-task the training of our model with the task of paraphrase prediction. We do so by alternating the training of S with that of a scoring function $S_{prp}(q_1, q_2) = f(q_1)^\top f(q_2)$, which uses the same embedding matrix $\mathbf{W}$ and makes the embeddings of a pair of questions (q_1, q_2) similar to each other if they are paraphrases (i.e. if they belong to the same paraphrase cluster), and make them different otherwise. Training S_{prp} is similar to that of S except that negative samples are obtained by sampling a question from another paraphrase cluster.

We also multitask the training of the embeddings with the mapping of the `mids` of FREEBASE entities to the actual words of their names, so that the model learns that the embedding of the `mid` of an entity should be similar to the embedding of the word(s) that compose its name(s).

3.4 Inference

Once $\mathbf{W}$ is trained, at test time, for a given question q the model predicts the answer with:

$$\hat{a} = \text{argmax}_{a' \in \mathcal{A}(q)} S(q, a') \quad (3)$$

where $\mathcal{A}(q)$ is the candidate answer set. This candidate set could be the whole KB but this has both speed and potentially precision issues. Instead, we create a candidate set $\mathcal{A}(q)$ for each question.

We recall that each question contains one identified FREEBASE entity. $\mathcal{A}(q)$ is first populated with all triples from FREEBASE involving this entity. This allows to answer simple factual questions whose answers are directly connected to them (i.e. 1-hop paths). This strategy is denoted C_1.

Since a system able to answer only such questions would be limited, we supplement $\mathcal{A}(q)$ with examples situated in the KB graph at 2-hops from the entity of the question. We do not add all such quadruplets since this would lead to very large candidate sets. Instead, we consider the following general approach: given that we are predicting a path, we can predict its elements in turn using

Method	P@1 (%)	F1 (Berant)	F1 (Yao)
Baselines			
(Berant et al., 2013)	–	31.4	–
(Bordes et al., 2014b)	31.3	29.7	31.8
(Yao and Van Durme, 2014)	–	33.0	42.0
(Berant and Liang, 2014)	–	39.9	43.0
Our approach			
Subgraph & $\mathcal{A}(q) = C_2$	**40.4**	39.2	43.2
Ensemble with (Berant & Liang, 14)	–	**41.8**	**45.7**
Variants			
Without multiple predictions	**40.4**	31.3	34.2
Subgraph & $\mathcal{A}(q) =$ All 2-hops	38.0	37.1	41.4
Subgraph & $\mathcal{A}(q) = C_1$	34.0	32.6	35.1
Path & $\mathcal{A}(q) = C_2$	36.2	35.3	38.5
Single Entity & $\mathcal{A}(q) = C_1$	25.8	16.0	17.8

Table 1: Results on the WEBQUESTIONS test set.

a beam search, and hence avoid scoring all candidates. Specifically, our model first ranks relation types using Eq. (1), i.e. selects which relation types are the most likely to be expressed in q. We keep the top 10 types (10 was selected on the validation set) and only add 2-hops candidates to $\mathcal{A}(q)$ when these relations appear in their path. Scores of 1-hop triples are weighted by 1.5 since they have one less element than 2-hops quadruplets. This strategy, denoted C_2, is used by default.

A prediction a' can commonly actually be a set of candidate answers, not just one answer, for example for questions like *"Who are David Beckham's children?"*. This is achieved by considering a prediction to be all the entities that lie on the same 1-hop or 2-hops path from the entity found in the question. Hence, all answers to the above question are connected to `david_beckham` via the same path (`david_beckham`, `people.person.children`, $\star$). The feature representation of the prediction is then the average over each candidate entity's features (see Section 3.1), i.e. $\psi_{all}(a') = \frac{1}{|a'|} \sum_{a'_j : a'} \psi(a'_j)$ where a'_j are the individual entities in the overall prediction a'. In the results, we compare to a baseline method that can only predict single candidates, which understandly performs poorly.

4 Experiments

We compare our system in terms of F1 score as computed by the official evaluation script[2] (F1 (Berant)) but also with a slightly different F1 definition, termed F1 (Yao) which was used in (Yao and Van Durme, 2014) (the difference being the way that questions with no answers are dealt with), and precision @ 1 (p@1) of the first candidate entity (even when there are a set of correct answers), comparing to recently published systems.[3] The upper part of Table 1 indicates that our approach outperforms (Yao and Van Durme, 2014), (Berant et al., 2013) and (Bordes et al., 2014b), and performs similarly as (Berant and Liang, 2014).

The lower part of Table 1 compares various versions of our model. Our default approach uses the Subgraph representation for answers and C_2 as the candidate answers set. Replacing C_2 by C_1 induces a large drop in performance because many questions do not have answers thatare directly connected to their inluded entity (not in C_1). However, using all 2-hops connections as a candidate set is also detrimental, because the larger number of candidates confuses (and slows a lot) our ranking based inference. Our results also verify our hypothesis of Section 3.1, that a richer representation for answers (using the local subgraph) can store more pertinent information. Finally, we demonstrate that we greatly improve upon the model of (Bordes et al., 2014b), which actually corresponds to a setting with the Path representation and C_1 as candidate set.

We also considered an ensemble of our approach and that of (Berant and Liang, 2014). As we only had access to their test predictions we used the following combination method. Our approach gives a score $S(q, a)$ for the answer it predicts. We chose a threshold such that our approach predicts 50% of the time (when $S(q, a)$ is above its value), and the other 50% of the time we use the prediction of (Berant and Liang, 2014) instead. We aimed for a 50/50 ratio because both methods perform similarly. The ensemble improves the state-of-the-art, and indicates that our models are significantly different in their design.

5 Conclusion

This paper presented an embedding model that learns to perform open QA using training data made of questions paired with their answers and of a KB to provide a structure among answers, and can achieve promising performance on the competitive benchmark WEBQUESTIONS.

[2] Available from www-nlp.stanford.edu/software/sempre/

[3] Results of baselines except (Bordes et al., 2014b) have been extracted from the original papers. For our experiments, all hyperparameters have been selected on the WEBQUESTIONS validation set: k was chosen among $\{64, 128, 256\}$, the learning rate on a log. scale between 10^{-4} and 10^{-1} and we used at most 100 paths in the subgraph representation.

References

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14)*, Baltimore, USA.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*, Seattle, USA.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, Vancouver, Canada. ACM.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. *CoRR*, abs/1406.3676.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. In *Proceedings of the 7th European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD'14)*, Nancy, France. Springer.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, Sofia, Bulgaria.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of 20th SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'14)*, New York City, USA. ACM.

Oleksandr Kolomiyets and Marie-Francine Moens. 2011. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412–5434.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*, Seattle, USA, October.

Thomas Lin, Mausam, and Oren Etzioni. 2012. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX'12)*, Montreal, Canada.

Benjamin Recht, Christopher Ré, Stephen J Wright, and Feng Niu. 2011. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems (NIPS 24).*, Vancouver, Canada.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over RDF data. In *Proceedings of the 21st international conference on World Wide Web (WWW'12)*, Lyon, France. ACM.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2010. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1).

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14)*, Baltimore, USA.

Correcting Keyboard Layout Errors and Homoglyphs in Queries

Derek Barnes debarnes@ebay.com
Mahesh Joshi mahesh.joshi@ebay.com
Hassan Sawaf hsawaf@ebay.com

eBay Inc., 2065 Hamilton Ave, San Jose, CA, 95125, USA

Abstract

Keyboard layout errors and homoglyphs in cross-language queries impact our ability to correctly interpret user information needs and offer relevant results. We present a machine learning approach to correcting these errors, based largely on character-level n-gram features. We demonstrate superior performance over rule-based methods, as well as a significant reduction in the number of queries that yield null search results.

1 Introduction

The success of an eCommerce site depends on how well users are connected with products and services of interest. Users typically communicate their desires through search queries; however, queries are often incomplete and contain errors, which impact the quantity and quality of search results.

New challenges arise for search engines in cross-border eCommerce. In this paper, we focus on two cross-linguistic phenomena that make interpreting queries difficult: **(i) Homoglyphs**: (Miller, 2013): Tokens such as "саse" (underlined letters Cyrillic), in which users mix characters from different character sets that are visually similar or identical. For instance, English and Russian alphabets share homoglyphs such as c, a, e, o, y, k, etc. Although the letters are visually similar or in some cases identical, the underlying character codes are different. **(ii) Keyboard Layout Errors (KLEs)**: (Baytin et al., 2013): When switching one's keyboard between language modes, users at times enter terms in the wrong character set. For instance, "чехол шзфв" may appear to be a Russian query. While "чехол" is the Russian word for "case", "шзфв" is actually the user's attempt to enter the characters "ipad" while leaving their keyboard in Russian language mode. Queries containing KLEs or homoglyphs are unlikely to produce any search results, unless the intended ASCII sequences can be recovered. In a test set sampled from Russian/English queries with null (i.e. empty) search results (see Section 3.1), we found approximately 7.8% contained at least one KLE or homoglyph.

In this paper, we present a machine learning approach to identifying and correcting query tokens containing homoglyphs and KLEs. We show that the proposed method offers superior accuracy over rule-based methods, as well as significant improvement in search recall. Although we focus our results on Russian/English queries, the techniques (particularly for KLEs) can be applied to other language pairs that use different character sets, such as Korean-English and Thai-English.

2 Methodology

In cross-border trade at eBay, multilingual queries are translated into the inventory's source language prior to search. A key application of this, and the focus of this paper, is the translation of Russian queries into English, in order to provide Russian users a more convenient interface to English-based inventory in North America. The presence of KLEs and homoglyphs in multilingual queries, however, leads to poor query translations, which in turn increases the incidence of null search results. We have found that null search results correlate with users exiting our site.

In this work, we seek to correct for KLEs and homoglyphs, thereby improving query translation, reducing the incidence of null search results, and increasing user engagement. Prior to translation and search, we preprocess multilingual queries by identifying and transforming KLEs and homoglyphs as follows (we use the query "чехол шзфв 2 new" as a running example):

(a) Tag Tokens: label each query token

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 621–626,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

with one of the following semantically motivated classes, which identify the user's information need: (i) E: a token intended as an English search term; (ii) R: a Cyrillic token intended as a Russian search term; (iii) K: A KLE, e.g. "шзфв" for the term "ipad". A token intended as an English search term, but at least partially entered in the Russian keyboard layout; (iv) H: A Russian homoglyph for an English term, e.g. "вмw" (underlined letters Cyrillic). Employs visually similar letters from the Cyrillic character set when spelling an intended English term; (v) A: Ambiguous tokens, consisting of numbers and punctuation characters with equivalent codes that can be entered in both Russian and English keyboard layouts. Given the above classes, our example query "чехол шзфв 2 new" should be tagged as "R K A E".

(b) Transform Queries: Apply a deterministic mapping to transform KLE and homoglyph tokens from Cyrillic to ASCII characters. For KLEs the transformation maps between characters that share the same location in Russian and English keyboard layouts (e.g. ф → a, ы → s). For homoglyphs the transformation maps between a smaller set of visually similar characters (e.g. е → e, м → m). Our example query would be transformed into "чехол ipad 2 new".

(c) Translate and Search: Translate the transformed query (into "case ipad 2 new" for our example), and dispatch it to the search engine.

In this paper, we formulate the token-level tagging task as a standard multiclass classification problem (each token is labeled independently), as well as a sequence labeling problem (a first order conditional Markov model). In order to provide end-to-end results, we preprocess queries by deterministically transforming into ASCII the tokens tagged by our model as KLEs or homoglyphs. We conclude by presenting an evaluation of the impact of this transformation on search.

2.1 Features

Our classification and sequence models share a common set of features grouped into the following categories:

2.1.1 Language Model Features

A series of 5-gram, character-level language models (LMs) capture the structure of different types of words. Intuitively, valid Russian terms will have high probability in Russian LMs. In contrast, KLEs or homoglyph tokens, despite appearing on the surface to be Russian terms, will generally have low probability in the LMs trained on valid Russian words. Once mapped into ASCII (see Section 2 above), however, these tokens tend to have higher probability in the English LMs. LMs are trained on the following corpora:

English and Russian Vocabulary: based on a collection of open source, parallel English/Russian corpora (∼50M words in all).

English Brands: built from a curated list of 35K English brand names, which often have distinctive linguistic properties compared with common English words (Lowrey et al., 2013).

Russian Transliterations: built from a collection of Russian transliterations of proper names from Wikipedia (the Russian portion of `guessed-names.ru-en` made available as a part of WMT 2013[1]).

For every input token, each of the above LMs fires a real-valued feature — the negated log-probability of the token in the given language model. Additionally, for tokens containing Cyrillic characters, we consider the token's KLE and homoglyph ASCII mappings, where available. For each mapping, a real-valued feature fires corresponding to the negated log-probability of the mapped token in the English and Brands LMs. Lastly, an equivalent set of LM features fires for the two preceding and following tokens around the current token, if applicable.

2.1.2 Token Features

We include several features commonly used in token-level tagging problems, such as case and shape features, token class (such as letters-only, digits-only), position of the token within the query, and token length. In addition, we include features indicating the presence of characters from the ASCII and/or Cyrillic character sets.

2.1.3 Dictionary Features

We incorporate a set of features that indicate whether a given lowercased query token is a member of one of the lexicons described below.

UNIX: The English dictionary shipped with CentOS, including ∼480K entries, used as a lexicon of common English words.

BRANDS: An expanded version of the curated list of brand names used for LM features. Includes

[1] `www.statmt.org/wmt13/translation-task.html#download`

∼58K brands.

PRODUCT TITLES: A lexicon of over 1.6M entries extracted from a collection of 10M product titles from eBay's North American inventory.

QUERY LOGS: A larger, in-domain collection of approximately 5M entries extracted from ∼100M English search queries on eBay.

Dictionary features fire for Cyrillic tokens when the KLE and/or homoglyph-mapped version of the token appears in the above lexicons. Dictionary features are binary for the Unix and Brands dictionaries, and weighted by relative frequency of the entry for the Product Titles and Query Logs dictionaries.

3 Experiments

3.1 Datasets

The following datasets were used for training and evaluating the baseline (see Section 3.2 below) and our proposed systems:

Training Set: A training set of 6472 human-labeled query examples (17,239 tokens).

In-Domain Query Test Set: A set of 2500 Russian/English queries (8,357 tokens) randomly selected from queries with null search results. By focusing on queries with null results, we emphasize the presence of KLEs and homoglyphs, which occur in 7.8% of queries in our test set.

Queries were labeled by a team of Russian language specialists. The test set was also independently reviewed, which resulted in the correction of labels for 8 out of the 8,357 query tokens.

Although our test set is representative of the types of problematic queries targeted by our model, our training data was not sampled using the same methodology. We expect that the differences in distributions between training and test sets, if anything, make the results reported in Section 3.3 somewhat pessimistic[2].

3.2 Dictionary Baseline

We implemented a rule-based baseline system employing the dictionaries described in Section 2.1.3. In this system, each token was assigned a class $k \in \{\texttt{E}, \texttt{R}, \texttt{K}, \texttt{H}, \texttt{A}\}$ using a set of rules: a token among a list of 101 Russian stopwords[3] is tagged as R. A token containing only ASCII characters is labeled as A if all characters are common to English and Russian keyboards (i.e. numbers and some punctuation), otherwise E. For tokens containing Cyrillic characters, KLE and homoglyph-mapped versions are searched in our dictionaries. If found, K or H are assigned. If both mapped versions are found in the dictionaries, then either K or H is assigned probabilistically[4]. In cases where neither mapped version is found in the dictionary, the token assigned is either R or A, depending on whether it consists of purely Cyrillic characters, or a mix of Cyrillic and ASCII, respectively.

Note that the above tagging rules allow tokens with classes E and A to be identified with perfect accuracy. As a result, we omit these classes from all results reported in this work. We also note that this simplification applies because we have restricted our attention to the Russian → English direction. In the bidirectional case, ASCII tokens could represent either English tokens or KLEs (i.e. a Russian term entered in the English keyboard layout). We leave the joint treatment of the bidirectional case to future work.

Tag	Prec	Recall	F1
K	.528	.924	.672
H	.347	.510	.413
R	.996	.967	.982

Table 1: Baseline results on the test set, using UNIX, BRANDS, and the PRODUCT TITLES dictionaries.

We experimented with different combinations of dictionaries, and found the best combination to be UNIX, BRANDS, and PRODUCT TITLES dictionaries (see Table 1). We observed a sharp decrease in precision when incorporating the QUERY LOGS dictionary, likely due to noise in the user-generated content.

Error analysis suggests that shorter words are the most problematic for the baseline system[5]. Shorter Cyrillic tokens, when transformed from Cyrillic to ASCII using KLE or homoglyph mappings, have a higher probability of spuriously mapping to valid English acronyms, model IDs, or short words. For instance, Russian car brand "ваз" maps across keyboard layouts to "dfp",

[2] As expected, cross-validation experiments on the training data (for parameter tuning) yielded results slightly higher than the results reported in Section 3.3, which use a held-out test set

[3] Taken from the Russian Analyzer packaged with Lucene — see lucene.apache.org.

[4] We experimented with selecting K or H based on a prior computed from training data; however, results were lower than those reported, which use random selection.

[5] Stopwords are particularly problematic, and hence excluded from consideration as KLEs or homoglyphs.

	Tag	Classification			Sequence		
		P	R	F1	P	R	F1
LR	K	.925	.944	.935	.915	.934	.925
	H	.708	.667	.687	.686	.686	.686
	R	.996	.997	.996	.997	.996	.997
RF	K	.926	.949	.937	.935	.949	.942
	H	.732	.588	.652	.750	.588	.659
	R	.997	.997	.997	.996	.998	.997

Table 2: Classification and sequence tagging results on the test set

a commonly used acronym in product titles for "Digital Flat Panel". Russian words "муки" and "рук" similarly map by chance to English words "verb" and "her".

A related problem occurs with product model IDs, and highlights the limits of treating query tokens independently. Consider Cyrillic query "БМВ e46". The first token is a Russian transliteration for the BMW brand. The second token, "e46", has three possible interpretations: i) as a Russian token; ii) a homoglyph for ASCII "e46"; or iii) a KLE for "t46". It is difficult to discriminate between these options without considering token context, and in this case having some prior knowledge that e46 is a BMW model.

3.3 Machine Learning Models

We trained linear classification models using logistic regression (LR)[6], and non-linear models using random forests (RFs), using implementations from the Scikit-learn package (Pedregosa et al., 2011). Sequence models are implemented as first order conditional Markov models by applying a beam search ($k = 3$) on top of the LR and RF classifiers. The LR and RF models were tuned using 5-fold cross-validation results, with models selected based on the mean F1 score across R, K, and H tags.

Table 2 shows the token-level results on our in-domain test set. As with the baseline, we focus the model on disambiguating between classes R, K and H. Each of the reported models performs significantly better than the baseline (on each tag), with statistical significance evaluated using McNemar's test. The differences between LR and RF models, as well as sequence and classification variants, however, are not statistically significant. Each of the machine learning models achieves a query-level accuracy score of roughly 98% (the LR sequence model achieved the lowest with 97.78%, the RF sequence model the highest with 97.90%).

Our feature ablation experiments show that the majority of predictive power comes from the character-level LM features. Dropping LM features results in a significant reduction in performance (F1 scores .878 and .638 for the RF Sequence model on classes K and H). These results are still significantly above the baseline, suggesting that token and dictionary features are by themselves good predictors. However, we do not see a similar performance reduction when dropping these feature groups.

We experimented with lexical features, which are commonly used in token-level tagging problems. Results, however, were slightly lower than the results reported in this section. We suspect the issue is one of overfitting, due to the limited size of our training data, and general sparsity associated with lexical features. Continuous word presentations (Mikolov et al., 2013), noted as future work, may offer improved generalization.

Error analysis for our machine learning models suggests patterns similar to those reported in Section 3.2. Although errors are significantly less frequent than in our dictionary baseline, shorter words still present the most difficulty. We note as future work the use of word-level LM scores to target errors with shorter words.

3.4 Search Results

Recall that we translate multilingual queries into English prior to search. KLEs and homoglyphs in queries result in poor query translations, often leading to null search results.

To evaluate the impact of KLE and homoglyph correction, we consider a set of 100k randomly selected Russian/English queries. We consider the subset of queries that the RF or baseline models predict as containing a KLE or homoglyph. Next, we translate into English both the original query, as well as a transformed version of it, with KLEs and homoglyphs replaced with their ASCII mappings. Lastly, we execute independent searches using original and transformed query translations.

Table 3 provides details on search results for original and transformed queries. The baseline model transforms over 12.6% of the 100k queries. Of those, 24.3% yield search results where the unmodified queries had null search results (i.e. Null → Non-null). In 20.9% of the cases, however, the

[6]Although CRFs are state-of-the-art for many tagging problems, in our experiments they yielded results slightly lower than LR or RF models.

transformations are destructive (i.e. Non-null → Null), and yield null results where the unmodified query produced results.

Compared with the baseline, the RF model transforms only 7.4% of the 100k queries; a fraction that is roughly in line with the 7.8% of queries in our test set that contain KLEs or homoglyphs. In over 42% of the cases (versus 24.3% for the baseline), the transformed query generates search results where the original query yields none. Only 4.81% of the transformations using the RF model are destructive; a fraction significantly lower than the baseline.

Note that we distinguish here only between queries that produce null results, and those that do not. We do not include queries for which original and transformed queries both produce (potentially differing) search results. Evaluating these cases requires deeper insight into the relevance of search results, which is left as future work.

	Baseline	**RF model**
#Transformed	12,661	7,364
Null → Non-Null	3,078 (24.3%)	**3,142 (42.7%)**
Non-Null → Null	2,651 (20.9%)	**354 (4.81%)**

Table 3: Impact of KLE and homoglyph correction on search results for 100k queries

4 Related Work

Baytin et al. (2013) first refer to keyboard layout errors in their work. However, their focus is on predicting the performance of spell-correction, not on fixing KLEs observed in their data. To our knowledge, our work is the first to introduce this problem and to propose a machine learning solution. Since our task is a token-level tagging problem, it is very similar to the part-of-speech (POS) tagging task (Ratnaparkhi, 1996), only with a very small set of candidate tags. We chose a supervised machine learning approach in order to achieve maximum precision. However, this problem can also be approached in an unsupervised setting, similar to the method Whitelaw et al. (2009) use for spelling correction. In that setup, the goal would be to directly choose the correct transformation for an ill-formed KLE or homoglyph, instead of a tagging step followed by a deterministic mapping to ASCII.

5 Conclusions and Future Work

We investigate two kinds of errors in search queries: keyboard layout errors (KLEs) and homoglyphs. Applying machine learning methods, we are able to accurately identify a user's intended query, in spite of the presence of KLEs and homoglyphs. The proposed models are based largely on compact, character-level language models. The proposed techniques, when applied to multilingual queries prior to translation and search, offer significant gains in search results.

In the future, we plan to focus on additional features to improve KLE and homoglyph discrimination for shorter words and acronyms. Although lexical features did not prove useful for this work, presumably due to data sparsity and overfitting issues, we intend to explore the application of continuous word representations (Mikolov et al., 2013). Compared with lexical features, we expect continuous representations to be less susceptible to overfitting, and to generalize better to unknown words. For instance, using continuous word representations, Turian et al. (2010) show significant gains for a named entity recognition task.

We also intend on exploring the use of features from in-domain, word-level LMs. Word-level features are expected to be particularly useful in the case of spurious mappings (e.g. "вaз" vs. "dfp" from Section 3.2), where context from surrounding tokens in a query can often help in resolving ambiguity. Word-level features may also be useful in re-ranking translated queries prior to search, in order to reduce the incidence of erroneous query transformations generated through our methods. Finally, our future work will explore KLE and homoglyph correction bidirectionally, as opposed to the unidirectional approach explored in this work.

Acknowledgments

We would like to thank Jean-David Ruvini, Mike Dillinger, Saša Hasan, Irina Borisova and the anonymous reviewers for their valuable feedback. We also thank our Russian language specialists Tanya Badeka, Tatiana Kontsevich and Olga Pospelova for their support in labeling and reviewing datasets.

References

Alexey Baytin, Irina Galinskaya, Marina Panina, and Pavel Serdyukov. 2013. Speller performance pre-

diction for query autocorrection. In *Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management*, pages 1821–1824.

Tina M. Lowrey, Larry J. Shrum, and Tony M. Dubitsky. 2013. The Relation Between Brand-name Linguistic Characteristics and Brand-name Memory. *Journal of Advertising*, 32(3):7–17.

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

Tristan Miller. 2013. Russian–English Homoglyphs, Homographs, and Homographic Translations. *Word Ways: The Journal of Recreational Linguistics*, 46(3):165–168.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Model for Part–of–Speech Tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

Casey Whitelaw, Ben Hutchinson, Grace Y. Chung, and Ged Ellis. 2009. Using the Web for Language Independent Spellchecking and Autocorrection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 890–899.

Non-linear Mapping for Improved Identification of 1300+ Languages

Ralf D. Brown
Carnegie Mellon University Language Technologies Institute
5000 Forbes Avenue, Pittsburgh PA 15213 USA
ralf@cs.cmu.edu

Abstract

Non-linear mappings of the form $P(ngram)^\gamma$ and $\frac{log(1+\tau P(ngram))}{log(1+\tau)}$ are applied to the n-gram probabilities in five trainable open-source language identifiers. The first mapping reduces classification errors by 4.0% to 83.9% over a test set of more than one million 65-character strings in 1366 languages, and by 2.6% to 76.7% over a subset of 781 languages. The second mapping improves four of the five identifiers by 10.6% to 83.8% on the larger corpus and 14.4% to 76.7% on the smaller corpus. The subset corpus and the modified programs are made freely available for download at http://www.cs.cmu.edu/∼ralf/langid.html.

1 Introduction

Language identification, particularly of short strings, is a task which is becoming quite important as a preliminary step in much automated processing of online data streams such as microblogs (e.g. Twitter). In addition, an increasing number of languages are represented online, so it is desireable that performance remain high as more languages are added to the identifier.

In this paper, we stress-test five open-source n-gram-based language identifiers by presenting them with 65-character strings (about one printed line of text in a book) in up to 1366 languages. We then apply a simple modification to their scoring algorithms which improves the classification accuracy of all five of them, three quite dramatically.

2 Method

The selected modification to the scoring algorithm is to apply a non-linear mapping which spreads out the lower probability values while compacting the higher ones. This low-end spreading of values is the opposite of what one sees in a Zipfian distribution (Zipf, 1935), where the probabilities of the most common items are the most spread out while the less frequent items become ever more crowded as there are increasing numbers of them in ever-smaller ranges. The hypothesis is that regularizing the spacing between values will improve language-identification accuracy by avoiding over-weighting frequent items (from having higher probabilities in the training data and also occurring more frequently in the test string).

Two functions were selected for experiments:

$$x = P(ngram)$$

$$\text{gamma:} \quad y = x^\gamma$$

$$\text{loglike:} \quad y = \frac{log(1 + 10^\tau x)}{log(1 + 10^\tau)}$$

The first simply raises the n-gram probability to a non-unity power; this exponent is named "gamma" as in image processing (Poynton, 1998). The second mapping function is a normalized variant of the logarithm function; the normalization provides fixed points at 0 and 1, as is the case for *gamma*. Each of the functions *gamma* and *loglike* has one tunable parameter, γ and τ, respectively.

3 Related Work

Although n-gram statistics as a basis for language identification has been in use for two decades since Cavnar and Trenkle (1994) and Dunning (1994), little work has been done on trying to optimize the values used for those n-gram statistics. Where some form of frequency mapping is used, it is often implicit (as in Cavnar and Trenkle's use of ranks instead of frequencies) and generally goes unremarked as such.

Vogel and Tresner-Kirsch (2012) use the logarithm of the frequency for some experimental runs, reporting that it improved accuracy in some cases. Gebre *et al* (2013) used logarithmic term-frequency scaling of words in an English-language

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 627–632,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

essay to classify the native language of the writer, reporting an improvement from 82.36% accuracy to 84.55% in conjunction with inverse document frequency (IDF) weighting, and from 79.18% accuracy to 80.82% without IDF.

4 Programs

4.1 `LangDetect`

`LangDetect`, version 2011-09-13 (Shuyo, 2014), uses the Naive Bayes approach. Inputs are split into a bag of character n-grams of length 1 through 3; each randomly-drawn n-gram's probability in each of the trained models is multiplied by the current score for that model. After 1000 n-grams, or when periodic renormalization into a probability distribution detects that one model has accumulated an overwhelming probability mass, the iteration is terminated. After averaging seven randomized iterations, each with a random gaussian offset (mean 5×10^{-6}, standard deviation 0.5×10^{-6}) that is added to each probability prior to multiplication (to avoid multiplication by zero), the highest-scoring model is declared to be the language of the input.

The mapping function is applied to the model's probability before adding the randomized offset. To work around the limitation of one model per language code, disambiguating digits are appended to the language code during training and removed from the output prior to scoring.

4.2 `libtextcat`

`libtextcat`, version 2.2-9 (Hugueney, 2011), is a C reimplementation of the Cavnar and Trenkle (1994) method. It compiles "fingerprints" containing a ranked list of the 400 (by default) most frequent 1- through 5-grams in the training data. An unknown text is classified by forming its fingerprint and comparing that fingerprint against the trained fingerprints. A penalty is assigned based on the number of positions by which each n-gram differs between the input and the trained model; n-grams which appear in only one of the two are assigned the maximum penalty, equal to the size of the fingerprints. The model with the lowest penalty score is selected as the language of the input.

For this work, the `libtextcat` source code was modified to remove the hard-coded fingerprint size of 400 n-grams. While adding the frequency mapping, the code was discovered to also hard-code the maximum distortion penalty at 400; this was corrected to set the maximum penalty equal to the maximum size of any loaded fingerprint.[1]

Score mapping was implemented by dividing each penalty value by the maximum penalty to produce a proportion, applying the mapping function, and then multiplying the result by the maximum penalty and rounding to an integer (to avoid other code changes). Because there are only a limited number of possible penalties, a lookup table is pre-computed, eliminating the impact on speed.

4.3 `mguesser`

`mguesser`, version 0.4 (Barkov, 2008), is part of the mnoGoSearch search engine. While its documentation indicates that it implements the Cavnar and Trenkle approach, its actual similarity computation is very different. Each training and test text is converted into a 4096-element hash table by extracting byte n-grams of length 6 (truncated at control characters and multiple consecutive blanks), hashing each n-gram using CRC-32, and incrementing the count for the corresponding hash entry. The hash table entries are then normalized to a mean of 0.0 and standard deviation of 1.0, and the similarity is computed as the inner (dot) product of the hash tables treated as vectors. The trained model receiving the highest similarity score against the input is declared the language of the input.

Nonlinear mapping was added by inserting a step just prior to the normalization of the hash table. The counts in the table are converted to probabilities by dividing by the sum of counts, the mapping is applied to that probability, and the result is converted back into a count by multiplying by the original sum of counts.

4.4 `whatlang`

`whatlang`, version 1.24 (Brown, 2014a), is the stand-alone identification program from LA-Strings (Brown, 2013). It performs identification by computing the inner product of byte trigrams through k-grams (k=6 by default and in this work) between the input and the trained models; for speed, the computation is performed incrementally, adding the length-weighted probabil-

[1] The behavior observed by (Brown, 2013) of performance rapidly degrading for fingerprints larger than 500 disappears with this correction. It was an artifact of an increasing proportion of n-grams present in the model receiving penalties greater than n-grams absent from the model.

ity of each n-gram as it is encountered in the input. Models are formed by finding the highest-frequency n-grams of the configured lengths, with some filtering as described in (Brown, 2012).

4.5 YALI

YALI (Yet Another Language Identifier) (Majlis, 2012) is an identifier written in Perl. It performs minor text normalization by collapsing multiple blanks into a single blank and removing leading and trailing blanks from lines. Thereafter, it uses a sliding window to generate byte n-grams of a (configurable) fixed length, and sums the probabilities for each n-gram in each trained model. As with whatlang, this effectively computes the inner products between the input and the models.

Mapping was added by applying the mapping function to the model probabilities as they are read in from disk. As with LangDetect, disambiguating digits were used to allow multiple models per language code.

5 Data

The data used for the experiments described in this paper comes predominantly from Bible translations, Wikipedia, and the Europarl corpus of European parliamentary proceedings (Koehn, 2005). The 1459 files of the training corpus generate 1483 models in 1368 languages. A number of training files generate models in both UTF-8 and ISO 8859-1, numerous languages have multiple training files in different writing systems, and several have multiple files for different regional variants (e.g. European and Brazilian Portugese).

The text for a language is split into training, test, and possibly a disjoint development set. The amount of text per language varies, with quartiles of 1.19/1.47/2.22 million bytes. In general, every thirtieth line of text is reserved for the test set; some smaller languages reserve a higher proportion. If more than 3.2 million bytes remain after reserving the test set, every thirtieth line of the remaining text is reserved as a development set. There are development sets for 220 languages. The unreserved test is used for model training.

The test data is word-wrapped to 65 characters or less, and wrapped lines shorter than 25 bytes are excluded. Up to the first 1000 lines of wrapped text are used for testing. One language with fewer than 50 test strings is excluded from the test set, as is the constructed language Klingon due to heavy pollution with English. In total, the test files contain 1,090,571 lines of text in 1366 languages.

Wikipedia text and many of the Bible translations are redistributable under Creative Commons licenses, and have been packaged into the LTI LangID Corpus (Brown, 2014b). This smaller corpus contains 781 languages, 119 of them with development sets, and a total of 649,589 lines in the test files. The languages are a strict subset of those in the larger corpus, but numerous languages have had Wikipedia text substituted for non-redistributable Bible translations.

6 Experiments

Using the data sets described in the previous section, we ran a sweep of different gamma and tau values for each language identifier to determine their optimal values on both development and test strings. Step sizes for γ were generally 0.1, while those for τ were 1.0, with smaller steps near the minima. Since it does not provide explicit control over model sizes, LangDetect was trained on a maximum of 1,000,000 bytes per model, as reported optimal in (Brown, 2013). The other programs were trained on a maximum of 2,500,000 bytes per model; libtextcat and whatlang used default model sizes of 400 and 3500, respectively, while mguesser was set to the previously-reported 1500 n-grams per model. After some experimentation, YALI was set to use 5-grams, with 3500 n-grams per model to match whatlang.

7 Results

Tables 1 and 2 show the absolute performance and relative percentage change in classification errors for the five programs using the two mapping functions, as well as the values of γ and τ at which the fewest errors were made on the development set. Overall, the smaller corpus performed worse due to the greater percentage of Wikipedia texts, which are polluted with words and phrases in other languages. In the test set, this occasionally causes a correct identification as another language to be scored as an error.

Figures 2 and 3 graph the classification error rates (number of incorrectly-labeled strings divided by total number of strings in the test set) in percent for different values of γ. A gamma of 1.0 is the baseline condition. The dramatic improvements in mguesser, whatlang and YALI are quite evident, while the smaller but non-trivial im-

		gamma mapping			*loglike* mapping		
Program	Error%	Error%	Δ%	γ	Error%	Δ%	τ
LangDet.	3.233	2.767	-14.4	0.80	2.889	-10.6	1.0
libtextcat	6.787	6.514	-4.0	2.20	–	–	–
mguesser	15.704	4.330	-72.4	0.39	4.177	-73.4	3.8
whatlang	13.309	2.136	-83.9	0.27	2.146	-83.8	4.5
YALI	9.883	2.313	-76.6	0.20	2.313	-76.6	8.0

Table 1: Language-identification accuracy on the 1366-language corpus. γ and τ were tuned on the 220-language development set; only marginally better results can be achieved by tuning on the test set.

		gamma mapping			*loglike* mapping		
Program	Error%	Error%	Δ%	γ	Error%	Δ%	τ
LangDet.	3.603	3.093	-14.2	0.68	3.083	-14.4	2.3
libtextcat	6.693	6.521	-2.6	1.70	–	–	–
mguesser	14.200	4.936	-65.2	0.40	4.779	-66.3	3.7
whatlang	11.879	2.770	-76.7	0.14	2.772	-76.7	5.6
YALI	8.726	2.972	-65.9	0.09	2.989	-65.7	9.0

Table 2: Language-identification accuracy on the 781-language corpus. γ and τ were tuned on the 119-language development set. `libtextcat` did not improve with the *loglike* mapping (see text).

provements in `libtextcat` are difficult to discern at this scale. Since `libtextcat` uses much smaller models than the others by default, Figure 1 gives a closer look at its performance for larger model sizes. As the models grow, the absolute baseline performance improves, but the change from gamma-correction decreases and the optimal value of γ also decreases toward 1.0. This hints that the implicit mapping of ranks either becomes closer to optimal, or that *gamma* becomes less effective at correcting it. At a model size of 3000 n-grams, the baseline error rate is 2.465% while the best performance is 2.457% at $\gamma = 1.10$.

That the best γ for `libtextcat` is greater than 1.0 was not entirely unexpected. The power-law distribution of n-gram frequencies implies that the conversion from frequencies to ranks is essentially logarithmic, and $\log n$ eventually becomes less than n^c for any $c > 0$. The implication of $\gamma > 1$ is simply that the conversion to ranks is too strong a correction, which must be partially undone by the *gamma* mapping.

Figures 4 and 5 graph the error rates for different values of τ. On the graph, zero is the baseline condition without mapping for comparison purposes; the mapping function is not the identity for $\tau = 0$. It can clearly be seen that `libtextcat` is hurt by the *loglike* mapping, which never reduces values, even with negative τ. Using the inverse of

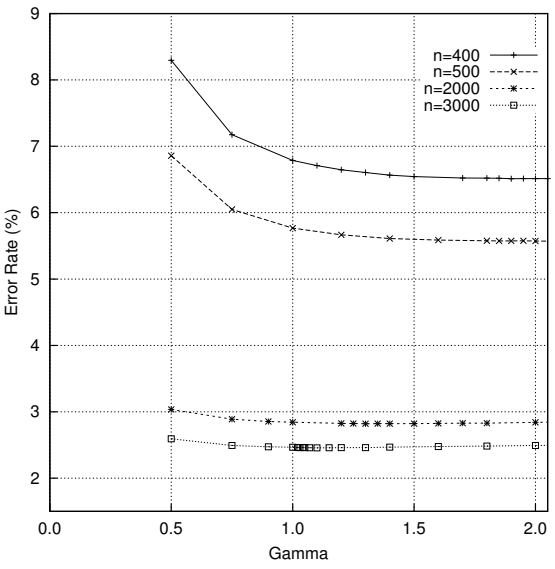


Figure 1: `libtextcat` performance at different fingerprint sizes. $\gamma = 1$ is the baseline.

the *loglike* mapping should improve performance, but has not yet been tried. The other programs show very similar behavior to their results with *gamma*.

8 Conclusions and Future Work

Non-linear mapping is shown to be effective at improving the accuracy of five different language

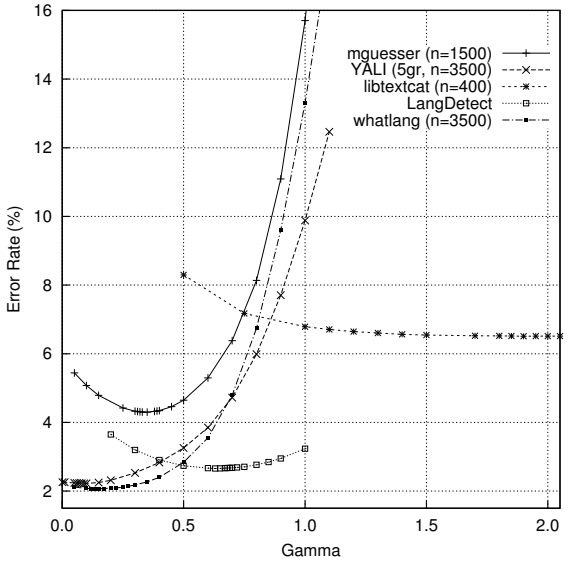


Figure 2: Performance of the identifiers on the 1366-language corpus using the *gamma* mapping.

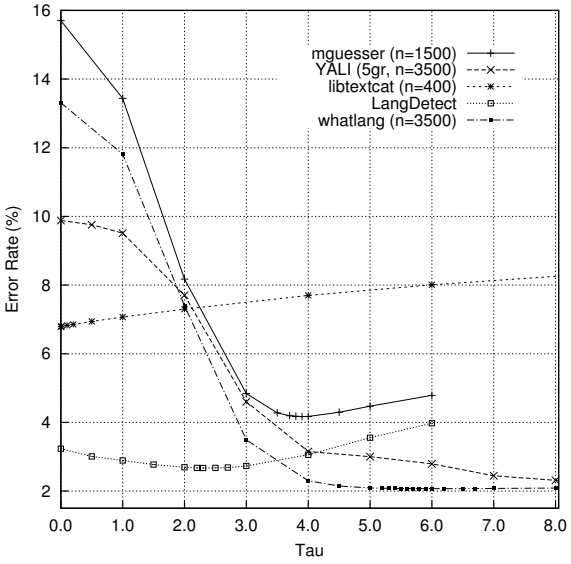


Figure 4: Performance of the identifiers on the 1366-language corpus using the *loglike* mapping.

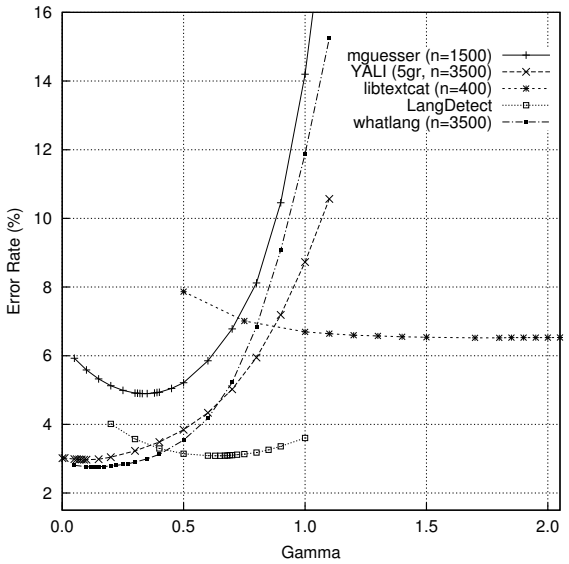


Figure 3: Performance of the identifiers on the 781-language corpus using the *gamma* mapping.

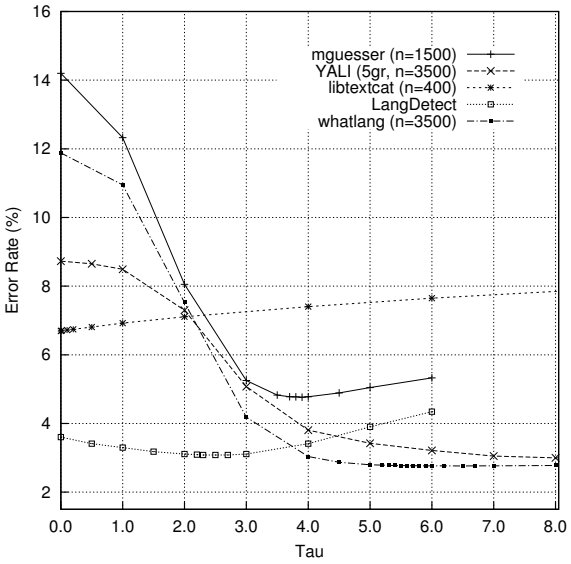


Figure 5: Performance of the identifiers on the 781-language corpus using the *loglike* mapping.

identifier using four highly-divergent algorithms for computing model scores from n-gram statistics. Improvements range from small – 2.6% reduction in classification errors – to dramatic for the three programs with the worst baselines – 65.2 to 76.7% reduction in errors on the smaller corpus and 72.4 to 83.9% on the larger. While both mappings have similar performance for four of the programs, `libtextcat` only benefits from the *gamma* mapping, as it can also reduce n-gram scores, unlike the *loglike* mapping.

Training data, source code, and supplementary information may be downloaded from http://www.cs.cmu.edu/∼ralf/langid.html.

Future work includes modifying additional language identifiers such as `langid.py` (Lui and Baldwin, 2012) and `VarClass` (Zampieri and Gebre, 2014), experimenting with other mapping functions, and investigating the method's efficacy on pluricentric languages like those `VarClass` is designed to identify.

References

Alexander Barkov. 2008. `mguesser` version 0.4. http://www.mnogosearch.org/guesser/-mguesser-0.4.tar.gz (accessed 2014-08-19).

Ralf D. Brown. 2012. Finding and Identifying Text in 900+ Languages. *Digital Investigation*, 9:S34–S43.

Ralf D. Brown. 2013. Selecting and Weighting N-Grams to Identify 1100 Languages. In *Proceedings of Text, Speech, and Discourse 2013*, September.

Ralf Brown. 2014a. Language-Aware String Extractor, August. https://sourceforge.net/projects/la-strings/ (accessed 2014-08-19).

Ralf D. Brown. 2014b. LTI LangID Corpus, Release 1. http://www.cs.cmu.edu/∼ralf/langid.html.

William B. Cavnar and John M. Trenkle. 1994. N-Gram-Based Text Categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175. UNLV Publications/Reprographics, April.

Ted Dunning. 1994. Statistical Identification of Language. Technical Report MCCS 94-273, New Mexico State University.

Binyam Gebrekidan Gebre, Marcos Zampieri, Peter Wittenburg, and Tom Heskes. 2013. Improving Native Language Identification with TF-IDF Weighting. In *Proceedings of the 8th NAACL Workshop on Innovative Use of NLP for Building Educational Applications (BEA8)*.

Bernard Hugueney. 2011. `libtextcat` 2.2-9: Faster Unicode-focused C++ reimplementation of libtextcat. https://github.com/scientific-coder/-libtextcat (accessed 2014-08-19).

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the Tenth Machine Translation Summit (MT Summix X)*, pages 79–86.

Marco Lui and Timothy Baldwin. 2012. langid.py: An Off-the-shelf Language Identification Tool. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL-2012)*, pages 25–30, July.

Martin Majlis. 2012. Yet Another Language Identifier. In *Proceedings of the Student Research Workshop at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 46–54, Avignon, France, April. Association for Computational Linguistics.

Charles Poynton. 1998. The Rehabilitation of Gamma. In *Human Vision and Electronic Imaging III, Proceedings of SPIE/IS&T Conference 3299*, January. http://www.poynton.com/PDFs/-Rehabilitation_of_gamma.pdf.

Nakatani Shuyo. 2014. Language Detection Library for Java, March. http://code.google.com/p/-language-detection/ (accessed 2014-08-19).

John Vogel and David Tresner-Kirsch. 2012. Robust Language Identification in Short, Noisy Texts: Improvements to LIGA. In *Proceedings of the Third International Workshop on Mining Ubiquitous and Social Environments (MUSE 2012)*, pages 43–50, September.

Marcos Zampieri and Binyam Gebrekidan Gebre. 2014. VarClass: An Open Source Language Identification Tool for Language Varieties. In *Proceedings of the Ninth International Language Resources and Evaluation Conference (LREC 2014)*, Reykjavik, Iceland, May.

George Kingsley Zipf. 1935. *The Psycho-biology of Language: An Introduction to Dynamic Philology*. Houghton-Mifflin Co., Boston.

A Neural Network for Factoid Question Answering over Paragraphs

Mohit Iyyer[1], **Jordan Boyd-Graber**[2], **Leonardo Claudino**[1], **Richard Socher**[3], **Hal Daumé III**[1]
[1]University of Maryland, Department of Computer Science and UMIACS
[2]University of Colorado, Department of Computer Science
[3]Stanford University, Department of Computer Science
{miyyer,claudino,hal}@umiacs.umd.edu,
Jordan.Boyd.Graber@colorado.edu, richard@socher.org

Abstract

Text classification methods for tasks like factoid question answering typically use manually defined string matching rules or bag of words representations. These methods are ineffective when question text contains very few individual words (e.g., named entities) that are indicative of the answer. We introduce a recursive neural network (RNN) model that can reason over such input by modeling textual compositionality. We apply our model, QANTA, to a dataset of questions from a trivia competition called quiz bowl. Unlike previous RNN models, QANTA learns word and phrase-level representations that combine across sentences to reason about entities. The model outperforms multiple baselines and, when combined with information retrieval methods, rivals the best human players.

1 Introduction

Deep neural networks have seen widespread use in natural language processing tasks such as parsing, language modeling, and sentiment analysis (Bengio et al., 2003; Socher et al., 2013a; Socher et al., 2013c). The vector spaces learned by these models cluster words and phrases together based on similarity. For example, a neural network trained for a sentiment analysis task such as restaurant review classification might learn that "tasty" and "delicious" should have similar representations since they are synonymous adjectives.

These models have so far only seen success in a limited range of text-based prediction tasks, where inputs are typically a single sentence and outputs are either continuous or a limited discrete set. Neural networks have not yet shown to be useful for tasks that require mapping paragraph-length inputs to rich output spaces.

Later in its existence, this polity's leader was chosen by a group that included three bishops and six laymen, up from the seven who traditionally made the decision. Free imperial cities in this polity included Basel and Speyer. Dissolved in 1806, its key events included the Investiture Controversy and the Golden Bull of 1356. Led by Charles V, Frederick Barbarossa, and Otto I, for 10 points, name this polity, which ruled most of what is now Germany through the Middle Ages and rarely ruled its titular city.

Figure 1: An example quiz bowl question about the Holy Roman Empire. The first sentence contains no words or named entities that by themselves are indicative of the answer, while subsequent sentences contain more and more obvious clues.

Consider factoid question answering: given a description of an entity, identify the person, place, or thing discussed. We describe a task with high-quality mappings from natural language text to entities in Section 2. This task—quiz bowl—is a challenging natural language problem with large amounts of diverse and compositional data.

To answer quiz bowl questions, we develop a dependency tree recursive neural network in Section 3 and extend it to combine predictions across sentences to produce a question answering neural network with trans-sentential averaging (QANTA). We evaluate our model against strong computer and human baselines in Section 4 and conclude by examining the latent space and model mistakes.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 633–644,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

2 Matching Text to Entities: Quiz Bowl

Every weekend, hundreds of high school and college students play a game where they map raw text to well-known entities. This is a trivia competition called *quiz bowl*. Quiz bowl questions consist of four to six sentences and are associated with factoid answers (e.g., history questions ask players to identify specific battles, presidents, or events). Every sentence in a quiz bowl question is guaranteed to contain clues that uniquely identify its answer, even without the context of previous sentences. Players answer at any time—ideally more quickly than the opponent—and are rewarded for correct answers.

Automatic approaches to quiz bowl based on existing NLP techniques are doomed to failure. Quiz bowl questions have a property called *pyramidality*, which means that sentences early in a question contain harder, more obscure clues, while later sentences are "giveaways". This design rewards players with deep knowledge of a particular subject and thwarts bag of words methods. Sometimes the first sentence contains no named entities—answering the question correctly requires an actual understanding of the sentence (Figure 1). Later sentences, however, progressively reveal more well-known and uniquely identifying terms.

Previous work answers quiz bowl questions using a bag of words (naïve Bayes) approach (Boyd-Graber et al., 2012). These models fail on sentences like the first one in Figure 1, a typical hard, initial clue. Recursive neural networks (RNNs), in contrast to simpler models, can capture the compositional aspect of such sentences (Hermann et al., 2013).

RNNs require many redundant training examples to learn meaningful representations, which in the quiz bowl setting means we need multiple questions about the same answer. Fortunately, hundreds of questions are produced during the school year for quiz bowl competitions, yielding many different examples of questions asking about any entity of note (see Section 4.1 for more details). Thus, we have built-in redundancy (the number of "askable" entities is limited), but also built-in diversity, as difficult clues cannot appear in every question without becoming well-known.

3 Dependency-Tree Recursive Neural Networks

To compute distributed representations for the individual sentences within quiz bowl questions, we use a dependency-tree RNN (DT-RNN). These representations are then aggregated and fed into a multinomial logistic regression classifier, where class labels are the answers associated with each question instance.

In previous work, Socher et al. (2014) use DT-RNNs to map text descriptions to images. DT-RNNs are robust to similar sentences with slightly different syntax, which is ideal for our problem since answers are often described by many sentences that are similar in meaning but different in structure. Our model improves upon the existing DT-RNN model by jointly learning answer and question representations in the same vector space rather than learning them separately.

3.1 Model Description

As in other RNN models, we begin by associating each word w in our vocabulary with a vector representation $x_w \in \mathbb{R}^d$. These vectors are stored as the columns of a $d \times V$ dimensional word embedding matrix W_e, where V is the size of the vocabulary. Our model takes dependency parse trees of question sentences (De Marneffe et al., 2006) and their corresponding answers as input.

Each node n in the parse tree for a particular sentence is associated with a word w, a word vector x_w, and a hidden vector $h_n \in \mathbb{R}^d$ of the same dimension as the word vectors. For internal nodes, this vector is a phrase-level representation, while at leaf nodes it is the word vector x_w mapped into the hidden space. Unlike in constituency trees where all words reside at the leaf level, internal nodes of dependency trees are associated with words. Thus, the DT-RNN has to combine the current node's word vector with its children's hidden vectors to form h_n. This process continues recursively up to the root, which represents the entire sentence.

We associate a separate $d \times d$ matrix W_r with each dependency relation r in our dataset and learn these matrices during training.[1] Syntactically untying these matrices improves com-

[1] We had 46 unique dependency relations in our quiz bowl dataset.

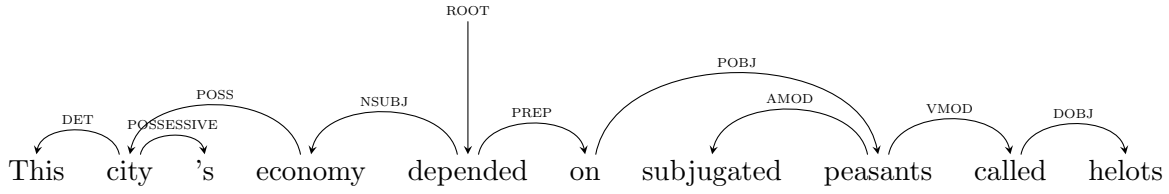


Figure 2: Dependency parse of a sentence from a question about Sparta.

positionality over the standard RNN model by taking into account relation identity along with tree structure. We include an additional $d \times d$ matrix, W_v, to incorporate the word vector x_w at a node into the node vector h_n.

Given a parse tree (Figure 2), we first compute leaf representations. For example, the hidden representation h_{helots} is

$$h_{\text{helots}} = f(W_v \cdot x_{\text{helots}} + b), \quad (1)$$

where f is a non-linear activation function such as tanh and b is a bias term. Once all leaves are finished, we move to interior nodes with already processed children. Continuing from "helots" to its parent, "called", we compute

$$h_{\text{called}} = f(W_{\text{DOBJ}} \cdot h_{\text{helots}} + W_v \cdot x_{\text{called}} + b). \quad (2)$$

We repeat this process up to the root, which is

$$h_{\text{depended}} = f(W_{\text{NSUBJ}} \cdot h_{\text{economy}} + W_{\text{PREP}} \cdot h_{\text{on}} + W_v \cdot x_{\text{depended}} + b). \quad (3)$$

The composition equation for any node n with children $K(n)$ and word vector x_w is $h_n =$

$$f(W_v \cdot x_w + b + \sum_{k \in K(n)} W_{R(n,k)} \cdot h_k), \quad (4)$$

where $R(n, k)$ is the dependency relation between node n and child node k.

3.2 Training

Our goal is to map questions to their corresponding answer entities. Because there are a limited number of possible answers, we can view this as a multi-class classification task. While a softmax layer over every node in the tree could predict answers (Socher et al., 2011; Iyyer et al., 2014), this method overlooks that most answers are themselves words (features) in other questions (e.g., a question on *World War II* might mention the *Battle of the Bulge* and vice versa). Thus, word vectors associated with such answers can be trained in the same vector space as question text,[2] enabling us to model relationships between answers instead of assuming incorrectly that all answers are independent.

To take advantage of this observation, we depart from Socher et al. (2014) by training both the answers and questions jointly in a single model, rather than training each separately and holding embeddings fixed during DT-RNN training. This method cannot be applied to the multimodal text-to-image mapping problem because text captions by definition are made up of words and thus cannot include images; in our case, however, question text can and frequently does include answer text.

Intuitively, we want to encourage the vectors of question sentences to be near their correct answers and far away from incorrect answers. We accomplish this goal by using a contrastive max-margin objective function described below. While we are not interested in obtaining a ranked list of answers,[3] we observe better performance by adding the weighted approximate-rank pairwise (WARP) loss proposed in Weston et al. (2011) to our objective function.

Given a sentence paired with its correct answer c, we randomly select j incorrect answers from the set of all incorrect answers and denote this subset as Z. Since c is part of the vocabulary, it has a vector $x_c \in W_e$. An incorrect answer $z \in Z$ is also associated with a vector $x_z \in W_e$. We define S to be the set of all nodes in the sentence's dependency tree, where an individual node $s \in S$ is associated with the

[2] Of course, questions never contain their own answer as part of the text.

[3] In quiz bowl, all wrong guesses are equally detrimental to a team's score, no matter how "close" a guess is to the correct answer.

hidden vector h_s. The error for the sentence is

$$C(S,\theta) = \sum_{s\in S}\sum_{z\in Z} L(rank(c,s,Z))\max(0, 1 - x_c \cdot h_s + x_z \cdot h_s), \quad (5)$$

where the function $rank(c,s,Z)$ provides the rank of correct answer c with respect to the incorrect answers Z. We transform this rank into a loss function[4] shown by Usunier et al. (2009) to optimize the top of the ranked list, $L(r) = \sum_{i=1}^{r} 1/i$.

Since $rank(c,s,Z)$ is expensive to compute, we approximate it by randomly sampling K incorrect answers until a violation is observed ($x_c \cdot h_s < 1 + x_z \cdot h_s$) and set $rank(c,s,Z) = (|Z|-1)/K$, as in previous work (Weston et al., 2011; Hermann et al., 2014). The model minimizes the sum of the error over all sentences T normalized by the number of nodes N in the training set,

$$J(\theta) = \frac{1}{N}\sum_{t\in T} C(t,\theta). \quad (6)$$

The parameters $\theta = (W_{r\in R}, W_v, W_e, b)$, where R represents all dependency relations in the data, are optimized using AdaGrad(Duchi et al., 2011).[5] In Section 4 we compare performance to an identical model (FIXED-QANTA) that excludes answer vectors from W_e and show that training them as part of θ produces significantly better results.

The gradient of the objective function,

$$\frac{\partial C}{\partial \theta} = \frac{1}{N}\sum_{t\in T}\frac{\partial J(t)}{\partial \theta}, \quad (7)$$

is computed using backpropagation through structure (Goller and Kuchler, 1996).

3.3 From Sentences to Questions

The model we have just described considers each sentence in a quiz bowl question independently. However, previously-heard sentences within the same question contain useful information that we do not want our model to ignore. While past work on RNN models have been restricted to the sentential and sub-sentential levels, we show that sentence-level representations can be easily combined to generate useful representations at the larger paragraph level.

The simplest and best[6] aggregation method is just to average the representations of each sentence seen so far in a particular question. As we show in Section 4, this method is very powerful and performs better than most of our baselines. We call this averaged DT-RNN model QANTA: a question answering neural network with trans-sentential averaging.

4 Experiments

We compare the performance of QANTA against multiple strong baselines on two datasets. QANTA outperforms all baselines trained only on question text and improves an information retrieval model trained on all of Wikipedia. QANTA requires that an input sentence describes an entity without mentioning that entity, a constraint that is not followed by Wikipedia sentences.[7] While IR methods can operate over Wikipedia text with no issues, we show that the representations learned by QANTA over just a dataset of question-answer pairs can significantly improve the performance of IR systems.

4.1 Datasets

We evaluate our algorithms on a corpus of over 100,000 question/answer pairs from two different sources. First, we expand the dataset used in Boyd-Graber et al. (2012) with publically-available questions from quiz bowl tournaments held after that work was published. This gives us 46,842 questions in fourteen different categories. To this dataset we add 65,212 questions from NAQT, an organization that runs quiz bowl tournaments and generously shared with us all of their questions from 1998–2013.

[4] Our experiments show that adding this loss term to the objective function not only increases performance but also speeds up convergence

[5] We set the initial learning rate $\eta = 0.05$ and reset the squared gradient sum to zero every five epochs.

[6] We experimented with weighting earlier sentences less than later ones in the average as well as learning an additional RNN on top of the sentence-level representations. In the former case, we observed no improvements over a uniform average, while in the latter case the model overfit even with strong regularization.

[7] We tried transforming Wikipedia sentences into quiz bowl sentences by replacing answer mentions with appropriate descriptors (e.g., "Joseph Heller" with "this author"), but the resulting sentences suffered from a variety of grammatical issues and did not help the final result.

Because some categories contain substantially fewer questions than others (e.g., astronomy has only 331 questions), we consider only literature and history questions, as these two categories account for more than 40% of the corpus. This leaves us with 21,041 history questions and 22,956 literature questions.

4.1.1 Data Preparation

To make this problem feasible, we only consider a limited set of the most popular quiz bowl answers. Before we filter out uncommon answers, we first need to map all raw answer strings to a canonical set to get around formatting and redundancy issues. Most quiz bowl answers are written to provide as much information about the entity as possible. For example, the following is the raw answer text of a question on the Chinese leader Sun Yat-sen: *Sun Yat-sen; or Sun Yixian; or Sun Wen; or Sun Deming; or Nakayama Sho; or Nagao Takano*. Quiz bowl writers vary in how many alternate acceptable answers they provide, which makes it tricky to strip superfluous information from the answers using rule-based approaches.

Instead, we use Whoosh,[8] an information retrieval library, to generate features in an active learning classifier that matches existing answer strings to Wikipedia titles. If we are unable to find a match with a high enough confidence score, we throw the question out of our dataset. After this standardization process and manual vetting of the resulting output, we can use the Wikipedia page titles as training labels for the DT-RNN and baseline models.[9]

65.6% of answers only occur once or twice in the corpus. We filter out all answers that do not occur at least six times, which leaves us with 451 history answers and 595 literature answers that occur on average twelve times in the corpus. These pruning steps result in 4,460 usable history questions and 5,685 literature questions. While ideally we would have used all answers, our model benefits from many training examples per answer to learn meaningful representations; this issue can possibly be addressed with techniques from zero shot learning (Palatucci et al., 2009; Pasupat and Liang, 2014), which we leave to future work.

We apply basic named entity recognition (NER) by replacing all occurrences of answers in the question text with single entities (e.g., *Ernest Hemingway* becomes *Ernest_Hemingway*). While we experimented with more advanced NER systems to detect non-answer entities, they could not handle multi-word named entities like the book *Love in the Time of Cholera* (title case) or battle names (e.g., *Battle of Midway*). A simple search/replace on all answers in our corpus works better for multi-word entities.

The preprocessed data are split into folds by tournament. We choose the past two national tournaments[10] as our test set as well as questions previously answered by players in Boyd-Graber et al. (2012) and assign all other questions to train and dev sets. History results are reported on a training set of 3,761 questions with 14,217 sentences and a test set of 699 questions with 2,768 sentences. Literature results are reported on a training set of 4,777 questions with 17,972 sentences and a test set of 908 questions with 3,577 sentences.

Finally, we initialize the word embedding matrix W_e with word2vec (Mikolov et al., 2013) trained on the preprocessed question text in our training set.[11] We use the hierarchical skip-gram model setting with a window size of five words.

4.2 Baselines

We pit QANTA against two types of baselines: bag of words models, which enable comparison to a standard NLP baseline, and information retrieval models, which allow us to compare against traditional question answering techniques.

BOW The BOW baseline is a logistic regression classifier trained on binary unigram indicators.[12] This simple discriminative model is an improvement over the generative quiz bowl answering model of Boyd-Graber et al. (2012).

[8] `https://pypi.python.org/pypi/Whoosh/`

[9] Code and non-NAQT data available at `http://cs.umd.edu/~miyyer/qblearn`.

[10] The tournaments were selected because NAQT does not reuse any questions or clues within these tournaments.

[11] Out-of-vocabulary words from the test set are initialized randomly.

[12] Raw word counts, frequencies, and TF-IDF weighted features did not increase performance, nor did adding bigrams to the feature set (possibly because multi-word named entities are already collapsed into single words).

BOW-DT The BOW-DT baseline is identical to BOW except we augment the feature set with dependency relation indicators. We include this baseline to isolate the effects of the dependency tree structure from our compositional model.

IR-QB The IR-QB baseline maps questions to answers using the state-of-the-art Whoosh IR engine. The knowledge base for IR-QB consists of "pages" associated with each answer, where each page is the union of training question text for that answer. Given a partial question, the text is first preprocessed using a query language similar to that of Apache Lucene. This processed query is then matched to pages uses BM-25 term weighting, and the top-ranked page is considered to be the model's guess. We also incorporate fuzzy queries to catch misspellings and plurals and use Whoosh's built-in query expansion functionality to add related keywords to our queries. **IR-WIKI** The IR-WIKI model is identical to the IR-QB model except that each "page" in its knowledge base also includes all text from the associated answer's Wikipedia article. Since all other baselines and DT-RNN models operate only on the question text, this is not a valid comparison, but we offer it to show that we can improve even this strong model using QANTA.

4.3 DT-RNN Configurations

For all DT-RNN models the vector dimension d and the number of wrong answers per node j is set to 100. All model parameters other than W_e are randomly initialized. The non-linearity f is the normalized tanh function,[13]

$$f(v) = \frac{\tanh(v)}{\|\tanh(v)\|}. \quad (8)$$

QANTA is our DT-RNN model with feature averaging across previously-seen sentences in a question. To obtain the final answer prediction given a partial question, we first generate a feature representation for each sentence within that partial question. This representation is computed by concatenating together the word embeddings and hidden representations averaged over all nodes in the tree as well as the root node's hidden vector. Finally, we send the average of all of the individual sentence features[14] as input to a logistic regression classifier for answer prediction.

FIXED-QANTA uses the same DT-RNN configuration as QANTA except the answer vectors are kept constant as in the text-to-image model.

4.4 Human Comparison

Previous work provides human answers (Boyd-Graber et al., 2012) for quiz bowl questions. We use human records for 1,201 history guesses and 1,715 literature guesses from twenty-two of the quiz bowl players who answered the most questions.[15]

The standard scoring system for quiz bowl is **10** points for a correct guess and **-5** points for an incorrect guess. We use this metric to compute a total score for each human. To obtain the corresponding score for our model, we force it to imitate each human's guessing policy. For example, Figure 3 shows a human answering in the middle of the second sentence. Since our model only considers sentence-level increments, we compare the model's prediction after the first sentence to the human prediction, which means our model is privy to less information than humans.

The resulting distributions are shown in Figure 4—our model does better than the average player on history questions, tying or defeating sixteen of the twenty-two players, but it does worse on literature questions, where it only ties or defeats eight players. The figure indicates that literature questions are harder than history questions for our model, which is corroborated by the experimental results discussed in the next section.

5 Discussion

In this section, we examine why QANTA improves over our baselines by giving examples of questions that are incorrectly classified by all baselines but correctly classified by QANTA. We also take a close look at some sentences that all models fail to answer correctly. Finally, we visualize the answer space learned by QANTA.

[13]The standard tanh function produced heavy saturation at higher levels of the trees, and corrective weighting as in Socher et al. (2014) hurt our model because named entities that occur as leaves are often more important than non-terminal phrases.

[14]Initial experiments with L_2 regularization hurt performance on a validation set.

[15]Participants were skilled quiz bowl players and are not representative of the general population.

	History			Literature		
Model	Pos 1	Pos 2	Full	Pos 1	Pos 2	Full
BOW	27.5	51.3	53.1	19.3	43.4	46.7
BOW-DT	35.4	57.7	60.2	24.4	51.8	55.7
IR-QB	37.5	65.9	71.4	27.4	54.0	61.9
FIXED-QANTA	38.3	64.4	66.2	28.9	57.7	62.3
QANTA	**47.1**	**72.1**	**73.7**	**36.4**	**68.2**	**69.1**
IR-WIKI	53.7	76.6	77.5	41.8	74.0	73.3
QANTA+IR-WIKI	**59.8**	**81.8**	**82.3**	**44.7**	**78.7**	**76.6**

Table 1: Accuracy for history and literature at the first two sentence positions of each question and the full question. The top half of the table compares models trained on questions only, while the IR models in the bottom half have access to Wikipedia. QANTA outperforms all baselines that are restricted to just the question data, and it substantially improves an IR model with access to Wikipedia despite being trained on much less data.

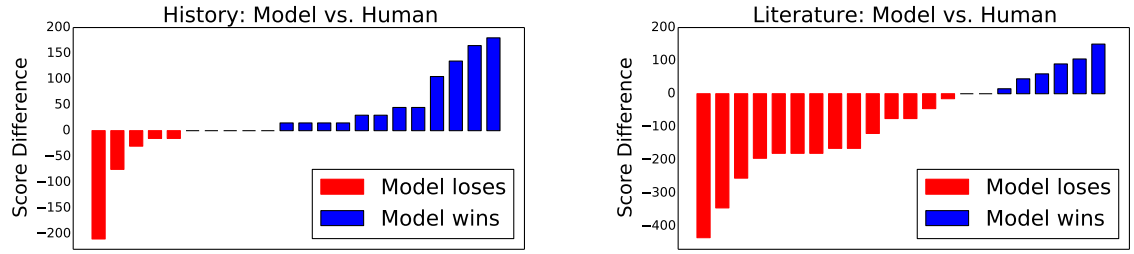


Figure 4: Comparisons of QANTA+IR-WIKI to human quiz bowl players. Each bar represents an individual human, and the bar height corresponds to the difference between the model score and the human score. Bars are ordered by human skill. Red bars indicate that the human is winning, while blue bars indicate that the model is winning. QANTA+IR-WIKI outperforms most humans on history questions but fails to defeat the "average" human on literature questions.

A minor character in this play can be summoned by a bell that does not always work; that character also doesn't have eyelids. Near the end, a woman who drowned her illegitimate child attempts to stab another woman in the Second Empire-style ◇ room in which the entire play takes place. For 10 points, Estelle and Ines are characters in which existentialist play in which Garcin claims "Hell is other people", written by Jean-Paul Sartre?

Figure 3: A question on the play "No Exit" with human buzz position marked as ◇. Since the buzz occurs in the middle of the second sentence, our model is only allowed to see the first sentence.

5.1 Experimental Results

Table 1 shows that when bag of words and information retrieval methods are restricted to question data, they perform significantly worse than QANTA on early sentence positions. The performance of BOW-DT indicates that while the dependency tree structure helps by itself, the compositional distributed representations learned by QANTA are more useful. The significant improvement when we train answers as part of our vocabulary (see Section 3.2) indicates that our model uses answer occurrences within question text to learn a more informative vector space.

The disparity between IR-QB and IR-WIKI indicates that the information retrieval models need lots of external data to work well at all sentence positions. IR-WIKI performs better than other models because Wikipedia contains many more sentences that partially match specific words or phrases found in early clues than the question training set. In particular, it is impossible for all other models to answer clues in the test set that have no semantically similar

or equivalent analogues in the training question data. With that said, IR methods can also operate over data that does not follow the special constraints of quiz bowl questions (e.g., every sentence uniquely identifies the answer, answers don't appear in their corresponding questions), which QANTA cannot handle. By combining QANTA and IR-WIKI, we are able to leverage access to huge knowledge bases along with deep compositional representations, giving us the best of both worlds.

5.2 Where the Attribute Space Helps Answer Questions

We look closely at the first sentence from a literature question about the author Thomas Mann: "He left unfinished a novel whose title character forges his father's signature to get out of school and avoids the draft by feigning desire to join".

All baselines, including IR-WIKI, are unable to predict the correct answer given only this sentence. However, QANTA makes the correct prediction. The sentence contains no named entities, which makes it almost impossible for bag of words or string matching algorithms to predict correctly. Figure 6 shows that the plot description associated with the "novel" node is strongly indicative of the answer. The five highest-scored answers are all male authors,[16] which shows that our model is able to learn the answer type without any hand-crafted rules.

Our next example, the first sentence in Table 2, is from the first position of a question on John Quincy Adams, which is correctly answered by only QANTA. The bag of words model guesses Henry Clay, who was also a Secretary of State in the nineteenth century and helped John Quincy Adams get elected to the presidency in a "corrupt bargain". However, the model can reason that while Henry Clay was active at the same time and involved in the same political problems of the era, he did not represent the Amistad slaves, nor did he negotiate the Treaty of Ghent.

5.3 Where all Models Struggle

Quiz bowl questions are intentionally written to make players work to get the answer, especially at early sentence positions. Our model fails to answer correctly more than half the time after hearing only the first sentence. We examine some examples to see if there are any patterns to what makes a question "hard" for machine learning models.

Consider this question about the Italian explorer John Cabot: "As a young man, this native of Genoa disguised himself as a Muslim to make a pilgrimage to Mecca".

While it is obvious to human readers that the man described in this sentence is not actually a Muslim, QANTA has to accurately model the verb *disguised* to make that inference. We show the score plot of this sentence in Figure 7. The model, after presumably seeing many instances of *muslim* and *mecca* associated with Mughal emperors, is unable to prevent this information from propagating up to the root node. On the bright side, our model is able to learn that the question is expecting a human answer rather than non-human entities like the Umayyad Caliphate.

More examples of impressive answers by QANTA as well as incorrect guesses by all systems are shown in Table 2.

5.4 Examining the Attribute Space

Figure 5 shows a t-SNE visualization (Van der Maaten and Hinton, 2008) of the 451 answers in our history dataset. The vector space is divided into six general clusters, and we focus in particular on the US presidents. Zooming in on this section reveals temporal clustering: presidents who were in office during the same timeframe occur closer together. This observation shows that QANTA is capable of learning attributes of entities during training.

6 Related Work

There are two threads of related work relevant to this paper. First, we discuss previous applications of compositional vector models to related NLP tasks. Then, we examine existing work on factoid question-answering and review the similarities and differences between these tasks and the game of quiz bowl.

6.1 Recursive Neural Networks for NLP

The principle of *semantic composition* states that the meaning of a phrase can be derived

[16] three of whom who also have well-known unfinished novels

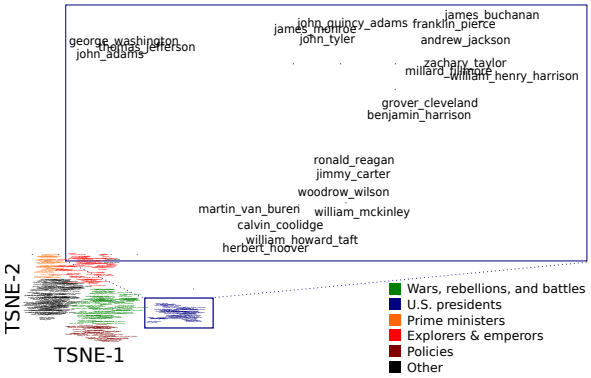


Figure 5: t-SNE 2-D projections of 451 answer vectors divided into six major clusters. The blue cluster is predominantly populated by U.S. presidents. The zoomed plot reveals temporal clustering among the presidents based on the years they spent in office.

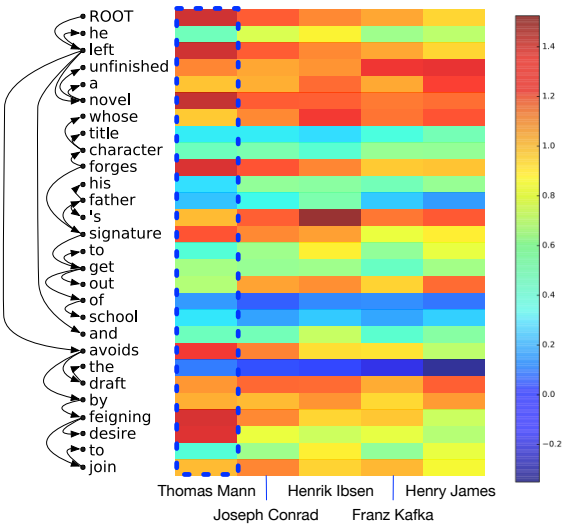


Figure 6: A question on the German novelist Thomas Mann that contains no named entities, along with the five top answers as scored by QANTA. Each cell in the heatmap corresponds to the score (inner product) between a node in the parse tree and the given answer, and the dependency parse of the sentence is shown on the left. All of our baselines, including IR-WIKI, are wrong, while QANTA uses the plot description to make a correct guess.

from the meaning of the words that it contains as well as the syntax that glues those words together. Many computational models of compositionality focus on learning vector spaces (Zanzotto et al., 2010; Erk, 2012; Grefenstette et al., 2013; Yessenalina and Cardie, 2011). Recent approaches towards modeling compositional vector spaces with neural networks have been successful, although simpler functions have been proposed for short phrases (Mitchell and Lapata, 2008).

Recursive neural networks have achieved state-of-the-art performance in sentiment analysis and parsing (Socher et al., 2013c; Hermann and Blunsom, 2013; Socher et al., 2013a). RNNs have not been previously used for learning attribute spaces as we do here, although recursive tensor networks were unsuccessfully applied to a knowledge base completion task (Socher et al., 2013b). More relevant to this work are the dialogue analysis model proposed by Kalchbrenner & Blunsom (2013) and the paragraph vector model described in Le and Mikolov (2014), both of which are able to generate distributed representations of paragraphs. Here we present a simpler approach where a single model is able to learn complex sentence representations and average them across paragraphs.

6.2 Factoid Question-Answering

Factoid question answering is often functionally equivalent to information retrieval. Given a knowledge base and a query, the goal is to return the answer. Many approaches to this problem rely on hand-crafted pattern matching and answer-type classification to narrow down the search space (Shen, 2007; Bilotti et al., 2010; Wang, 2006). More recent factoid QA systems incorporate the web and social media into their retrieval systems (Bian et al., 2008). In contrast to these approaches, we place the burden of learning answer types and patterns on the model.

7 Future Work

While we have shown that DT-RNNs are effective models for quiz bowl question answering, other factoid QA tasks are more challenging. Questions like *what does the* AARP *stand for?* from TREC QA data require additional infrastructure. A more apt comparison would be to IBM's proprietary Watson system (Lally et al., 2012) for Jeopardy, which is limited to single sentences, or to models trained on Yago (Hoffart et al., 2013).

We would also like to fairly compare QANTA

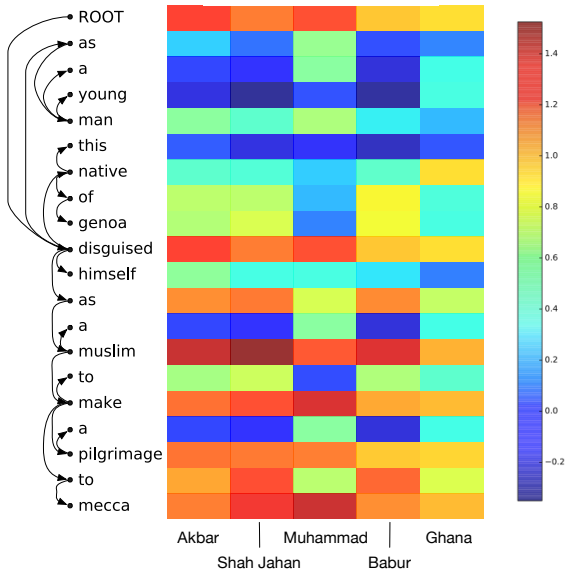


Figure 7: An extremely misleading question about John Cabot, at least to computer models. The words *muslim* and *mecca* lead to three Mughal emperors in the top five guesses from QANTA; other models are similarly led awry.

Q	he also successfully represented the amistad slaves and negotiated the treaty of ghent and the annexation of florida from spain during his stint as secretary of state under james monroe
A	**john quincy adams**, henry clay, andrew jackson
Q	this work refers to people who fell on their knees in hopeless cathedrals and who jumped off the brooklyn bridge
A	**howl**, the tempest, paradise lost
Q	despite the fact that twenty six martyrs were crucified here in the late sixteenth century it remained the center of christianity in its country
A	**nagasaki**, guadalcanal, ethiopia
Q	this novel parodies freudianism in a chapter about the protagonist 's dream of holding a live fish in his hands
A	billy budd, the ambassadors, all my sons
Q	a contemporary of elizabeth i he came to power two years before her and died two years later
A	grover cleveland, benjamin harrison, henry cabot lodge

Table 2: Five example sentences occuring at the first sentence position along with their top three answers as scored by QANTA; correct answers are marked with blue and wrong answers are marked with red. QANTA gets the first three correct, unlike all other baselines. The last two questions are too difficult for all of our models, requiring external knowledge (e.g., *Freudianism*) and temporal reasoning.

with IR-WIKI. A promising avenue for future work would be to incorporate Wikipedia data into QANTA by transforming sentences to look like quiz bowl questions (Wang et al., 2007) and to select relevant sentences, as not every sentence in a Wikipedia article directly describes its subject. Syntax-specific annotation (Sayeed et al., 2012) may help in this regard.

Finally, we could adapt the attribute space learned by the DT-RNN to use information from knowledge bases and to aid in knowledge base completion. Having learned many facts about entities that occur in question text, a DT-RNN could add new facts to a knowledge base or check existing relationships.

8 Conclusion

We present QANTA, a dependency-tree recursive neural network for factoid question answering that outperforms bag of words and information retrieval baselines. Our model improves upon a contrastive max-margin objective function from previous work to dynamically update answer vectors during training with a single model. Finally, we show that sentence-level representations can be easily and effectively combined to generate paragraph-level representations with more predictive power than those of the individual sentences.

Acknowledgments

We thank the anonymous reviewers, Stephanie Hwa, Bert Huang, and He He for their insightful comments. We thank Sharad Vikram, R. Hentzel, and the members of NAQT for providing our data. This work was supported by NSF Grant IIS-1320538. Boyd-Graber is also supported by NSF Grant CCF-1018625. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *JMLR*.

Jiang Bian, Yandong Liu, Eugene Agichtein, and Hongyuan Zha. 2008. Finding the right facts in the crowd: factoid question answering over social media. In *WWW*.

Matthew W. Bilotti, Jonathan Elsas, Jaime Carbonell, and Eric Nyberg. 2010. Rank learning for factoid question answering with linguistic and semantic constraints. In *CIKM*.

Jordan Boyd-Graber, Brianna Satinoff, He He, and Hal Daume III. 2012. Besting the quiz master: Crowdsourcing incremental classification games. In *EMNLP*.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 999999:2121–2159.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1.

Edward Grefenstette, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. 2013. Multi-step regression learning for compositional distributional semantics. *CoRR*.

Karl Moritz Hermann and Phil Blunsom. 2013. The Role of Syntax in Vector Space Models of Compositional Semantics. In *ACL*.

Karl Moritz Hermann, Edward Grefenstette, and Phil Blunsom. 2013. "not not bad" is not "bad": A distributional account of negation. *Proceedings of the ACL Workshop on Continuous Vector Space Models and their Compositionality*.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *ACL*.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *Proceedings of the 2013 Workshop on Continuous Vector Space Models and their Compositionality*.

Adam Lally, John M Prager, Michael C McCord, BK Boguraev, Siddharth Patwardhan, James Fan, Paul Fodor, and Jennifer Chu-Carroll. 2012. Question analysis: How watson reads a clue. *IBM Journal of Research and Development*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents.

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

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *ACL*.

Mark Palatucci, Dean Pomerleau, Geoffrey E. Hinton, and Tom M. Mitchell. 2009. Zero-shot learning with semantic output codes. In *NIPS*.

P. Pasupat and P. Liang. 2014. Zero-shot entity extraction from web pages. In *ACL*.

Asad B Sayeed, Jordan Boyd-Graber, Bryan Rusk, and Amy Weinberg. 2012. Grammatical structures for word-level sentiment detection. In *NAACL*.

Dan Shen. 2007. Using semantic role to improve question answering. In *EMNLP*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *EMNLP*.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013a. Parsing With Compositional Vector Grammars. In *ACL*.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013b. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In *NIPS*.

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

Richard Socher, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *TACL*.

Nicolas Usunier, David Buffoni, and Patrick Gallinari. 2009. Ranking with ordered weighted pairwise classification. In *ICML*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *JMLR*.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *EMNLP*.

Mengqiu Wang. 2006. A survey of answer extraction techniques in factoid question answering. *Computational Linguistics*, 1(1).

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *EMNLP*.

Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *COLT*.

Joint Relational Embeddings for Knowledge-based Question Answering

Min-Chul Yang[†] **Nan Duan**[‡] **Ming Zhou**[‡] **Hae-Chang Rim**[†]
[†]Dept. of Computer & Radio Comms. Engineering, Korea University, Seoul, South Korea
[‡]Microsoft Research Asia, Beijing, China
mcyang@nlp.korea.ac.kr
{nanduan, mingzhou}@microsoft.com
rim@nlp.korea.ac.kr

Abstract

Transforming a natural language (NL) question into a corresponding logical form (LF) is central to the knowledge-based question answering (KB-QA) task. Unlike most previous methods that achieve this goal based on mappings between lexicalized phrases and logical predicates, this paper goes one step further and proposes a novel embedding-based approach that maps NL-questions into LFs for KB-QA by leveraging semantic associations between lexical representations and KB-properties in the latent space. Experimental results demonstrate that our proposed method outperforms three KB-QA baseline methods on two publicly released QA data sets.

1 Introduction

Knowledge-based question answering (KB-QA) involves answering questions posed in natural language (NL) using existing knowledge bases (KBs). As most KBs are structured databases, how to transform the input question into its corresponding structured query for KB (KB-query) as a logical form (LF), also known as semantic parsing, is the central task for KB-QA systems. Previous works (Mooney, 2007; Liang et al., 2011; Cai and Yates, 2013; Fader et al., 2013; Berant et al., 2013; Bao et al., 2014) usually leveraged mappings between NL phrases and logical predicates as lexical triggers to perform transformation tasks in semantic parsing, but they had to deal with two limitations: (i) as the meaning of a logical predicate often has different natural language expression (NLE) forms, the lexical triggers extracted for a predicate may at times are limited in size; (ii) entities detected by the named entity recognition (NER) component will be used to compose the logical forms together with the logical predicates, so their types should be consistent with the predicates as well. However, most NER components used in existing KB-QA systems are independent from the NLE-to-predicate mapping procedure.

We present a novel embedding-based KB-QA method that takes all the aforementioned limitations into account, and maps NLE-to-entity and NLE-to-predicate simultaneously using simple vector operations for structured query construction. First, low-dimensional embeddings of n-grams, entity types, and predicates are jointly learned from an existing knowledge base and from entries $<entity_{subj}$, *NL relation phrase,* $entity_{obj}>$ that are mined from NL texts labeled as KB-properties with weak supervision. Each such entry corresponds to an NL expression of a triple $<entity_{subj}$, *predicate,* $entity_{obj}>$ in the KB. These embeddings are used to measure the semantic associations between lexical phrases and two properties of the KB, entity type and logical predicate. Next, given an NL-question, all possible structured queries as candidate LFs are generated and then they are ranked by the similarity between the embeddings of observed features (n-grams) in the NL-question and the embeddings of logical features in the structured queries. Last, answers are retrieved from the KB using the selected LFs.

The contributions of this work are two-fold: (1) as a smoothing technique, the low-dimensional embeddings can alleviate the coverage issues of lexical triggers; (2) our joint approach integrates entity span selection and predicate mapping tasks for KB-QA. For this we built independent entity embeddings as the additional component, solving the entity disambiguation problem.

2 Related Work

Supervised semantic parsers (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Mooney, 2007) heavily rely on the *<sentence, semantic an-*

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 645–650,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

notation> pairs for lexical trigger extraction and model training. Due to the data annotation requirement, such methods are usually restricted to specific domains, and struggle with the coverage issue caused by the limited size of lexical triggers.

Studies on weakly supervised semantic parsers have tried to reduce the amount of human supervision by using question-answer pairs (Liang et al., 2011) or distant supervision (Krishnamurthy and Mitchell, 2012) instead of full semantic annotations. Still, for KB-QA, the question of how to leverage KB-properties and analyze the question structures remains.

Bordes et al. (2012) and Weston et al. (2013) designed embedding models that connect free texts with KBs using the relational learning method (Weston et al., 2010). Their inputs are often statement sentences which include subject and object entities for a given predicate, whereas NL-questions lack either a subject or object entity that is the potential answer. Hence, we can only use the information of a subject or object entity, which leads to a different training instance generation procedure and a different training criterion.

Recently, researchers have developed open domain systems based on large scale KBs such as FREEBASE[1] (Cai and Yates, 2013; Fader et al., 2013; Berant et al., 2013; Kwiatkowski et al., 2013; Bao et al., 2014; Berant and Liang, 2014; Yao and Van Durme, 2014). Their semantic parsers for Open QA are unified formal and scalable: they enable the NL-question to be mapped into the appropriate logical form. Our method obtains similar logical forms, but using only low-dimensional embeddings of n-grams, entity types, and predicates learned from texts and KB.

3 Setup

3.1 Relational Components for KB-QA

Our method learns semantic mappings between NLEs and the KB[2] based on the paired relationships of the following three components: $\mathcal{C}$ denotes a set of bag-of-words (or n-grams) as context features (c) for NLEs that are the lexical representations of a logical predicate (p) in KB; $\mathcal{T}$ denotes a set of entity types (t) in KB and each type can be used as the abstract expression of a subject entity (s) that occurs in the input question; $\mathcal{P}$ denotes a set of logical predicates (p) in KB, each of which is the canonical form of different NLEs sharing an identical meaning (bag-of-words; c).

Based on the components defined above, the paired relationships are described as follows: $\mathcal{T}$-$\mathcal{P}$ can investigate the relationship between subject entity and logical predicate, as object entity is always missing in KB-QA; $\mathcal{C}$-$\mathcal{T}$ can scrutinize subject entity's attributes for the entity span selection such as its positional information and relevant entity types to the given context, which may solve the entity disambiguation problem in KB-QA; $\mathcal{C}$-$\mathcal{P}$ can leverage the semantic overlap between question contexts (n-gram features) and logical predicates, which is important for mapping NL-questions to their corresponding predicates.

3.2 NLE-KB Pair Extraction

This section describes how we extract the semantic associated pairs of NLE-entries and KB-triples to learn the relational embeddings (Section 4.1).

<Relation Mention, Predicate> Pair ($\mathcal{MP}$) Each relation mention denotes a lexical phrase of an existing KB-predicate. Following information extraction methods, such as PATTY (Nakashole et al., 2012), we extracted the *<relation mention, logical predicate>* pairs from English WIKIPEDIA[3], which is closely connected to our KB, as follows: Given a KB-triple *<entity$_{subj}$, logical predicate, entity$_{obj}$>*, we extracted NLE-entries *<entity$_{subj}$, relation mention, entity$_{obj}$>* where *relation mention* is the shortest path between *entity$_{subj}$* and *entity$_{obj}$* in the dependency tree of sentences. The assumption is that any *relation mention* (m) in the NLE-entry containing such entity pairs that occurred in the KB-triple is likely to express the *predicate* (p) of that triple.

With obtaining high-quality $\mathcal{MP}$ pairs, we kept only relation mentions that were highly associated with a predicate measured by the scoring function:

$$S(m,p) = \mathsf{PMI}(e_m; e_p) + \mathsf{PMI}(u_m; u_p) \quad (1)$$

where e_x is the set of total pairs of both-side entities of entry x (m or p) and u_x is the set of unique (distinct) pairs of both-side entities of entry x. In this case, the both-side entities indicate *entity$_{subj}$* and *entity$_{obj}$*. For a frequency-based probability, $\mathsf{PMI}(x;y) = \log \frac{P(x,y)}{P(x)P(y)}$

[1] http://www.freebase.com

[2] For this paper, we used a large scale knowledge base that contains 2.3B entities, 5.5K predicates, and 18B assertions. A 16-machine cluster was used to host and serve the whole data.

[3] http://en.wikipedia.org/

(Church and Hanks, 1990) can be re-written as $\mathsf{PMI}(x;y) = \log \frac{|x \bigcap y| \cdot C}{|x| \cdot |y|}$, where C denotes the total number of items shown in the corpus. The function is partially derived from the *support* score (Gerber and Ngonga Ngomo, 2011), but we focus on the correlation of shared entity pairs between relation mentions and predicates using the PMI computation.

<Question Pattern, Predicate> Pair ($\mathcal{QP}$) Since WIKIPEDIA articles have no information to leverage interrogative features which highly depend on the object entity (answer), it is difficult to distinguish some questions that are composed of only different *5W1H* words, e.g., {*When*|*Where*} *was Barack Obama born?* Hence, we used the method of collecting question patterns with human labeled predicates that are restricted by the set of predicates used in $\mathcal{MP}$ (Bao et al., 2014).

4 Embedding-based KB-QA

Our task is as follows. First, our model learns the semantic associations of $\mathcal{C}$-$\mathcal{T}$, $\mathcal{C}$-$\mathcal{P}$, and $\mathcal{T}$-$\mathcal{P}$ (Section 3.1) based on NLE-KB pairs (Section 3.2), and then predicts the semantic-related KB-query which can directly find the answer to a given NL-question.

For our feature space, given an NLE-KB pair, the NLE (*relation mention* in $\mathcal{MP}$ or *question pattern* in $\mathcal{QP}$) is decomposed into n-gram features: $\mathbb{C} = \{c \mid c \textit{ is a segment of NLE}\}$, and the KB-properties are represented by entity type t of *entity$_{subj}$* and predicate p. Then we can obtain a training triplet $w = [\mathbb{C}, t, p]$. Each feature ($c \in \mathcal{C}, t \in \mathcal{T}, p \in \mathcal{P}$) is encoded in the distributed representation which is n-dimensional embedding vectors ($\mathbb{E}^n$): $\forall x,\ x \overset{encode}{\Rightarrow} \mathbb{E}(x) \in \mathbb{E}^n$.

All n-gram features ($\mathbb{C}$) for an NLE are merged into one embedding vector to help speed up the learning process: $\mathbb{E}(\mathbb{C}) = \sum_{c \in \mathbb{C}} \mathbb{E}(c)/|\mathbb{C}|$. This feature representation is inspired by previous work in embedding-based relation extraction (Weston et al., 2013), but differs in the following ways: (1) entity information is represented on a separate embedding, but its positional information remains as symbol ⟨entity⟩; (2) when the vectors are combined, we use the average of each index to normalize features.

For our joint relational approach, we focus on the set of paired relationships $\mathbb{R} = \{\mathbb{C}\text{-}t, \mathbb{C}\text{-}p, t\text{-}p\}$ that can be semantically leveraged. Formally, these features are embedded into the same latent space ($\mathbb{E}^n$) and their semantic similarities can be computed by a dot product operation:

$$Sim(a,b) = Sim(r_{ab}) = \mathbb{E}(a)^{\intercal}\mathbb{E}(b) \quad (2)$$

where r_{ab} denotes a paired relationship a-b (or (a, b)) in the above set $\mathbb{R}$. We believe that our joint relational learning can smooth the surface (lexical) features for semantic parsing using the aligned entity and predicate.

4.1 Joint Relational Embedding Learning

Our ranking-based relational learning is based on a ranking loss (Weston et al., 2010) that supports the idea that the similarity scores of observed pairs in the training set (*positive instances*) should be larger than those of any other pairs (*negative instances*):

$$\forall i, \forall y' \neq y_i,\ Sim(x_i, y_i) > 1 + Sim(x_i, y') \quad (3)$$

More precisely, for each triplet $w_i = [\mathbb{C}_i, t_i, p_i]$ obtained from an NLE-KB pair, the relationships $\mathbb{R}_i = \{\mathbb{C}_i\text{-}t_i,\ \mathbb{C}_i\text{-}p_i,\ t_i\text{-}p_i\}$ are trained under the soft ranking criterion, which conducts Stochastic Gradient Descent (SGD). We thus aim to minimize the following:

$$\forall i, \forall y' \neq y_i, \max(0, 1 - Sim(x_i, y_i) + Sim(x_i, y')) \quad (4)$$

Our learning strategy is as follows. First, we initialize embedding space $\mathbb{E}^n$ by randomly giving mean 0 and standard deviation $1/n$ to each vector. Then for each training triplet w_i, we select the negative pairs against positive pairs ($\mathbb{C}_i$-t_i, $\mathbb{C}_i$-p_i, and t_i-p_i) in the triplet. Last, we make a stochastic gradient step to minimize Equation 4 and update $\mathbb{E}^n$ at each step.

4.2 KB-QA using Embedding Models

Our goal for KB-QA is to translate a given NL-question to a KB-query with the form <*subject entity, predicate, ?*>, where *?* denotes the answer entity we are looking for. The decoding process consists of two stages. The first stage involves generating all possible KB-queries ($\mathcal{K}^q$) for an NL-question q. We first extract n-gram features ($\mathbb{C}^q$) from the NL-question q. Then for a KB-query k^q, we find all available entity types (t^q) of the identified subject entities (s^q) using the dictionary-based entity detection on the NL-question q (all of spans can be candidate entities), and assign all items of predicate set ($\mathcal{P}$) as the candidate predicates (p^q). Like the training triplets,

q	*where is the city of david?*
$\hat{k}(q)$	[The City of David, contained_by, ?]
$\mathbb{C}^q$	n-grams of "where is ⟨entity⟩ ?"
t^q	location
p^q	contained_by

Table 1: The corresponding KB-query $\hat{k}(q)$ for a NL-question q and its decoding triplet w^q.

we also represent the above features as the triplet form $w_i^q = [\mathbb{C}_i^q, t_i^q, p_i^q]$ which is directly linked to a KB-query $k_i^q = [s_i^q, p_i^q, ?]$. The second stage involves ranking candidate KB-queries based on the similarity scores between the following paired relationships from the triplet w_i^q: $\mathbb{R}_i^q = \{\mathbb{C}_i^q\text{-}t_i^q, \mathbb{C}_i^q\text{-}p_i^q, t_i^q\text{-}p_i^q\}$. Unlike in the training step, the similarities of $\mathbb{C}_i^q$-t_i^q and $\mathbb{C}_i^q$-p_i^q are computed by summation of all pairwise elements (each context embedding $\mathbb{E}(c)$, not $\mathbb{E}(\mathbb{C})$, with each paired $\mathbb{E}(t)$ or $\mathbb{E}(p)$) for a more precise measurement. Since similarites of $\mathbb{R}^q$ are calculated on different scales, we normalize each value using Z-score ($Z(x) = \frac{x-\mu}{\sigma}$) (Kreyszig, 1979). The final score is measured by:

$$Sim_{q2k}(q, k^q) = \sum_{r \in \mathbb{R}^q} Z(Sim(r)) \qquad (5)$$

Then, given any NL-question q, we can predict the corresponding KB-query $\hat{k}(q)$:

$$\hat{k}(q) = \arg\max_{k \in \mathcal{K}^q} Sim_{q2k}(q, k) \qquad (6)$$

Last, we can retrieve an answer from the KB using a structured query $\hat{k}(q)$. Table 1 shows an example of our decoding process.

Multi-related Question Some questions include two-subject entities, both of which are crucial to understanding the question. For the question *who plays gandalf in the lord of the rings?* Gandalf (character) and The Lord Of The Rings (film) are explicit entities that should be joined to a pair of the two entities (implicit entity). More precisely, the two entities can be combined into one concatenated entity (character-in-film) using our manual rule, which compares the possible pairs of entity types in the question with the list of pre-defined entity type pairs that can be merged into a concatenated entity. Our solution enables a multi-related question to be transformed to a single-related question which can be directly translated to a KB-query. Then, the two entity mentions are replaced with the symbol ⟨entity⟩ (who play ⟨entity⟩ in ⟨entity⟩ ?). We regard the result of this transformation as one of the candidate KB-queries in the decoding step.

	# Entries	Accuracy
$\mathcal{MP}$ pairs	291,585	89%
$\mathcal{QP}$ pairs	4,764	98%

Table 2: Statistics of NLE-KB pairs

5 Experiments

Experimental Setting We first performed preprocessing, including lowercase transformation, lemmatization and tokenization, on NLE-KB pairs and evaluation data. We used 71,310 n-grams (uni-, bi-, tri-), 990 entity types, and 660 predicates as relational components shown in Section 3.1. The sum of these three numbers (72,960) equals the size of the embeddings we are going to learn. In Table 2, we evaluated the quality of NLE-KB pairs ($\mathcal{MP}$ and $\mathcal{QP}$) described in Section 3.2. We can see that the quality of $\mathcal{QP}$ pairs is good, mainly due to human efforts. Also, we obtained $\mathcal{MP}$ pairs that have an acceptable quality using *threshold 3.0* for Equation 1, which leverages the redundancy information in the large-scale data (WIKIPEDIA). For our embedding learning, we set the embedding dimension n to 100, the learning rate (λ) for SGD to 0.0001, and the iteration number to 30. To make the decoding procedure computable, we kept only the popular KB-entity in the dictionary to map different entity mentions into a KB-entity.

We used two publicly released data sets for QA evaluations: **Free917** (Cai and Yates, 2013) includes the annotated lambda calculus forms for each question, and covers 81 domains and 635 Freebase relations; **WebQ.** (Berant et al., 2013) provides 5,810 question-answer pairs that are built by collecting common questions from Web-query logs and by manually labeling answers. We used the previous three approaches (Cai and Yates, 2013; Berant et al., 2013; Bao et al., 2014) as our baselines.

Experimental Results Table 3 reports the overall performances of our proposed KB-QA method on the two evaluation data sets and compares them with those of the three baselines. Note that we did not re-implement the baseline systems, but just borrowed the evaluation results reported in their

Methods	Free917	WebQ.
Cai and Yates (2013)	59.00%	N/A
Berant et al. (2013)	62.00%	31.40%
Bao et al. (2014)	N/A	37.50%
Our method	71.38%	41.34%

Table 3: Accuracy on the evaluation data

Methods	Free917	WebQ.
Our method	71.38%	41.34%
w/o $\mathcal{T}$-$\mathcal{P}$	70.65%	40.55%
w/o $\mathcal{C}$-$\mathcal{T}$	67.03%	38.44%
w/o $\mathcal{C}$-$\mathcal{P}$	31.16%	19.24%

Table 4: Ablation of the relationship types

papers. Although the KB used by our system is much larger than FREEBASE, we still think that the experimental results are directly comparable because we disallow all the entities that are not included in FREEBASE.

Table 3 shows that our method outperforms the baselines on both **Free917** and **WebQ.** data sets. We think that using the low-dimensional embeddings of n-grams rather than the lexical triggers greatly improves the coverage issue. Unlike the previous methods which perform entity disambiguation and predicate prediction separately, our method jointly performs these two tasks. More precisely, we consider the relationships $\mathcal{C}$-$\mathcal{T}$ and $\mathcal{C}$-$\mathcal{P}$ simultaneously to rank candidate KB-queries. In Table 1, the most independent NER in KB-QA systems may detect David as the subject entity, but our joint approach can predict the appropriate subject entity The City of David by leveraging not only the relationships with other components but also other relationships at once. The syntax-based (grammar formalism) approaches such as Combinatory Categorial Grammar (CCG) may experience errors if a question has grammatical errors. However, our bag-of-words model-based approach can handle any question as long as the question contains keywords that can help in understanding it.

Table 4 shows the contributions of the relationships ($\mathbb{R}$) between relational components $\mathcal{C}$, $\mathcal{T}$, and $\mathcal{P}$. For each row, we remove the similarity from each of the relationship types described in Section 3.1. We can see that the $\mathcal{C}$-$\mathcal{P}$ relationship plays a crucial role in translating NL-questions to KB-queries, while the other two relationships are slightly helpful.

Result Analysis Since the majority of questions in **WebQ.** tend to be more natural and diverse, our method cannot find the correct answers to many questions. The errors can be caused by any of the following reasons. First, some NLEs cannot be easily linked to existing KB-predicates, making it difficult to find the answer entity. Second, some entities can be mentioned in several different ways, e.g., nickname (*shaq*→Shaquille O'neal) and family name (*hitler*→Adolf Hitler). Third, in terms of KB coverage issues, we cannot detect the entities that are unpopular. Last, feature representation for a question can fail when the question consists of rare n-grams.

The two training sets shown in Section 3.2 are complementary: $\mathcal{QP}$ pairs provide more opportunities for us to learn the semantic associations between interrogative words and predicates. Such resources are especially important for understanding NL-questions, as most of them start with such 5W1H words; on the other hand, $\mathcal{MP}$ pairs enrich the semantic associations between context information (n-gram features) and predicates.

6 Conclusion

In this paper, we propose a novel method that transforms NL-questions into their corresponding logical forms using joint relational embeddings. We also built a simple and robust KB-QA system based on only the learned embeddings. Such embeddings learn the semantic associations between natural language statements and KB-properties from NLE-KB pairs that are automatically extracted from English WIKIPEDIA using KB-triples with weak supervision. Then, we generate all possible structured queries derived from latent logical features of the given NL-question, and rank them based on the similarity scores between those relational attributes. The experimental results show that our method outperforms the latest three KB-QA baseline systems. For our future work, we will build concept-level context embeddings by leveraging latent meanings of NLEs rather than their surface n-grams with the aligned logical features on KB.

Acknowledgement This research was supported by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2012M3C4A7033344).

References

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 967–976. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1415–1425. Association for Computational Linguistics.

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

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *In Proceedings of 15th International Conference on Artificial Intelligence and Statistics*.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Association for Computational Linguistics (ACL)*, pages 423–433. The Association for Computer Linguistics.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16(1):22–29, March.

Anthony Fader, Luke S. Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Association for Computational Linguistics (ACL)*, pages 1608–1618. The Association for Computer Linguistics.

Daniel Gerber and Axel-Cyrille Ngonga Ngomo. 2011. Bootstrapping the linked data web. In *1st Workshop on Web Scale Knowledge Extraction @ ISWC 2011*.

E. Kreyszig. 1979. *Advanced Engineering Mathematics*. Wiley.

Jayant Krishnamurthy and Tom M. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 754–765, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556, Seattle, Washington, USA, October. Association for Computational Linguistics.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 590–599, Stroudsburg, PA, USA. Association for Computational Linguistics.

RaymondJ. Mooney. 2007. Learning for semantic parsing. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 4394 of *Lecture Notes in Computer Science*, pages 311–324. Springer Berlin Heidelberg.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1135–1145, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2010. Large scale image annotation: Learning to rank with joint word-image embeddings. *Machine Learning*, 81(1):21–35, October.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1366–1371, Seattle, Washington, USA, October. Association for Computational Linguistics.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 956–966, Baltimore, Maryland, June. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 1050–1055. AAAI Press.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, pages 658–666. AUAI Press.

Adding High-Precision Links to Wikipedia

Thanapon Noraset **Chandra Bhagavatula** **Doug Downey**
Department of Electrical Engineering & Computer Science
Northwestern University
Evanston, IL 60208
{nor|csbhagav}@u.northwestern.edu, ddowney@eecs.northwestern.edu

Abstract

Wikipedia's link structure is a valuable resource for natural language processing tasks, but only a fraction of the concepts mentioned in each article are annotated with hyperlinks. In this paper, we study how to augment Wikipedia with additional high-precision links. We present *3W*, a system that identifies concept mentions in Wikipedia text, and links each mention to its referent page. *3W* leverages rich semantic information present in Wikipedia to achieve high precision. Our experiments demonstrate that *3W* can add an average of seven new links to each Wikipedia article, at a precision of 0.98.

1 Introduction

Wikipedia forms a valuable resource for many Natural Language Processing and Information Extraction tasks, such as Entity Linking (Cucerzan, 2007; Han and Zhao, 2009), Ontology Construction (Wu and Weld, 2008; Syed et al., 2008) and Knowledge Base Population (Hoffart et al., 2013; Lehmann et al., 2013). Wikipedia's links provide disambiguated semantic information. For example, when a system processes the text *"Chicago was received with critical acclaim"* from an article, the system does not need to infer the referent entity of *"Chicago"* if the word is already hyperlinked to the Wikipedia page of the Oscar-winning film. Unfortunately, in Wikipedia only a fraction of the phrases that can be linked are in fact annotated with a hyperlink. This is due to Wikipedia's conventions of only linking to each concept once, and only when the links have a certain level of utility for human readers.[1] We see this as an opportunity to improve Wikipedia as a resource for NLP systems. Our experiments estimate that as of September 2013, there were an average of 30 references to Wikipedia concepts left unlinked within each of English Wikipedia's four million pages.

In this paper, our goal is to augment Wikipedia with additional high-precision links, in order to provide a new resource for systems that use Wikipedia's link structure as a foundation. Identifying references to concepts (called *mentions*) in text and linking them to Wikipedia is a task known as *Wikification*. Wikification for general text has been addressed in a wide variety of recent work (Mihalcea and Csomai, 2007; Milne and Witten, 2008b; McNamee and Dang, 2009; Ratinov et al., 2011). The major challenge of this task is to resolve the ambiguity of phrases, and recent work makes use of various kinds of information found in the document to tackle the challenge. In contrast to this body of work, here we focus on the special case of Wikifying *Wikipedia articles*, instead of general documents. This gives us an advantage over general-text systems due to Wikipedia's rich content and existing link structure.

We introduce *3W*, a system that identifies mentions within Wikipedia and links each to its referent concept. We show how a Wikipedia-specific Semantic Relatedness measure that leverages the link structure of Wikipedia (Milne and Witten, 2008b) allows *3W* to be radically more precise at high levels of yield when compared to baseline Wikifiers that target general text. Our experiment shows that *3W* can add on average seven new links per article at precision of 0.98, adding approximately 28 million new links to 4 million articles across English Wikipedia.[2]

[1] http://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style_(linking)

[2] http://websail.cs.northwestern.edu/projects/3W

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 651–656,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

2 Problem Definition

In this section, we define our link extraction task. A link l is a pair of a surface form s_l and a concept t_l. A *surface form* is a span of tokens in an article, and the *concept* is a Wikipedia article referred to by the surface form. For existing hyperlinks, the surface form corresponds to the anchor text and the concept is the link target. For example, a hyperlink *[[Chicago City | Chicago]]* has surface form *"Chicago City"* and referent concept *Chicago*.[3] Given documents $\mathcal{D} = \{d_1, ..., d_{|\mathcal{D}|}\}$ and a set of links $\mathcal{L} = \{l_1, .., l_{|\mathcal{L}|}\} \in \mathcal{D}$, our goal is to generate a set of high-precision links $\mathcal{L}^*$ for $\mathcal{D}$, distinct from $\mathcal{L}$. In this paper, the document set $\mathcal{D}$ consists of articles from English Wikipedia, and $\mathcal{L}$ is the set of existing links on Wikipedia.

The task can be divided into 3 steps. The first step is to *extract* a set of potential mentions $\mathcal{M} = \{m_1, .., m_{|\mathcal{M}|}\}$ where m is, similar to l, a pair of surface form s_m and a set of candidate concepts $\mathcal{C}(m) = \{t_1, ..., t_{|\mathcal{C}(m)|}\}$. For m having $|\mathcal{C}(m)| > 1$, we need to *disambiguate* it by selecting only one target concept $t_m \in \mathcal{C}(m)$. Since the correct concept may not exist in $\mathcal{C}(m)$ and the previous step could output an incorrect concept, the final step is to decide whether to *link* and include m in $\mathcal{L}^*$. We describe the details of these steps in the following section.

3 System Overview

In this section, we describe in detail how *3W* adds high-precision links to Wikipedia.

3.1 Mention Extraction

In this step, we are given a document d, and the goal is to output a set of mentions $\mathcal{M}$. Our system finds a set of potential surface forms, s_m, by finding substrings in d that match the surface form of some links in $\mathcal{L}$. For example, from the phrase *"map of the United States on the wall"*, we can match 4 potential surface forms: *"map"*, *"United States"*, *"map of the United States"*, and *"wall"*. Notice that some of them are overlapping. The system selects a non-overlapping subset of the surface forms that maximizes the following score function:

$$Score(\mathcal{M}) = \sum_{m \in \mathcal{M}} \frac{T(s_m) PL(s_m)}{|\mathcal{C}(m)|} \quad (1)$$

[3] http://en.wikipedia.org/wiki/Chicago

where $PL(s_m)$ is the probability that the text s_m is linked (that is, the fraction of the occurrences of the string s_m in the corpus that are hyperlinked), $T(s_m)$ is the number of tokens in s_m, and $|\mathcal{C}(m)|$ is the number of candidate concepts. Intuitively, we prefer a longer surface form that is frequently linked and has a specific meaning. Furthermore, we eliminate common surface forms (i.e. *"wall"*) by requiring that $PL(s_m)$ exceed a threshold. In the previous example, we are left with only *"map of the United States"*.

Because Wikipedia's concepts are largely noun phrases, *3W* only looks for surface forms from top-level noun phrases generated by the Stanford Parser (Socher et al., 2013). In addition, each name entity (NE) (Finkel et al., 2005) is treated as an atomic token, meaning that multi-word NEs such as *"California Institute of the Arts"* will not be broken into multiple surface forms.

Finally, the system pairs the result surface forms with a set of candidate concepts, $\mathcal{C}(m)$, and outputs a set of mentions. $\mathcal{C}(m)$ consists of those concepts previously linked to the surface form in $\mathcal{L}$. For instance, the surface form *"map of the United States"* has been linked to three distinct concepts in English Wikipedia.

3.2 Disambiguation

Given a set of mentions $\mathcal{M}$ from the previous step, The next step is to select a concept $t \in \mathcal{C}(m)$ for each $m \in \mathcal{M}$. We take the common approach of ranking the candidate concepts. *3W* uses a machine learning model to perform pair-wise ranking of $t \in \mathcal{C}(m)$ and select the top-ranked candidate concept. We refer to *3W*'s disambiguation component as the *ranker*. The ranker requires a feature vector for each candidate concept of a mention. The rest of this section describes the features utilized by the ranker. The first two feature groups are commonly used in Wikification systems. The third feature group is specifically designed for mentions in Wikipedia articles.

3.2.1 Prior Probability Features

The conditional probability of a concept t given mention surface s_m, $P(t|s_m)$, is a common feature used for disambiguation. It forms a very strong Wikification baseline ($\sim$ 86% in micro-accuracy). This probability can be estimated using Wikipedia links ($\mathcal{L}$). In addition, we use the *external* partition of the

Google "Cross-Lingual Dictionary" described in (Spitkovsky and Chang, 2012) to get the estimates for the probability from links outside Wikipedia.

3.2.2 Lexical Features

To make use of text around a mention m, we create bag-of-word vectors of the mention's source document $d(m)$, and of a set of words surrounding the mention, referred to as the *context* $c(m)$. To compare with a concept, we also create bag-of-word vectors of candidate concept's document $d(t)$ and candidate concept's context $c(t)$. We then compute cosine similarities between the mention's vectors for $d(m)$ and $c(m)$, with the concept candidate vectors for $d(t)$ and $c(t)$ as in the Illinois Wikifier (Ratinov et al., 2011). In addition to similarities computed over the top-200 words (utilized in the Illinois Wikifier), we also compute similarity features over vectors of all words.

3.2.3 Wikipedia-specific Features

Because the links in an article are often related to one another, the existing links in a document form valuable clues for disambiguating mentions in the document. For each concept candidate $t \in \mathcal{C}(m)$, we compute a Semantic Relatedness (SR) measure between t and each concept from existing links in the source document. Our SR measure is based on the proportion of shared inlinks, as introduced by Milne and Witten (2008b). However, because Milne and Witten were focused on general text, they computed SR only between t and the *unambiguous* mentions (i.e. those m with $|\mathcal{C}(m)| = 1$) identified in the document. In our work, $d(m)$ is a Wikipedia article which is rich in existing links to Wikipedia concepts, and we can compute SR with all of them, resulting in a valuable feature for disambiguation as illustrated in our experiments. We use the SR implementation of Hecht et al. (2012). It is a modified version of Milne and Witten's measure that emphasizes links in Wikipedia article's overview. In addition, we add boolean features indicating whether s_m or t has already been linked in a document.

3.2.4 Reranking

The millions of existing Wikipedia links in $\mathcal{L}$ form a valuable source of training examples for our ranker. However, simply training on the links in $\mathcal{L}$ may result in poor performance, because those links exhibit systematic differences from the mentions in $\mathcal{M}$ that the ranker will be applied to. The reason is that our mention extractor attempts to populate $\mathcal{M}$ with *all* mentions, whereas $\mathcal{L}$ which contains only the specific subset of mentions that meet the hyperlinking conventions of Wikipedia, As a result, the features for $\mathcal{M}$ are distributed differently from those in $\mathcal{L}$, and a model trained on $\mathcal{L}$ may not might not perform well on $\mathcal{M}$. Our strategy is to leverage $\mathcal{L}$ to train an initial ranker, and then hand-label a small set of mentions from $\mathcal{M}$ to train a second-stage *re-ranker* that takes the ranking output of the initial ranker as a feature.

3.3 Linker

Our linker is a binary classifier that decides whether to include (link) each mention in $\mathcal{M}$ to the final output $\mathcal{L}^*$. Previous work has typically used a linker to determine so-called NIL mentions, where the referred-to concept is not in the target knowledge base (e.g., in the TAC KBP competition, half of the given mentions are NIL (Ji and Grishman, 2011)). The purpose of our linker is slightly different, because we also use a linker to control the precision of our output. We use a *probabilistic linker* that predicts a confidence estimate that the mention with its top-ranked candidate is correct. Our linker uses the same features as the ranker and an additional set of confidence signals: the number of times the top candidate concept appears in $\mathcal{L}$, and the score difference between the top-ranked candidate and the second-ranked candidate.

4 Experiments and Result

In this section, we provide an evaluation of our system and its subcomponents.

4.1 Experiment Setup

We trained our initial ranker models from 100,000 randomly selected existing links ($\mathcal{L}$). These links were excluded when building feature values (i.e. the prior probability, or Semantic Relatedness).

We formed an evaluation set of new links by applying our mention extractor to 2,000 randomly selected articles, and then manually labeling 1,900 of the mentions with either the correct concept or "no correct concept." We trained and tested our system on the evaluation set, using 10-fold cross validation. For each fold, we partitioned data

Model	Acc	Prec	Recall	F1
Prior	0.876	0.891	0.850	0.870
OnlyWikiLink −Wiki	0.896	0.905	0.871	0.888
OnlyWikiLink	0.944	0.950	0.920	0.935

Table 1: 10-fold cross validation performance of the initial rankers by Accuracy (excluded $\emptyset$-candidate mentions), BOT Precision, BOT Recall, BOT F1 on the 100,000 existing links.

Model	Acc	Yield	%Yield
Baseline	0.828	5	0.33%
OnlyWikiLink	0.705	150	9.80%
3W−Wiki	0.868	253	16.54%
3W	0.877	365	23.86%

Table 2: 10-fold cross validation performance of the system over 1,900 labeled mentions. Acc is disambiguation accuracy of *solvable* mentions. Yield is the number of output new mentions at precision ≥ 0.98, and %Yield is the percentage of Yield over the *solvable* mentions (recall).

into 3 parts. We used 760 mentions for training the final ranker. The linker was trained with 950 mentions and we tested our system using the other 190 mentions. Previous work has used various ML approaches for ranking, such as SVMs (Dredze et al., 2010). We found logistic regression produces similar accuracy to SVMs, but is faster for our feature set. For the linker, we use an SVM with probabilistic output (Wu et al., 2004; Chang and Lin, 2011) to estimate a confidence score for each output link.

4.2 Result

We first evaluate *3W*'s mention extraction. From the selected $2,000$ articles, the system extracted $59,454$ mentions ($\sim$30/article), in addition to the original $54,309$ links ($\sim$27/article). From the $1,900$ hand-labeled mentions, $1,530$ (80.5%) were *solvable* in that *3W* candidate set contained the correct target.

As described in section 3.2.4, *3W* employs a 2-stage ranker. We first evaluate just the initial ranker, using 10-fold cross validation on 100,000 existing links. We show micro accuracy and bag-of-title (BOT) performance used by Milne and Witten (2008b) in Table 1. The ranker with all features (*OnlyWikiLink*) outperforms the ranker without Wikipedia-specific features (*OnlyWikiLink−Wiki*) by approximately five points in F1. This demonstrates that Wikipedia's rich semantic content is helpful for disambiguation.

Next, we evaluate our full system performance (disambiguation and linking) over the hand-labeled evaluation set. We experimented with different configurations of the rankers and linkers. Our *Baseline* system disambiguates a mention m by selecting the most common concept for the surface $s(m)$. *OnlyWikiLink* uses the ranker model trained on only Wikipedia links, ignoring the labeled mentions. *3W* is our system using all features described in section 3.2,

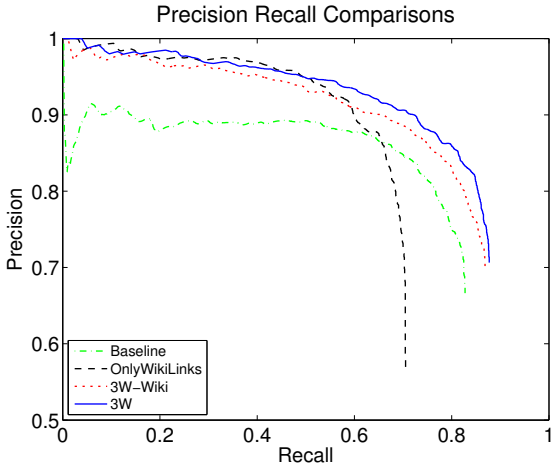


Figure 1: Plot between Precision and Recall of systems on 1,900 mentions from 10-fold cross validation.

and *3W−Wiki* is *3W* without Wikipedia-specific features. The last two configurations are trained using the labeled mentions.

Table 2 shows the disambiguation accuracy of each system over the solvable mentions. Our final system, *3W*, has the best disambiguation accuracy.

To evaluate the linking performance, we select the confidence threshold such that the system outputs mentions with precision of ≥ 0.98. The third column in Table 2 shows the *yield*, i.e. the number of mentions output at precision 0.98. *3W* outputs the largest number of new links (365). Nearly half (157) are new concepts that have not been linked in the source article. We find that the Rerank feature helps increase recall: without it, the yield of *3W* drops by 27%. Using %Yield, we estimate that *3W* will output $14,000$ new links for the selected $2,000$ articles ($\sim$7/article), and approximately 28 million new links across the 4 million articles of English Wikipedia.

Adjusting the confidence threshold allows the system to trade off precision and recall. Figure 1 shows a precision and recall curve. *3W* and *OnlyWikiLink* are comparable for

many high-precision points, but below 0.95 *OnlyWikiLink*'s precision drops quickly. Plots that finish at higher rightmost points in the graph indicate systems that achieve higher accuracy on the complete evaluation set.

5 Conclusions and Future Work

We presented *3W*, a system that adds high-precision links to Wikipedia. Whereas many Wikification systems focus on general text, *3W* is specialized toward Wikipedia articles. We showed that leveraging the link structure of Wikipedia provides advantages in disambiguation. In experiments, *3W* was shown to Wikipedia with $\sim$7 new links per article (an estimated 28m across 4 million Wikipedia articles) at high precision.

Acknowledgments

This work was supported in part by DARPA contract D11AP00268 and the Allen Institute for Artificial Intelligence.

References

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of EMNLP-CoNLL 2007*, pages 708–716.

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 277–285. Association for Computational Linguistics.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Xianpei Han and Jun Zhao. 2009. Named entity disambiguation by leveraging wikipedia semantic knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 215–224. ACM.

Brent Hecht, Samuel H Carton, Mahmood Quaderi, Johannes Schöning, Martin Raubal, Darren Gergle, and Doug Downey. 2012. Explanatory semantic relatedness and explicit spatialization for exploratory search. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 415–424. ACM.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1148–1158. Association for Computational Linguistics.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. 2013. Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the tac 2009 knowledge base population track. In *Text Analysis Conference (TAC)*, volume 17, pages 111–113.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 233–242. ACM.

David Milne and Ian H Witten. 2008b. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *ACL*.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

Valentin I. Spitkovsky and Angel X. Chang. 2012. A cross-lingual dictionary for english wikipedia concepts. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uur Doan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Zareen Saba Syed, Tim Finin, and Anupam Joshi. 2008. Wikipedia as an ontology for describing documents. In *ICWSM*.

Fei Wu and Daniel S Weld. 2008. Automatically refining the wikipedia infobox ontology. In *Proceedings of the 17th international conference on World Wide Web*, pages 635–644. ACM.

Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. 2004. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5(975-1005):4.

Finding Good Enough: A Task-Based Evaluation of Query Biased Summarization for Cross Language Information Retrieval

Jennifer Williams, Sharon Tam, Wade Shen
MIT Lincoln Laboratory Human Language Technology Group
244 Wood Street, Lexington, MA 02420 USA
jennifer.williams@ll.mit.edu, sharontam@alum.mit.edu
swade@ll.mit.edu

Abstract

In this paper we present our task-based evaluation of query biased summarization for cross-language information retrieval (CLIR) using relevance prediction. We describe our 13 summarization methods each from one of four summarization strategies. We show how well our methods perform using Farsi text from the CLEF 2008 shared-task, which we translated to English automtatically. We report precision/recall/F1, accuracy and time-on-task. We found that different summarization methods perform optimally for different evaluation metrics, but overall query biased word clouds are the best summarization strategy. In our analysis, we demonstrate that using the ROUGE metric on our sentence-based summaries cannot make the same kinds of distinctions as our evaluation framework does. Finally, we present our recommendations for creating much-needed evaluation standards and datasets.

1 Introduction

Despite many recent advances in query biased summarization for cross-language information retrieval (CLIR), there are no existing evaluation standards or datasets to make comparisons among different methods, and across different languages (Tombros and Sanderson, 1998; Pingali et al., 2007; McCallum et al., 2012; Bhaskar and Bandyopadhyay, 2012). Consider that creating this kind of summary requires familiarity with techniques from machine translation (MT), summarization, and information retrieval (IR). In this paper, we arrive at the intersection of each of these research areas. Query biased summarization (also known as *query-focused*, *query-relevant*, and *query-dependent*) involves automatically capturing relevant ideas and content from a document with respect to a given query, and presenting it as a condensed version of the original document. This kind of summarization is mostly used in search engines because when search results are tailored to a user's information need, the user can find texts that they are looking for more quickly and more accurately (Tombros and Sanderson, 1998; Mori et al., 2004). Query biased summarization is a valuable research area in natural language processing (NLP), especially for CLIR. Users of CLIR systems meet their information needs by submitting their queries in L_1 to search through documents that have been composed in L_2, even though they may not be familiar with L_2 (Hovy et al., 1999; Pingali et al., 2007).

There are no standards for objectively evaluating summaries for CLIR – a research gap that we begin to address in this paper. The problem we explore is two-fold: what kinds of summaries are well-suited for CLIR applications, and how should the summaries be evaluated. Our evaluation is *extrinsic*, that is to say we are interested in how summarization affects performance on a different task (Mani et al., 2002; McKeown et al., 2005; Dorr et al., 2005; Murray et al., 2009; McCallum et al., 2012). We use relevance prediction as our extrinsic task: a human must decide if a summary for a given document is relevant to a particular information need, or not. Relevance prediction is known to be useful as it correlates with some automatic *intrinsic* methods as well (President and Dorr, 2006; Hobson et al., 2007). To the best of our knowledge, we are the first to apply this evaluation framework to cross language query biased summarization.

Each one of the summarization methods that we

This work was sponsored by the Federal Bureau of Investigation under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 657–669,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

present in this paper belongs to one of the following strategies: (1) unbiased full machine translated text, (2) unbiased word clouds, (3) query biased word clouds, and (4) query biased sentence summaries. The methods and strategies that we present are fast, cheap, and language-independent. All of these strategies are *extractive*, meaning that we used existing parts of a document to create the condensed version, or summary.

We approach our task as an engineering problem: the goal is to decide if summaries are good enough to help CLIR system users find what they are looking for. We have simplified the task by assuming that a set of documents has already been retrieved from a search engine, as CLIR techniques are outside the scope of this paper. We predict that showing the full MT English text as a summarization strategy would not be particularly helpful in our relevance prediction task because the words in the text could be mixed-up, or sentences could be nonsensical, resulting in poor readability. For the same reasons, we expect that showing the full MT English text would take longer to arrive at a relevance decision. Finally, we predict that query biased summaries will result in faster, more accurate decisions from the participants (Tombros and Sanderson, 1998).

We treat the actual CLIR search engine as if it were a black box so that we can focus on evaluating if the summaries themselves are useful. As a starting point, we begin with some principles that we expect to hold true when we evaluate. These principles provide us with the kind of framework that we need for a productive and judicious discussion about how well a summarization method works. We encourage the NLP community to consider the following concepts when developing evaluation standards for this problem:

- End-user intelligiblity
- Query-salience
- Retrieval-relevance

Summaries should be presented to the end-user in a way that is both concise and intelligible, even if the machine translated text is difficult to understand. Our notions of *query-salience* and *retrieval-relevance* capture the expectation that good summaries will be efficient enough to help end-users fulfill their information needs. For query-salience, we want users to positively identify relevant documents. Similarly, for retrieval-relevance we want users to be able to find as many relevant documents as possible.

This paper is structured as follows: Section 2 presents related work; Section 3 describes our data and pre-processing; Section 4 details our summarization methods and strategies; Section 5 describes our experiments; Section 6 shows our results and analysis; and in Section 7, we conclude and discuss some future directions for the NLP community.

2 Related Work

Automatic summarization is generally a well-investigated research area. Summarization is a way of describing the relationships of words in documents to the information content of that document (Luhn, 1958; Edmunson, 1969; Salton and Yang, 1973; Robertson and Walker, 1994; Church and Gale, 1999; Robertson, 2004). Recent work has looked at creating summaries of single and multiple documents (Radev et al., 2004; Erkan and Radev, 2004; Wan et al., 2007; Yin et al., 2012; Chatterjee et al., 2012), as well as summary evaluation (Jing et al., 1998; Tombros and Sanderson 1998; Mani et al., 1998; Mani et al., 1999; Mani, 2001; Lin and Hovy, 2003; Lin, 2004; Nenkova et al., 2007; Hobson et al., 2007; Owczarzak et al., 2012), query and topic biased summarization (Berger and Mittal, 2000; Otterbacher et al., 2005; Daume and Marcu, 2006; Chali and Joty, 2008; Otterbacher et al., 2009; Bando et al., 2010; Bhaskar and Bandyopadhyay, 2012; Harwath and Hazen, 2012; Yin et al., 2012), and summarization across languages (Pingali et al., 2007; Orăsan and Chiorean, 2008; Wan et al., 2010; Azarbonyad et al., 2013).

2.1 Query Biased Summarization

Previous work most closely related to our own comes from Pingali et al., (2007). In their work, they present their method for cross-language query biased summarization for Telugu and English. Their work was motivated by the need for people to have access to foreign-language documents from a search engine even though the users were not familiar with the foreign language, in their case English. They used language modeling and translation probability to translate a user's query into L_2, and then summarized each document in L_2 with respect to the query. In their final step, they translated the summary from L_2 back

to L_1 for the user. They evaluated their method on the DUC 2005 query-focused summarization shared-task with ROUGE scores. We compare our methods to this work also on the DUC 2005 task. Our work demonstrates the first attempt to draw at a comparison between user-based studies and intrinsic evaluation with ROUGE. However, one of the limitations with evaluating this way is that the shared-task documents and queries are monolingual.

Bhaskar and Bandyopadhyay (2012) tried a subjective evaluation of extractive cross-language query biased summarization for 7 different languages. They extracted sentences, then scored and ranked the sentences to generate query dependent snippets of documents for their cross lingual information access (CLIA) system. However, the snippet quality was determined subjectively based on scores on a scale of 0 to 1 (with 1 being best). Each score indicated annotator satisfaction for a given snippet. Our evaluation methodology is objective: we ask users to decide if a given document is relevant to an information need, or not.

2.2 Machine Translation Effects

Machine translation quality can affect summarization quality. Wan et al. (2010) researched the effects of MT quality prediction on cross-language document summarization. They generated 5-sentence summaries in Chinese using English source documents. To select sentences, they used predicted translation quality, sentence position, and sentence informativeness. In their evaluation, they employed 4 Chinese-speakers to subjectively rate summaries on a 5-point scale (5 being best) along the dimensions of content, readability, and overall impression. They showed that their approach of using MT quality scores did improve summarization quality on average. While their findings are important, their work did not address query biasing or objective evaluation of the summaries. We attempt to overcome limitations of machine translation quality by using word clouds as one of our summarization strategies.

Knowing when to translate is another challenge for cross-language query biased summarization. Several options exist for when and what to translate during the summarization process: (1) the source documents can be translated, (2) the user's query can be translated, (3) the final summary can be translated, or (4) some combination of these. An example of translating only the summaries themselves can be found in Wan et al., (2010). On the other hand, Pingali et al. (2007) translated the queries and the summaries. In our work, we used gold-translated queries from the CLEF 2008 dataset, and machine translated source documents. We briefly address this in our work, but note that a full discussion of when and what to translate, and those effects on summarization quality, is outside of the scope of this paper.

2.3 Summarization Evaluation

There has been a lot of work towards developing metrics for understanding what makes a summary good. Evaluation metrics are either *intrinsic* or *extrinsic*. Intrinsic metrics, such as ROUGE, measure the quality of a summary with respect to gold human-generated summaries (Lin, 2004; Lin and Hovy, 2003). Generating gold standard summaries is expensive and time-consuming, a problem that persists with cross-language query biased summarization because those summaries must be query biased as well as in a different language from the source documents.

On the other hand, extrinsic metrics measure the quality of summaries at the system level, by looking at overall system performance on downstream tasks (Jing et al, 1998; Tombros and Sanderson, 1998). One of the most important findings for query biased summarization comes from Tombros and Sanderson (1998). In their monolingual task-based evaluation, they measured user speed and accuracy at identifying relevant documents. They found that query biased summarization improved the user speed and accuracy when the user was asked to make relevance judgements for IR tasks. We also expect that our evaluation will demonstrate that user speed and accuracy is better when summaries are query biased.

3 Data and Pre-Processing

We used data from the Farsi CLEF 2008 ad hoc task (Agirre et al., 2009). Each of the queries included in this dataset consisted of a title, narrative, and description. Figure 1 shows an example of the elements of a CLEF 2008 query. All of the automatic query-biasing in this work was based on the query titles. For our human relevance prediction task on Mechanical Turk, we used the narrative version. The CLEF 2008 dataset included a ground-truth answer key indicating which docu-

ments were relevant to each query. For each query, we randomly selected 5 documents that were relevant as well as 5 documents that were not relevant. The subset of CLEF 2008 data that we used therefore consisted of 500 original Farsi documents and 50 parallel English-Farsi queries. Next we will describe our text pre-processing steps for both languages as well as how we created our parallel English documents.

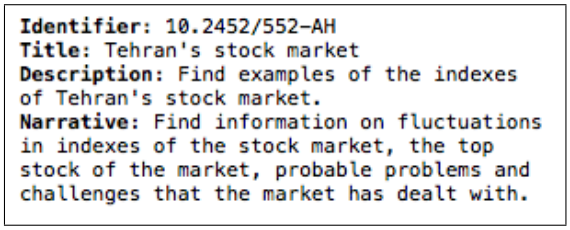


Figure 1: Full MT English summary and CLEF 2008 English query (title, description, narrative).

3.1 English Documents

All of our English documents were created automatically by translating the original Farsi documents into English (Drexler et al., 2012). The translated documents were sentence-aligned with one sentence per line. For all of our summarization experiments (except unbised full MT text), we processed the text as follows: removed extra spaces, removed punctuation, folded to lowercase, and removed digits. We also removed common English stopwords[2] from the texts.

3.2 Farsi Documents

We used the original CLEF 2008 Farsi documents for two of our summarization methods. We stemmed words in each document using automatic morphological analysis with Morfessor CatMAP. We note that within-sentence punctuation was removed during this process (Creutz and Lagus, 2007). We also removed Farsi stopwords and digits.

4 Summarization Strategies

All of our summarization methods were extractive except for unbiased full machine translated text. In this section, we describe each of our 13 summarization methods which we have organized into one of the following strategies: (1) unbiased full machine translated text, (2) unbiased word cloud summaries, (3) query biased word cloud summaries, and (4) query biased sentence summaries. Regardless of which summarization method used, we highlighted words in yellow that also appeard in the query. Let t be a term in document d where $d \in D_L$ and D_L is a collection of documents in a particular language. Note that for our summarization methods, term weightings were calculated separately for each language. While $|D| = 1000$, we calculated term weightings based on $|D_E| = 500$ and $|D_F| = 500$. Finally, let q be a query where $q \in Q$ and Q is our set of 50 parallel English-Farsi CLEF queries. Assume that log refers to log_{10}.

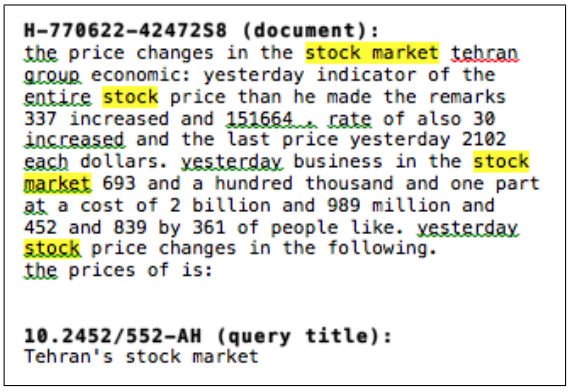


Figure 2: Full MT English summary and CLEF 2008 English query.

4.1 Unbiased Full Machine Translated English

Our first baseline approach was to use all of the raw machine translation output (no subsets of the sentences were used). Each summary therefore consisted of the full text of an entire document automatically translated from Farsi to English (Drexler et al., 2012). Figure 2 shows an example full text document translated from Farsi to English and a gold-standard English CLEF query. Note that we use this particular document-query pair as an example throughout this paper (document: H-770622-42472S8, query: 10.2452/552-AH). According to the CLEF answer key, the sample document is relevant to the sample query.

4.2 Unbiased Word Clouds

For our second baseline approach, we ranked terms in a document and displayed them as *word clouds*. Word clouds are one a way to arrange a collection of words where each word can vary

[2]English and Farsi stopword lists from: http://members.unine.ch/jacques.savoy/clef/index.html

in size. We used word clouds as a summarization strategy to overcome any potential disfluencies from the machine translation output and also to see if they are feasible at all for summarization. All of our methods for word clouds used words from machine translated English text. Each term-ranking method below generates different ranked lists of terms, which we used to create different word clouds. We created one word cloud per document using the top 12 ranked words. We used the raw term scores to scale text font size, so that words with a highter score appeared larger and more prominent in a word cloud. Words were shuffled such that the exact ordering of words was at random.

I: Term Frequency (TF) Term frequency is very commonly used for finding important terms in a document. Given a term t in a document d, the number of times that term occurs is:

$$tf_{t,d} = |t \in d|$$

II: Inverse Document Frequency (IDF) The idf term weighting is typically used in IR and other text categorization tasks to make distinctions between documents. The version of idf that we used throughout our work came from Erkan and Radev (2004) and Otterbacher et al. (2009), in keeping consistent with theirs. Let N be the number of documents in the collection, such that $N = |D|$ and n_t is the number of documents that contain term t, such that $n_t = |\{d \in D : t \in d\}|$, then:

$$idf_t = log \frac{N+1}{0.5 \times n_t}$$

While *idf* is usually thought of as a type of heuristic, there have been some discussions about its theoretical basis (Robertson, 2004; Robertson and Walker, 1994; Church and Gale, 1999; Salton and Yang, 1973). An example of this summary is shown in Figure 3.

III: Term Frequency Inverse Document Frequency (TFIDF) We use $tfidf_{t,d}$ term weighting to find terms which are both rare and important for a document, with respect to terms across all other documents in the collection:

$$tfidf_{t,d} = tf_{t,d} \times idf_t$$

4.3 Query Biased Word Clouds

We generated query biased word clouds following the same principles as our unbiased word clouds,

Figure 3: Word cloud summary for inverse document frequency (IDF), for query "Tehran's stock market".

namely the text font scaling and highlighting remained the same.

IV. Query Biased Term Frequency (TFQ) In Figure 4 we show a sample word cloud summary based on query biased term frequency. We define query biased term frequency tfQ at the document level, as:

$$tfQ_{t,d,q} = \begin{cases} 2tf_{t,d}, & \text{if } t \in q \\ tf_{t,d}, & \text{otherwise} \end{cases}$$

Figure 4: Word cloud summary for query biased term frequency (TFQ), for query "Tehran's stock market".

V. Query Biased Inverse Document Frequency (IDFQ) Since idf helps with identifying terms that discriminate documents in a collection, we would expect that query biased idf would help to identify documents that are relevant to a query:

$$idfQ_{t,q} = \begin{cases} 2idf_t, & \text{if } t \in q \\ idf_t, & \text{otherwise} \end{cases}$$

VI. Query Biased TFIDF (TFIDFQ) We define query biased $tf \times idf$ similarly to our TFQ and IDFQ, at the document level:

$$tfidfQ_{t,d,q} = \begin{cases} 2tf_{t,d} \times idf_t, & \text{if } t \in q \\ tf_{t,d} \times idf_t, & \text{otherwise} \end{cases}$$

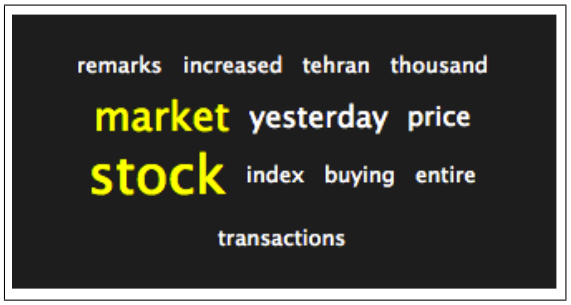


Figure 5: Word cloud summary for scaled query biased term frequency (SFQ) for query "Tehran's stock market".

VII. Query Biased Scaled Frequency (SFQ) This term weighting scheme, which we call scaled query biased term frequency or sfQ, is a variant of the traditional $tf \times idf$ weighting. First, we project the usual term frequency into log-space, for a term t in document d with:

$$tfS_{t,d} = log(tf_{t,d})$$

We let $tfS_{t,d} \approx 0$ when $tf_{t,d} = 1$. We believe that singleton terms in a document provide no indication that a document is query-relevant, and treament of singleton terms in this way would have the potential to reduce false-positives in our relevance prediction task. Note that scaled term frequency differs from Robertson's (2004) *inverse total term frequency* in the sense that our method involves no consideration of term position within a document. Scaled query biased term frequency, shown in Figure 5, is defined as:

$$sfQ_{t,d,q} = \begin{cases} 2tfS_{t,d} \times idf_t, & \text{if } t \in q \\ tfS_{t,d} \times idf_t, & \text{otherwise} \end{cases}$$

VIII. Word Relevance (W) We adapted an existing relevance weighting from Allan et al., (2003), that was originally formulated for ranking sentences with respect to a query. However, we modified their originaly ranking method so that we could rank individual terms in a document instead of sentences. Our method for word relevance, W is defined as:

$$W_{t,d,q} = log(tf_{t,d} + 1) \times log(tf_{t,q} + 1) \times idf_t$$

In W, term frequency values are *smoothed* by adding 1. The smoothing could especially affect rare terms and singletons, when $tf_{t,d}$ is very low. All terms in a query or a document will be weighted and each term could potentially contribute to summary.

4.4 Query Biased Sentence Summaries

Sentences are a canonical unit to use in extractive summaries. In this section we describe four different sentence scoring methods that we used. These methods show how to calculate sentence scores for a given document with respect to a given query. Sentences for a document were always ranked using the raw score value output generated from a scoring method. Each document summary contained the top 3 ranked sentences where the sentences were simply listed out. Each of these methods used sentence-aligned English machine translated documents, and two of them also used the original Farsi text.

IX. Sentence Relevance (REL) Our sentence relevance scoring method comes from Allan et al. (2003). The sentence weight is a summation over words that appear in the query. We provide their sentence scoring formula here. This calculates the relevance score for a sentence s from document d, to a query q:

$$rel_{(s|q)} = \sum_{t \in s} log(tf_{t,s} + 1) \times log(tf_{t,q} + 1) \times idf_t$$

Terms will occur in either the sentence or the query, or both. We applied this method to machine tranlsated English text. The output of this method is a relevance score for each sentence in a given document. We used those scores to rank sentences in each document from our English machine translated text.

X. Query Biased Lexrank (LQ) We implemented query biased LexRank, a well-known graph-based summarization method (Otterbacher et al., 2009). It is a modified version of the original LexRank algorithm (Erkan and Radev, 2004; Page et al., 1998). The similarity metric, $sim_{x,y}$, also known as *idf-modified cosine similarity*, measures the distance between two sentences x and y in a document d, defined as:

$$sim_{x,y} = \frac{\sum_{t \in x,y} tf_{t,x} \times tf_{t,y} \times (idf_t)^2}{\sqrt{\sum_{t \in x} tfidf_{t,x}^2} \sqrt{\sum_{t \in y} tfidf_{t,y}^2}}$$

We used $sim_{x,y}$ to score the similarity of sentence-to-sentence, resulting in a similarity

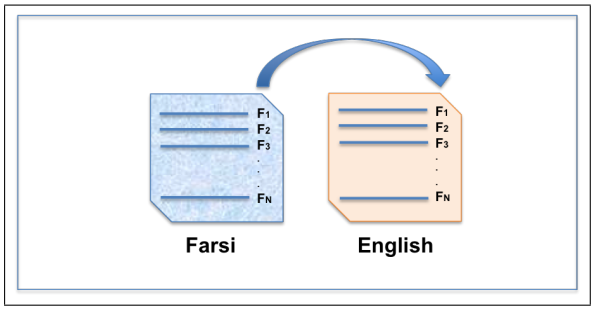


Figure 6: LQP - projecting Farsi sentence scores onto parallel English sentences.

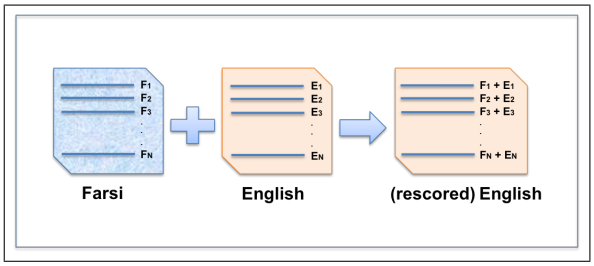


Figure 7: LQC - Farsi sentence scores are combined with parallel English sentence scores to obtain sentence re-ranking.

graph where each vertex was a sentence and each edge was the cosine similarity between sentences. We normalized the cosine matrix with a similarity threshold ($t = 0.05$), so that sentences above this threshold were given similarity 1, and 0 otherwise. We used $rel_{(s|q)}$ to score sentence-to-query. The LexRank score for each sentence was then calculated as:

$$LQ_{s|q} = \frac{d \times rel_{s|q}}{\sum_{z \in C} rel_{z|q}} + (1-d) \times \sum_{v \in adj[s]} \frac{sim_{s,v}}{\sum_{r \in adj[v]} sim_{v,r}} LQ_{v|q}$$

where C is the set of all sentences in a given document. Here the parameter d is just a damper to designate a probability of randomly jumping to one of the sentences in the graph ($d = 0.7$). We found the stationary distribution by applying the power method ($\epsilon = 5$), which is guaranteed to converge to a stationary distribution (Otterbacher et al., 2009). The output of LQ is a score for each sentence from a given document with respect to a query. We used that score to rank sentences in each document from our English machine translated text.

XI. Projected Cross-Language Query Biased Lexrank (LQP) We introduce LQP to describe a way of scoring and ranking sentences such that the L_1 (English) summaries are biased from the L_2 (Farsi) query and source document. Our gold-standard Farsi queries were included with our CLEF 2008 data, making them more reliable than what we could get from automatic translation. First, sentences from each Farsi document were scored with Farsi queries using LQ, described above. Then each LQ score was projected onto sentence-aligned English. We demonstrate LQP in Figure 6. By doing this, we simulated translating the user's English query into Farsi with the best possible query translation, before proceeding with summarization. This approach to cross-language summarization could be of interest for CLIR systems that do query translation on-the-fly. It is also of interest for summarization systems that need to utilize previously translated source documents the capability is lacking to translate summaries from L_2 to L_1.

XII. Combinatory Query Biased Lexrank (LQC) Another variation of LexRank that we introduce in this work is LQC, which combines LexRank scores from both languages to re-rank sentences. A visual summary of this method is shown in Figure 7. We accomplished our re-ranking by first running LQ on Farsi and English separately, then adding the two scores together. This combination of Farsi and English scores provided us with a different way to score and rank sentences, compared with LQ and LQP. The idea behind combinatory query biased LexRank is to take advantage of sentences which are high-ranking in Farsi but not in English. The LQC method exploits all available resources in our dataset: L_1 and L_2 queries as well as L_1 and L_2 documents.

5 Experiments

We tested each of our summarization methods and overall strategies in a task-based evaluation framework using relevance prediction. We used Mechanical Turk for our experiments since it has been shown to be useful for evaluating NLP systems (Callison-Burch 2009; Gillick and Liu, 2010). We obtained human judgments for whether or not a document was considered relevant to a query, or information need. We measured the relevance

judgements by precision/recall/F1, accuracy, and also time-on-task based on the average response time per Human Intelligence Task (HIT).

5.1 Mechanical Turk

In our Mechanical Turk experiment, we used terminology from CLEF 2008 to describe a query as an "information need". All of the Mechanical Turk workers were presented with the following for their individual HIT: instructions, an information need and one summary for a document. Workers were asked to indicate if the given summary for a document was relevant to the given information need (Hobson et al., 2007). Workers were not shown the original Farsi source documents. We paid workers $0.01 per HIT. We obtained 5 HITs for each information need and summary pair. We used a built-in approval rate qualification provided by Mechanical Turk to restrict which workers could work on our tasks. Each worker had an approval rate of at least 95

> Instructions: Each image below consists of a statement summarizing the information you are trying to find from a set of documents followed by a summary of one of the documents returned when you query the documents. Based on the summary, choose whether you think the document returned is relevant to the information need. NOTE: It may be difficult to distinguish whether the document is relevant as the text may be difficult to understand. Just use your best judgment.

6 Results and Analysis

We present our experiment results and additional analysis. First, we report the results of our relevance prediction task, showing performance for individual summarization methods as well as performance for the overall strategies. Then we show analysis of our results from the monolingual question-biased shared-task for DUC 2005, as well as a comparison to previous work.

6.1 Results for Individual Methods

Our results are shown in Table 1. We report performance for 13 individual methods as well as overall peformance on the 4 different summarization strategies. To calculate the performance for each strategy, we used the arithmetic mean of the corresponding individual methods. We measured precision, recall and F1 to give us a sense of our summaries might influence document retrieval in an actual CLIR system. We also measured accuracy and time-on-task. For these latter two metrics, we distinguish between summaries that were relevant (R) and non-relevant (NR).

All of the summarization-based methods favored recall over precision: documents were marked 'relevant' more often than 'non-relevant'. For many of the methods shown in Table 1, workers spent more time correctly deciding 'relevant' than correctly deciding 'non-relevant'. This suggests some workers participated in our Mechanical Turk task purposefully. For many of the summarization methods, workers were able to positively identify relevant documents.

From Table 1 we see that Full MT performed better on precision than all of the other methods and strategies, but we note that performance on precision was generally very low. This might be due to Mechanical Turk workers overgeneralizing by marking summaries as relevant when they were not. Some individual methods preserve our principle of *retrieval-relevance*, as indicated by the higher recall scores for SQF, LQEF, and TFQ. That is to say, these particular query biased summarization methods can be used to assist users with identifying more relevant documents. The accuracy on relevant documents addresses our principle of *query-salience*, and it is especially high for our query-biased methods: LQEF, SQF, LQ, and TFQ. The results also seem to fit our intuition that the summary in Figure 3 seems less relevant to the summaries shown in Figures 4 & 5 even though these are the same documents biased on the same query "Tehran stock market".

Overall, query biased word clouds outperform the other summarization strategies for 5 out of 7 metrics. This could be due to the fact that word clouds provide a very concise and overview of a document, which is one of the main goals for automatic summarization. Along these lines, word clouds are probably not subject to the effects of MT quality and we believe it is possible that MT quality could have had a negative impact on our query biased extracted sentence summaries, as well as our full MT English texts.

Table 1: Individual method results: precision/recall/F1, time-on-task, and accuracy. Note that results for time-on-task and accuracy scores are distinguished for relevant (R) and non-relevant (NR) documents.

	Precision, Recall, F1			Time-on-Task		Accuracy	
Summarization Strategy	**Prec.**	**Rec.**	**F1**	**R**	**NR**	**R**	**NR**
Unbiased Full MT English	**0.653**	**0.636**	**0.644**	**219.5**	**77.6**	**0.696**	**0.712**
TF	0.615	0.777	0.686	33.5	34.6	0.840	0.508
IDF	0.537	0.470	0.501	84.7	45.8	0.444	0.700
TFIDF	0.647	0.710	0.677	33.2	38.2	0.772	0.656
Unbiased Word Clouds	**0.599**	**0.652**	**0.621**	**50.5**	**39.5**	**0.685**	**0.621**
TFQ	0.605	0.809	0.692	55.3	82.4	0.864	0.436
IDFQ	0.582	0.793	0.671	23.6	31.6	0.844	0.436
TFIDFQ	0.599	0.738	0.661	37.9	26.9	0.804	0.500
SFQ	0.591	0.813	0.685	55.7	49.4	0.876	0.504
W	0.611	0.738	0.669	28.2	28.9	0.840	0.564
Query Biased Word Clouds	**0.597**	**0.778**	**0.675**	**36.4**	**34.2**	**0.846**	**0.488**
REL	0.582	0.746	0.654	30.6	44.3	0.832	0.548
LQ	0.549	0.783	0.646	64.4	54.8	0.868	0.292
LQP	0.578	0.734	0.647	28.2	28.0	0.768	0.472
LQC	0.557	0.810	0.660	33.9	38.8	0.896	0.292
Query Biased Sentences	**0.566**	**0.768**	**0.651**	**39.2**	**41.5**	**0.841**	**0.401**

Table 2: Comparison of peer systems on DUC 2005 shared-task for monolingual question-biased summarization, f-scores from ROUGE-2 and ROUGE-SU4.

Peer ID	**ROUGE-2**	**ROUGE-SU4**
17	0.07170	0.12970
8	0.06960	0.12790
4	0.06850	0.12770
Tel-Eng-Sum	0.06048	0.12058
LQ	0.05124	0.09343
REL	0.04914	0.09081

6.2 Analysis with DUC 2005

We analysed our summarization methods by comparing two of our sentence-based methods (LQ and REL) with peers from the monolingual question-biased summarization shared-task for DUC 2005. Even though DUC 2005 is a monolingual task, we decided to use it as part of our analysis for two reasons: (1) to see how well we could do with query/question biasing while ignoring the variables introduced by MT and cross-language text, and (2) to make a comparison to previous work. Pingali et al., (2007) also used this the same DUC task to assess their cross-language query biased summarization system. Systems from the DUC 2005 question-biased summarization task were evaluated automatically against human gold-standard summaries using ROUGE (Lin and Hovy, 2003) . Our results from the DUC 2005 shared-task are shown in Table 2, reported as ROUGE-2 and ROUGE-SU4 f-scores, as these two variations of ROUGE are the most helpful (Dang, 2005; Pingali et al., 2007).

Table 2 shows scores for several top peer systems, as well as results for the Tel-Eng-Sum method from Pingali et al., (2007). While we have reported f-scores in our analysis, we also note that our implementations of LQ and REL outperform all of the DUC 2005 peer systems for precision, as shown in Table 3. We also know that ROUGE cannot be used for comparing sentence summaries to ranked lists of words and there are no existing intrinsic methods to make that kind of comparison. Therefore we were able to successfully compare just 2 of our sentence-based methods to previous work using ROUGE.

7 Discussion and Future Work

Cross-language query biased summarization is an important part of CLIR, because it helps the user decide which foreign-language documents they might want to read. But, how do we know if

Table 3: Top 3 system precision scores for ROUGE-2 and ROUGE-SU4.

Peer ID	ROUGE-2	ROUGE-SU4
LQ	**0.08272**	**0.15197**
REL	**0.0809**	**0.15049**
15	0.07249	0.13129

a query biased summary is "good enough" to be used in a real-world CLIR system? We want to be able to say that we can do query biased summarization just as well for both monolingual and cross-language IR systems. From previous work, there has been some variability with regard to when and what to translate - variables which have no impact on monolingual summarization. We attempted to address this issue with two of our methods: LQP and LQC. To fully exploit the MT variable, we would need many more relevance prediction experiments using humans who know L_1 and others who know L_2. Unfortunately in our case, we were not able to find Farsi speakers on Mechanical Turk. Access to these speakers would have allowed us to try further experiments as well as other kinds of analysis.

Our results on the relevance prediction task tell us that query biased summarization strategies help users identify relevant documents faster and with better accuracy than unbiased summaries. Our findings support the findings of Tombros and Sanderson (1998). Another important finding is that now we can weigh tradeoffs so that different summarization methods could be used to optimize over different metrics. For example, if we want to optimize for retrieval-relevance we might select a summarization method that tends to have higher recall, such as scaled query biased term frequency (SFQ). Similarly, we could optimize over accuracy on relevant documents, and use Combinatory LexRank (LQC) with Farsi and English together.

We have shown that the relevance prediction tasks can be crowdsourced on Mechanical Turk with reasonable results. The data we used from the Farsi CLEF 2008 ad-hoc task included an answer key, but there were no parallel English documents. However, in order for the NLP community to make strides in evaluating cross-language query biased summarization for CLIR, we will need stardards and data. Optimal data would be parallel datasets consisting of documents in L_1 and L_2 with queries in L_1 and L_2 along with an answer key specifying which documents are relevant to the queries. Further we would also need sets of human gold-standard query biased summaries in L_1 and L_2. These standards and data would allow us to compare method-to-method across different languages, while simultaneously allowing us to tease apart other variables such as: when and what to translate, translation quality, methods for biasing, and type of summarization strategy (sentences, words, etc). And of course it would be better if this standard dataset was multilingual instead of billingual, for obvious reasons.

We have approached cross-language query biased summarization as a stand-alone problem, treating the CLIR system and document retrieval as a black box. However, summaries need to preserve query-salience: summaries should not make it more difficult to positively identify relavant documents. And they should also preserve retrieval-relevance: summaries should help users identify as many relevant documents as possible.

Acknowledgments

We would like to express thanks to David Harwath at MIT Computer Science and Artificial Intelligence Laboratory (CSAIL), who helped us develop and implement ideas in this paper. We also want to thank Terry Gleason from MIT Lincoln Laboratory for providing machine translations.

References

Eneko Agirre, Giorgio Maria Di Nunzio, Nicola Ferro, Thomas Mandl, and Carol Peters. CLEF 2008: Ad hoc track overview. In *Evaluating Systems for Multilingual and Multimodal Information Access*, pp 15–37. Springer Berlin Heidelberg, 2009.

James Allan, Courtney Wade, and Alvaro Bolivar. Retrieval and Novelty Detection at the Sentence Level. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, (SIGIR '03). ACM, New York, NY, USA, 314-321.

Hosein Azarbonyad, Azadeh Shakery, and Heshaam Faili. Exploiting Multiple Translation Resources for English-Persian Cross Language Information Retrieval. In P. Forner, H. Müller, R. Paredes, P. Rosso, and B. Stein, editors, *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*, volume 8138 of *Lecture Notes in Computer Science*, pp 93–99. Springer Berlin Heidelberg, 2013.

Lorena Leal Bando, Falk Scholer, Andrew Turpin. Constructing Query-biased Summaries: A Comparison of Human and System Generated Snippets. In

Proceedings of the Third Symposium on Information Interaction in Context (IIiX '10), ACM 2010, New York, NY, USA, 195-204.

Adam Berger and Vibhu O Mittal. Query-Relevant Summarization Using FAQs. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL 2000.

Pinaki Bhaskar and Sivaji Bandyopadhyay. Cross-Lingual Query Dependent Snippet Generation. *International Journal of Computer Science and Information Technology (IJCSIT)*, 3(4), 2012.

Pinaki Bhaskar and Sivaji Bandyopadhyay. Language Independent Query Focused Snippet Generation. In T. Catarci, P. Forner, D. Hiemstra, A. Peñas, and G. Santucci, editors, *Information Access Evaluation. Multilinguality, Multimodality, and Visual Analytics*, volume 7488 of *Lecture Notes in Computer Science*, pp 138–140. Springer Berlin Heidelberg, 2012.

Stephen P. Borgatti, Kathleen M. Carley, David Krackhardt. On the Robustness of Centrality Measures Under Conditions of Imperfect Data. *Social Networks*, (28):124–136, 2006.

Florian Boudin, Stéphane Huet, and Juan-Manuel Torres-Moreno. A Graph-Based Approach to Cross-Language Multi-Document Summarization. *Polibits*, (43):113–118, 2011.

Chris Callison-Burch. Fast, Cheap, and Creative: Evaluating Translation Quality Using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp 286–295, Singapore, ACL 2009.

Yllias Chali and Shafiq R. Joty. Unsupervised Approach for Selecting Sentences in Query-Based Summarization. In *Proceedings of the Twenty-First International FLAIRS Conference*, 2008.

Niladri Chatterjee, Amol Mittal, and Shubham Goyal. Single Document Extractive Text Summarization Using Genetic Algorithms. In *Emerging Applications of Information Technology (EAIT), 2012 Third International Conference*, pp 19–23, 2012.

Kenneth W. Church and William A. Gale. Inverse Document Frequency (IDF): A Measure of Deviations From Poisson. In *Natural language processing using very large corpora*, pages 283–295. Springer, 1999.

Mathias Creutz and Krista Lagus. Unsupervised Models for Morpheme Segmentation and Morphology Learning. *ACM Transactions on Speech and Language Processing*, 4(1):3:1–3:34, February 2007.

Hoa Trang Dang. Overview of DUC 2005. In *Proceedings of the Document Understanding Conference*, 2005.

Hal Daumé III, Daniel Marcu. Bayesian Query-Focused Summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL 2006.

Jennifer Drexler, Wade Shen, Terry P. Gleason, Timothy R. Anderson, Raymond E. Slyh, Brian M. Ore, and Eric G. Hansen. The MIT-LL/AFRL IWSLT-2012 MT System. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, Hong Kong, December 2012.

Bonnie J. Dorr, Christof Monz, Stacy President, Richard Schwartz, and David Zajic. A Methodology for Extrinsic Evaluation of Text Summarization: Does ROUGE Correlate? In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp 1-8. Ann Arbor, ACL 2005.

H. P. Edmundson. New Methods in Automatic Extracting. In *Journal of the ACM*, 16(2):264–285, April 1969.

Güneş Erkan and Dragomir R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479, December 2004.

Dan Gillick and Yang Liu. Non-Expert Evaluation of Summarization Systems is Risky. In *Proceedings of NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pp 148-151, Los Angeles, California, USA, June, 2010.

David Harwath and Timothy J. Hazen. Topic Identification Based Extrinsic Evaluation of Summarization Techniques Applied to Conversational Speech. In *Proceedings of ICASSP*, 2012: 5073-5076.

Stacy P. Hobson, Bonnie J. Dorr, Christof Monz, and Richard Schwartz. Task-Eased Evaluation of Text Summarization Using Relevance Prediction. In *Information Processing Management*, 43(6): 1482-1499, 2007.

Hongyan Jing, Regina Barzilay, Kathleen McKeown, and Michael Elhadad. Summarization Evaluation Methods: Experiments and Analysis. In *Proceedings of American Association for Artificial Ingelligence (AAAI)*, 1998.

Reza Karimpour, Amineh Ghorbani, Azadeh Pishdad, Mitra Mohtarami, Abolfazl AleAhmad, Hadi Amiri, and Farhad Oroumchian. Improving Persian Information Retrieval Systems Using Stemming and Part of Speech Tagging. In *Proceedings of the 9th Cross-language Evaluation Forum Conference on Evaluating Systems for Multilingual and Multimodal Information Access*, CLEF 2008, pp 89–96, Berlin, Heidelberg, 2009. Springer-Verlag.

Chin-Yew Lin. Looking For A Few Good Metrics: Automatic Summarization Evaluation - How Many Samples Are Enough? In *Proceedings of NTCIR Workshop 4*, Tokyo, Japan, June 2004.

Annie Louis and Ani Nenkova. Automatic Summary Evaluation without Human Models. In *Proceedings of Empirical Methods in Natural Language Processing*, EMNLP 2009.

H. P. Luhn. The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development*, 2(2):159–165, April 1958.

Inderjeet Mani, Eric Bloedorn, and Barbara Gates. Using Cohesion and Coherence Models for Text Summarization. In *AAAI Symposium Technical Report SS-989-06*, AAAI Press, 69–76, 1998.

Inderjeet Mani, David House, Gary Klein, Lynette Hirschman, Therese Firmin, and Beth Sundheim. The TIPSTER SUMMAC Text Summarization Evaluation. In *Proceedings of European Association for Coputational Linguistics*, EACL 1999.

Inderjeet Mani. Summarization Evaluation: An Overview. In *Proceedings of the NTCIR Workshop*, Vol. 2, 2001.

Inderjeet Mani, Gary Klein, David House, Lynette Hirschman, Therese Firmin, and Beth Sundheim. SUMMAC: A Text Summarization Evaluation. *Natural Language Engineering*, 8(1) 43-68. March 2002.

Kathleen McKeown, Rebecca J. Passonneau, David K. Elson, Ani Nenkova, and Julia Hirschberg. Do Summaries Help? A Task-Based Evaluation of Multi-Document Summarization. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp 210-217. ACM 2005.

Anthony McCallum, Gerald Penn, Cosmin Munteanu, and Xiaodan Zhu. Ecological Validity and the Evaluation of Speech Summarization Quality. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*. 2012 Association for Computational Linguistics, Stroudsburg, PA, USA, 28-35.

Tatsunori Mori, Masanori Nozawa, and Yoshiaki Asada. Multi-Answer Focused Multi-Document Summarization Using a Question-Answering Engine. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, ACL 2004.

Gabriel Murray, Thomas Kleinbauer, Peter Poller, Tilman Becker, Steve Renals, and Jonathan Kilgour. Extrinsic Summarization Evaluation: A Decision Audit Task. *ACM Transactions on Speech and Language Processing*, 6(2) Article 2, October 2009.

Ani Nenkova and Kathleen McKeown. A Survey of Text Summarization Techniques. In C. C. Aggarwal and C. Zhai, editors, *Mining Text Data*, pp 43–76. Springer US, 2012.

Constantin Orăsan and Oana Andreea Chiorean. Evaluation of a Cross-Lingual Romanian-English Multi-Document Summariser. In *Proceedings of Language Resources and Evaluation Conference*, LREC 2008.

Jahna Otterbacher, Güneş Erkan, and Dragomir R Ravev. Using Random Walks for Question-focused Sentence Retrieval. In *Proceedings of Human Language Technology Conference on Empirical Methods in Natural Language Processing*, Vancouver, Canada, pp 915-922, EMNLP 2005.

Jahna Otterbacher, Güneş Erkan, and Dragomir R. Ravev. Biased LexRank: Passage Retrieval Using Random Walks With Question-Based Priors. In *Information Processing Management*, 45(1), January 2009, pp 42-54.

Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. An Assessment of the Accuracy of Automatic Evaluation in Summarization. In *Proceedings of the Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pp 1-9, Montréal, Canada, ACL 2012.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The Pagerank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

Prasad Pingali, Jagadeesh Jagarlamudi, and Vasudeva Varma. Experiments in Cross Language Query Focused Multi-Document Summarization In *Workshop on Cross Lingual Information Access Addressing the Information Need of Multilingual Societies*, IJCAI 2007.

Stacy F. President and Bonnie J. Dorr. Text Summarization Evaluation: Correlating Human Performance on an Extrinsic Task With Automatic Intrinsic Metrics. No. LAMP-TR-133. University of Maryland College Park Language and Media Processing Laboratory Institute for Advanced Computer Studies (UMIACS), 2006.

Dragomir R. Radev, Hongyan Jing, Malgorzata Styś, and Daniel Tam. Centroid-Based Summarization of Multiple Documents. In*Proceedings of Informaion Processing Management*, 40(6):919–938, Nov. 2004.

Stephen Robertson. Understanding Inverse Document Frequency: on Theoretical Arguments for IDF. *Journal of Documentation*, 60(5):503–520, 2004.

Stephen E. Robertson and Steve Walker. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *Proceedings of the 17th annual international ACM SIGIR*

conference on Research and development in information retrieval, pp 232–241. Springer-Verlag New York, Inc., 1994.

Gerard Salton and Chung-Shu Yang. On the Specification of Term Values in Automatic Indexing. *Journal of Documentation*, 29(4):351–372, 1973.

Anastasios Tombros and Mark Sanderson. Advantages of Query Biased Summaries in Information Retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp 2–10. ACM, 1998.

Xiaojun Wan and Jianguo Xiao. Graph-Based Multi-Modality Learning for Topic-Focused Multi-Document Summarization. In *Proceedings of the 21st international jont conference on Artifical intelligence* (IJCAI'09), San Francisco, CA, USA, 1586-1591.

Xiaojun Wan, Huiying Li, and Jianguo Xiao. Cross-Language Document Summarization Based on Machine Translation Quality Prediction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (ACL '10). Association for Computational Linguistics, Stroudsburg, PA, USA, 917-926.

Xiaojun Wan, Houping Jia, Shanshan Huang, and Jianguo Xiao. Summarizing the Differences in Multilingual News. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pp 735–744, New York, NY, USA, 2011. ACM.

Wenpeng Yin, Yulong Pei, Fan Zhang, and Lian'en Huang. SentTopic-MultiRank: A Novel Ranking Model for Multi-Document Summarization. In *Proceedings of COLING*, pages 2977–2992, 2012.

Junlin Zhang, Le Sun, and Jinming Min. Using the Web Corpus to Translate the Queries in Cross-Lingual Information Retrieval. In *Proceedings in 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering, 2005*, IEEE NLP-KE '05, pp 493–498, 2005.

Chinese Poetry Generation with Recurrent Neural Networks

Xingxing Zhang and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
x.zhang@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

We propose a model for Chinese poem generation based on recurrent neural networks which we argue is ideally suited to capturing poetic content and form. Our generator *jointly* performs content selection ("what to say") and surface realization ("how to say") by learning representations of individual characters, and their combinations into one or more lines as well as how these mutually reinforce and constrain each other. Poem lines are generated incrementally by taking into account the entire history of what has been generated so far rather than the limited horizon imposed by the previous line or lexical *n*-grams. Experimental results show that our model outperforms competitive Chinese poetry generation systems using both automatic and manual evaluation methods.

相 思
Missing You
红 豆 生 南 国， (* Z P P Z)
Red berries born in the warm southland.
春 来 发 几 **枝**？ (P P Z Z P)
How many branches flush in the spring?
愿 君 多 采 撷， (* P P Z Z)
Take home an armful, for my sake,
此 物 最 相 **思**。 (* Z Z P P)
As a symbol of our love.

Table 1: An example of a 5-char *quatrain* exhibiting one of the most popular tonal patterns. The tone of each character is shown at the end of each line (within parentheses); P and Z are shorthands for *Ping* and *Ze* tones, respectively; * indicates that the tone is not fixed and can be either. Rhyming characters are shown in boldface.

1 Introduction

Classical poems are a significant part of China's cultural heritage. Their popularity manifests itself in many aspects of everyday life, e.g., as a means of expressing personal emotion, political views, or communicating messages at festive occasions as well as funerals. Amongst the many different types of classical Chinese poetry, *quatrain* and *regulated verse* are perhaps the best-known ones. Both types of poem must meet a set of structural, phonological, and semantic requirements, rendering their composition a formidable task left to the very best scholars.

An example of a quatrain is shown in Table 1. Quatrains have four lines, each five or seven characters long. Characters in turn follow specific phonological patterns, within each line and across lines. For instance, the final characters in the second, fourth and (optionally) first line must rhyme, whereas there are no rhyming constraints for the third line. Moreover, poems must follow a prescribed tonal pattern. In traditional Chinese, every character has one tone, *Ping* (level tone) or *Ze* (downward tone). The poem in Table 1 exemplifies one of the most popular tonal patterns (Wang, 2002). Besides adhering to the above formal criteria, poems must exhibit concise and accurate use of language, engage the reader/hearer, stimulate their imagination, and bring out their feelings.

In this paper we are concerned with generating traditional Chinese poems automatically. Although computers are no substitute for poetic creativity, they can analyze very large online text repositories of poems, extract statistical patterns, maintain them in memory and use them to generate many possible variants. Furthermore, while amateur poets may struggle to remember and apply formal tonal and structural constraints, it is relatively straightforward for the machine to check

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 670–680,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

whether a candidate poem conforms to these requirements. Poetry generation has received a fair amount of attention over the past years (see the discussion in Section 2), with dozens of computational systems written to produce poems of varying sophistication. Beyond the long-term goal of building an autonomous intelligent system capable of creating meaningful poems, there are potential short-term applications for computer generated poetry in the ever growing industry of electronic entertainment and interactive fiction as well as in education. An assistive environment for poem composition could allow teachers and students to create poems subject to their requirements, and enhance their writing experience.

We propose a model for Chinese poem generation based on recurrent neural networks. Our generator *jointly* performs content selection ("what to say") and surface realization ("how to say"). Given a large collection of poems, we learn representations of individual characters, and their combinations into one or more lines as well as how these mutually reinforce and constrain each other. Our model generates lines in a poem probabilistically: it estimates the probability of the current line given the probability of all previously generated lines. We use a recurrent neural network to learn the representations of the lines generated so far which in turn serve as input to a recurrent language model (Mikolov et al., 2010; Mikolov et al., 2011b; Mikolov et al., 2011a) which generates the current line. In contrast to previous approaches (Greene et al., 2010; Jiang and Zhou, 2008), our generator makes no Markov assumptions about the dependencies of the words within a line and across lines.

We evaluate our approach on the task of quatrain generation (see Table 1 for a human-written example). Experimental results show that our model outperforms competitive Chinese poetry generation systems using both automatic and manual evaluation methods.

2 Related Work

Automated poetry generation has been a popular research topic over the past decades (see Colton et al. (2012) and the references therein). Most approaches employ templates to construct poems according to a set of constraints (e.g., rhyme, meter, stress, word frequency) in combination with corpus-based and lexicographic resources. For example, the Haiku poem generator presented in Wu et al. (2009) and Tosa et al. (2008) produces poems by expanding user queries with rules extracted from a corpus and additional lexical resources. Netzer et al. (2009) generate Haiku with Word Association Norms, Agirrezabal et al. (2013) compose Basque poems using patterns based on parts of speech and WordNet (Fellbaum, 1998), and Oliveira (2012) presents a generation algorithm for Portuguese which leverages semantic and grammar templates.

A second line of research uses genetic algorithms for poem generation (Manurung, 2003; Manurung et al., 2012; Zhou et al., 2010). Manurung et al. (2012) argue that at a basic level all (machine-generated) poems must satisfy the constraints of grammaticality (i.e., a poem must syntactically well-formed), meaningfulness (i.e., a poem must convey a message that is meaningful under some interpretation) and poeticness (i.e., a poem must exhibit features that distinguishes it from non-poetic text, e.g., metre). Their model generates several candidate poems and then uses stochastic search to find those which are grammatical, meaningful, and poetic.

A third line of research draws inspiration from statistical machine translation (SMT) and related text-generation applications such as summarization. Greene et al. (2010) infer meters (stressed/unstressed syllable sequences) from a corpus of poetic texts which they subsequently use for generation together with a cascade of weighted finite-state transducers interpolated with IBM Model 1. Jiang and Zhou (2008) generate Chinese couplets (two line poems) using a phrase-based SMT approach which translates the first line to the second line. He et al. (2012) extend this algorithm to generate four-line quatrains by sequentially translating the current line from the previous one. Yan et al. (2013) generate Chinese quatrains based on a query-focused summarization framework. Their system takes a few keywords as input and retrieves the most relevant poems from a corpus collection. The retrieved poems are segmented into their constituent terms which are then grouped into clusters. Poems are generated by iteratively selecting terms from clusters subject to phonological, structural, and coherence constraints.

Our approach departs from previous work in two important respects. Firstly, we model the tasks of surface realization and content selection jointly

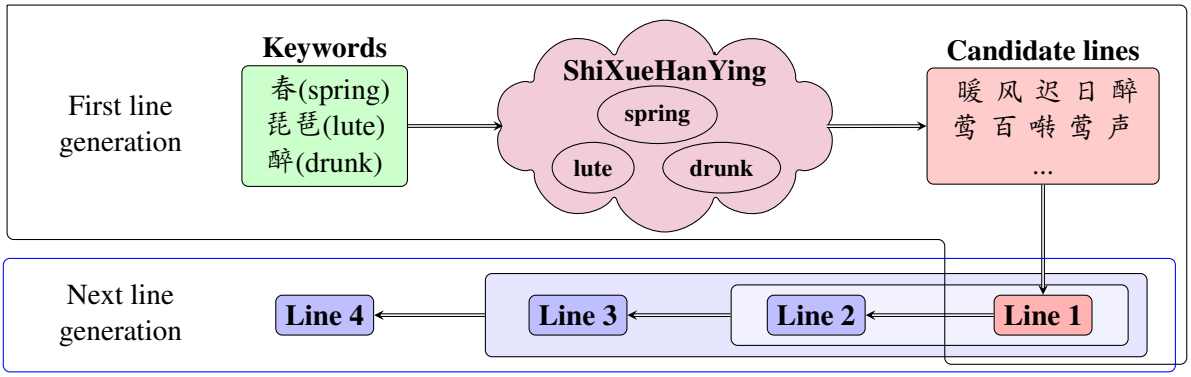


Figure 1: Poem generation with keywords *spring*, *lute*, and *drunk*. The keywords are expanded into phrases using a poetic taxonomy. Phrases are then used to generate the first line. Following lines are generated by taking into account the representations of all previously generated lines.

using recurrent neural networks. Structural, semantic, and coherence constraints are captured naturally in our framework, through learning the representations of individual characters and their combinations. Secondly, generation proceeds by taking into account multi-sentential context rather than the immediately preceding sentence. Our work joins others in using continuous representations to express the meaning of words and phrases (Socher et al., 2012; Mikolov et al., 2013) and how these may be combined in a language modeling context (Mikolov and Zweig, 2012). More recently, continuous translation models based on recurrent neural networks have been proposed as a means to map a sentence from the source language to sentences in the target language (Auli et al., 2013; Kalchbrenner and Blunsom, 2013). These models are evaluated on the task of rescoring n-best lists of translations. We use neural networks more directly to perform the actual poem generation task.

3 The Poem Generator

As common in previous work (Yan et al., 2013; He et al., 2012) we assume that our generator operates in an interactive context. Specifically, the user supplies keywords (e.g., *spring*, *lute*, *drunk*) highlighting the main concepts around which the poem will revolve. As illustrated in Figure 1, our generator expands these keywords into a set of related phrases. We assume the keywords are restricted to those attested in the *ShiXueHanYing* poetic phrase taxonomy (He et al., 2012; Yan et al., 2013). The latter contains 1,016 manual clusters of phrases (Liu, 1735); each cluster is labeled with a keyword id describing general poem-worthy topics. The generator creates the first line of the poem based on these keywords. Subsequent lines are generated based on all previously generated lines, subject to phonological (e.g., admissible tonal patterns) and structural constraints (e.g., whether the quatrain is five or seven characters long).

To create the first line, we select all phrases corresponding to the user's keywords and generate all possible combinations satisfying the tonal pattern constraints. We use a language model to rank the generated candidates and select the best-ranked one as the first line in the poem. In implementation, we employ a character-based recurrent neural network language model (Mikolov et al., 2010) interpolated with a Kneser-Ney trigram and find the n-best candidates with a stack decoder (see Section 3.5 for details). We then generate the second line based on the first one, the third line based on the first two lines, and so on. Our generation model computes the probability of line $S_{i+1} = w_1, w_2, \ldots, w_m$, given all previously generated lines $S_{1:i} (i \geq 1)$ as:

$$P(S_{i+1}|S_{1:i}) = \prod_{j=1}^{m-1} P(w_{j+1}|w_{1:j}, S_{1:i}) \qquad (1)$$

Equation (1), decomposes $P(S_{i+1}|S_{1:i})$ as the product of the probability of each character w_j in the current line given all previously generated characters $w_{1:j-1}$ and lines $S_{1:i}$. This means that $P(S_{i+1}|S_{1:i})$ is sensitive to previously generated content and currently generated characters.

The estimation of the term $P(w_{j+1}|w_{1:j}, S_{1:i})$ lies at the heart of our model. We learn representations for $S_{1:i}$, the context generated so far, using a recurrent neural network whose output

serves as input to a second recurrent neural network used to estimate $P(w_{j+1}|w_{1:j}, S_{1:i})$. Figure 2 illustrates the generation process for the $(j+1)$th character w_{j+1} in the $(i+1)$th line S_{i+1}. First, lines $S_{1:i}$ are converted into vectors $v_{1:i}$ with a *convolutional sentence model* (CSM; described in Section 3.1). Next, a *recurrent context model* (RCM; see Section 3.2) takes $v_{1:i}$ as input and outputs u_i^j, the representation needed for generating $w_{j+1} \in S_{i+1}$. Finally, $u_i^1, u_i^2, \ldots, u_i^j$ and the first j characters $w_{1:j}$ in line S_{i+1} serve as input to a *recurrent generation model* (RGM) which estimates $P(w_{j+1} = k|w_{1:j}, S_{1:i})$ with $k \in V$, the probability distribution of the $(j+1)$th character over all words in the vocabulary V. More formally, to estimate $P(w_{j+1}|w_{1:j}, S_{1:i})$ in Equation (1), we apply the following procedure:

$$v_i = \text{CSM}(S_i) \tag{2a}$$

$$u_i^j = \text{RCM}(v_{1:i}, j) \tag{2b}$$

$$P(w_{j+1}|w_{1:j}, S_{1:i}) = \text{RGM}(w_{1:j+1}, u_i^{1:j}) \tag{2c}$$

We obtain the probability of the $(i+1)$th sentence $P(S_{i+1}|S_{1:i})$, by running the RGM in (2c) above $m-1$ times (see also Equation (1)). In the following, we describe how the different components of our model are obtained.

3.1 Convolutional Sentence Model (CSM)

The CSM converts a poem line into a vector. In principle, any model that produces vector-based representations of phrases or sentences could be used (Mitchell and Lapata, 2010; Socher et al., 2012). We opted for the convolutional sentence model proposed in Kalchbrenner and Blunsom (2013) as it is n-gram based and does not make use of any parsing, POS-tagging or segmentation tools which are not available for Chinese poems. Their model computes a continuous representation for a sentence by sequentially merging neighboring vectors (see Figure 3).

Let V denote the character vocabulary in our corpus; $L \in \mathbb{R}^{q \times |V|}$ denotes a character embedding matrix whose columns correspond to character vectors (q represents the hidden unit size). Such vectors can be initialized randomly or obtained via a training procedure (Mikolov et al., 2013). Let w denote a character with index k; $e(w) \in \mathbb{R}^{|V| \times 1}$ is a vector with zero in all positions except $e(w)_k = 1$; $T^l \in \mathbb{R}^{q \times N^l}$ is the sentence representation in the lth layer, where N^l is the number of columns in the lth layer ($N^l = 1$ in the top layer); $C^{l,n} \in \mathbb{R}^{q \times n}$ is an array of weight matrices which compress neighboring n columns in the lth layer to one column in the $(l+1)$th layer. Given a sentence $S = w_1, w_2, \ldots, w_m$, the first layer is represented as:

$$\begin{aligned} T^1 &= [L \cdot e(w_1), L \cdot e(w_2), \ldots, L \cdot e(w_m)] \\ N^1 &= m \end{aligned} \tag{3}$$

The $(l+1)$th layer is then computed as follows:

$$\begin{aligned} T^{l+1}_{:,j} &= \sigma\Big(\sum_{i=1}^{n} T^l_{:,j+i-1} \odot C^{l,n}_{:,i}\Big) \\ N^{l+1} &= N^l - n + 1 \\ 1 &\le j \le N^{l+1} \end{aligned} \tag{4}$$

where T^l is the representation of the previous layer l, $C^{l,n}$ a weight matrix, $\odot$ element-wise vector product, and σ a non-linear function. We compress two neighboring vectors in the first two layers and three neighboring vectors in the remaining layers. Specifically, for quatrains with seven characters, we use $C^{1,2}$, $C^{2,2}$, $C^{3,3}$, $C^{4,3}$ to merge vectors in each layer (see Figure 3); and for quatrains with five characters we use $C^{1,2}$, $C^{2,2}$, $C^{3,3}$.

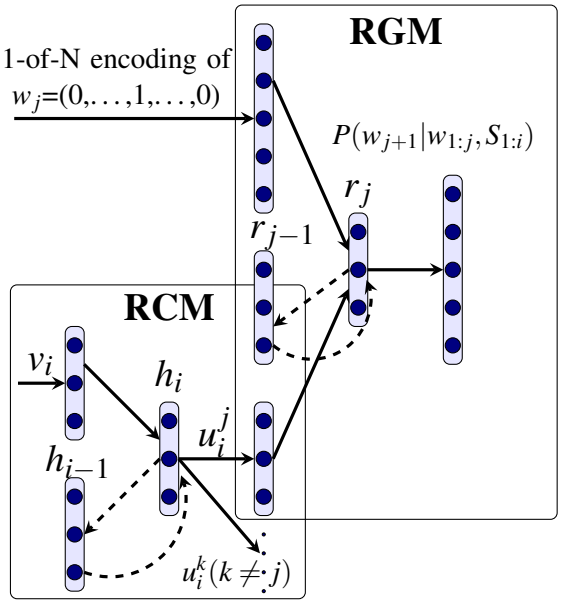


Figure 2: Generation of the $(j+1)$th character w_{j+1} in the $(i+1)$th line S_{i+1}. The recurrent context model (RCM) takes i lines as input (represented by vectors $v_1, \ldots, v_i$) and creates context vectors for the recurrent generation model (RGM). The RGM estimates the probability $P(w_{j+1}|w_{1:j}, S_{1:i})$.

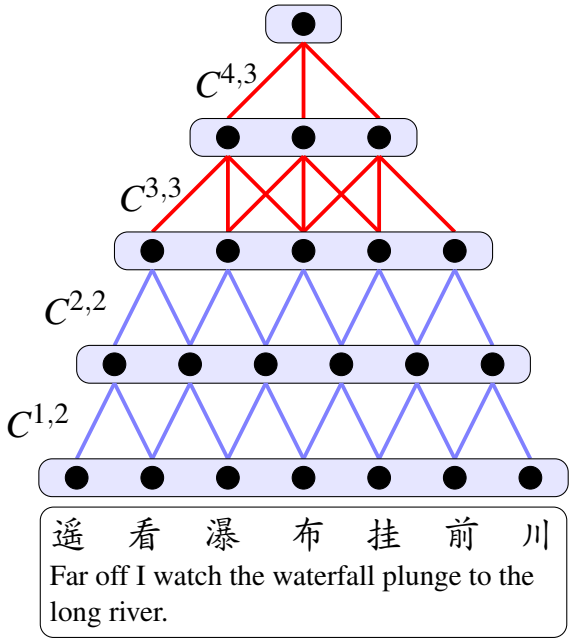


Figure 3: Convolutional sentence model for 7-char quatrain. The first layer has seven vectors, one for each character. Two neighboring vectors are merged to one vector in the second layer with weight matrix $C^{1,2}$. In other layers, either two or three neighboring vectors are merged.

3.2 Recurrent Context Model (RCM)

The RCM takes as input the vectors representing the i lines generated so far and reduces them to a single context vector which is then used to generate the next character (see Figure 2). We compress the i previous lines to one vector (the hidden layer) and then decode the compressed vector to different character positions in the current line. The output layer consists thus of several vectors (one for each position) connected together. This way, different aspects of the context modulate the generation of different characters.

Let $v_1, \ldots, v_i \quad (v_i \in \mathbb{R}^{q\times 1})$ denote the vectors of the previous i lines; $h_i \in \mathbb{R}^{q\times 1}$ is their compressed representation (hidden layer) which is obtained with matrix $M \in \mathbb{R}^{q\times 2q}$; matrix U_j decodes h_i to $u_i^j \in \mathbb{R}^{q\times 1}$ in the $(i+1)$th line. The computation of the RCM proceeds as follows:

$$\begin{aligned} h_0 &= \mathbf{0} \\ h_i &= \sigma(M \cdot \begin{bmatrix} v_i \\ h_{i-1} \end{bmatrix}) \\ u_i^j &= \sigma(U_j \cdot h_i) \qquad 1 \le j \le m-1 \end{aligned} \tag{5}$$

where σ is a non-linear function such as sigmoid and m the line length. Advantageously, lines in classical Chinese poems have a fixed length of five or seven characters. Therefore, the output layer of the recurrent context model only needs two weight matrices (one for each length) and the number of parameters still remains tractable.

3.3 Recurrent Generation Model (RGM)

As shown in Figure 2, the RGM estimates the probability distribution of the next character (over the entire vocabulary) by taking into account the context vector provided by the RCM and the 1-of-N encoding of the previous character. The RGM is essentially a recurrent neural network language model (Mikolov et al., 2010) with an auxiliary input layer, i.e., the context vector from the RCM. Similar strategies for encoding additional information have been adopted in related language modeling and machine translation work (Mikolov and Zweig, 2012; Kalchbrenner and Blunsom, 2013; Auli et al., 2013).

Let $S_{i+1} = w_1, w_2, \ldots, w_m$ denote the line to be generated. The RGM must estimate $P(w_{j+1}|w_{1:j}, S_{1:i})$, however, since the first i lines have been encoded in the context vector u_i^j, we compute $P(w_{j+1}|w_{1:j}, u_i^j)$ instead. Therefore, the probability $P(S_{i+1}|S_{1:i})$ becomes:

$$P(S_{i+1}|S_{1:i}) = \prod_{j=1}^{m-1} P(w_{j+1}|w_{1:j}, u_i^j) \tag{6}$$

Let $|V|$ denote the size of the character vocabulary. The RGM is specified by a number of matrices. Matrix $H \in \mathbb{R}^{q\times q}$ (where q represents the hidden unit size) transforms the context vector to a hidden representation; matrix $X \in \mathbb{R}^{q\times|V|}$ transforms a character to a hidden representation, matrix $R \in \mathbb{R}^{q\times q}$ implements the recurrent transformation and matrix $Y \in \mathbb{R}^{|V|\times q}$ decodes the hidden representation to weights for all words in the vocabulary. Let w denote a character with index k in V; $e(w) \in \mathbb{R}^{|V|\times 1}$ represents a vector with zero in all positions except $e(w)_k = 1$, r_j is the hidden layer of the RGM at step j, and y_{j+1} the output of the RGM, again at step j. The RGM proceeds as follows:

$$r_0 = \mathbf{0} \tag{7a}$$

$$r_j = \sigma(R \cdot r_{j-1} + X \cdot e(w_j) + H \cdot u_i^j) \tag{7b}$$

$$y_{j+1} = Y \cdot r_j \tag{7c}$$

where σ is a nonlinear function (e.g., sigmoid).

The probability of the $(j+1)$th word given the previous j words and the previous i lines is estimated by a *softmax* function:

$$P(w_{j+1} = k | w_{1:j}, u_i^j) = \frac{\exp(y_{j+1,k})}{\sum_{k=1}^{|V|} \exp(y_{j+1,k})} \quad (8)$$

We obtain $P(S_{i+1}|S_{1:i})$ by multiplying all the terms in the right hand-side of Equation (6).

3.4 Training

The objective for training is the cross entropy errors of the predicted character distribution and the actual character distribution in our corpus. An l_2 regularization term is also added to the objective. The model is trained with back propagation through time (Rumelhart et al., 1988) with sentence length being the time step. The objective is minimized by stochastic gradient descent. During training, the cross entropy error in the output layer of the RGM is back-propagated to its hidden and input layers, then to the RCM and finally to the CSM. The same number of hidden units ($q = 200$) is used throughout (i.e., in the RGM, RCM, and CSM). In our experiments all parameters were initialized randomly, with the exception of the word embedding matrix in the CSM which was initialized with *word2vec* embeddings (Mikolov et al., 2013) obtained from our poem corpus (see Section 4 for details on the data we used).

To speed up training, we employed word-classing (Mikolov et al., 2011b). To compute the probability of a character, we estimate the probability of its class and then multiply it by the probability of the character conditioned on the class. In our experiments we used 82 (square root of $|V|$) classes which we obtained by applying hierarchical clustering on character embeddings. This strategy outperformed better known frequency-based classing methods (Zweig and Makarychev, 2013) on our task.

Our poem generator models content selection and lexical choice and their interaction, but does not have a strong notion of local coherence, as manifested in poetically felicitous line-to-line transitions. In contrast, machine translation models (Jiang and Zhou, 2008) have been particularly successful at generating adjacent lines (couplets). To enhance coherence, we thus interpolate our model with two machine translation features (i.e., inverted phrase translation model feature and inverted lexical weight feature). Also note, that in our model surface generation depends on the last observed character and the state of the hidden layer before this observation. This way, there is no explicitly defined context, and history is captured implicitly by the recurrent nature of the model. This can be problematic for our texts which must obey certain stylistic conventions and sound poetic. In default of a better way of incorporating poeticness into our model, we further interpolate it with a language model feature (i.e., a Kneser-Ney trigram model).

Throughout our experiments, we use the RNNLM toolkit to train the character-based recurrent neural network language model (Mikolov et al., 2010). Kneser-Ney n-grams were trained with KenLM (Heafield, 2011).

3.5 Decoding

Our decoder is a stack decoder similar to Koehn et al. (2003). In addition, it implements the tonal pattern and rhyming constraints necessary for generating well-formed Chinese quatrains. Once the first line in a poem is generated, its tonal pattern is determined. During decoding, phrases violating this pattern are ignored. As discussed in Section 1, the final characters of the second and the fourth lines must rhyme. We thus remove during decoding fourth lines whose final characters do not rhyme with the second line. Finally, we use MERT training (Och, 2003) to learn feature weights for the decoder.

4 Experimental Design

Data We created a corpus of classical Chinese poems by collating several online resources: *Tang Poems*, *Song Poems*, *Song Ci*, *Ming Poems*, *Qing Poems*, and *Tai Poems*. The corpus consists of 284,899 poems in total. 78,859 of these are quatrains and were used for training and evaluating our model.[1] Table 2 shows the different partitions of this dataset (POEMLM) into training (QTRAIN)[2], validation (QVALID) and testing (QTEST). Half of the poems in QVALID and QTEST are 5-char quatrains and the other half are 7-char quatrains. All poems except QVALID

[1]The data used in our experiments can be downloaded from `http://homepages.inf.ed.ac.uk/mlap/index.php?page=resources`.

[2]Singleton characters in QTRAIN (6,773 in total) were replaced by <R> to reduce data sparsity.

	Poems	Lines	Characters
QTRAIN	74,809	299,236	2,004,460
QVALID	2,000	8,000	48,000
QTEST	2,050	8,200	49,200
POEMLM	280,849	2,711,034	15,624,283

Table 2: Dataset partitions of our poem corpus.

Models	Perplexity
KN5	172
RNNLM	145
RNNPG	**93**

Table 3: Perplexities for different models.

and QTEST were used for training the character-based language models (see row POEMLM in Table 2). We also trained *word2vec* embeddings on POEMLM. In our experiments, we generated quatrains following the eight most popular tonal patterns according to Wang (2002).

Perplexity Evaluation Evaluation of machine-generated poetry is a notoriously difficult task. Our evaluation studies were designed to assess Manurung et al.'s (2012) criteria of grammaticality, meaningfulness, and poeticness. As a sanity check, we first measured the perplexity of our model with respect to the goldstandard. Intuitively, a better model should assign larger probability (and therefore lower perplexity) to goldstandard poems.

BLEU-based Evaluation We also used BLEU to evaluate our model's ability to generate the second, third and fourth line given previous goldstandard lines. A problematic aspect of this evaluation is the need for human-authored references (for a partially generated poem) which we do not have. We obtain references automatically following the method proposed in He et al. (2012). The main idea is that if two lines share a similar topic, the lines following them can be each other's references. Let A and B denote two adjacent lines in a poem, with B following A. Similarly, let line B' follow line A' in another poem. If lines A and A' share some keywords in the same cluster in the *Shixuehanying* taxonomy, then B and B' can be used as references for both A and A'. We use this algorithm on the *Tang Poems* section of our corpus to build references for poems in the QVALID and QTEST data sets. Poems in QVALID (with auto-generated references) were used for MERT training and Poems in QTEST (with auto-generated references) were used for BLEU evaluation.

Human Evaluation Finally, we also evaluated the generated poems by eliciting human judgments. Specifically, we invited 30 experts[3] on Chinese poetry to assess the output of our generator (and comparison systems). These experts were asked to rate the poems using a 1–5 scale on four dimensions: *fluency* (is the poem grammatical and syntactically well-formed?), *coherence* (is the poem thematically and logically structured?), *meaningfulness* (does the poem convey a meaningful message to the reader?) and *poeticness* (does the text display the features of a poem?). We also asked our participants to evaluate system outputs by ranking the generated poems relative to each other as a way of determining overall poem quality (Callison-Burch et al., 2012).

Participants rated the output of our model and three comparison systems. These included He et al.'s (2012) SMT-based model (SMT), Yan et al.'s (2013) summarization-based system (SUM), and a random baseline which creates poems by randomly selecting phrases from the *Shixuehanying* taxonomy given some keywords as input. We also included human written poems whose content matched the input keywords. All systems were provided with the same keywords (i.e., the same cluster names in the *ShiXueHanYing* taxonomy). In order to compare all models on equal footing, we randomly sampled 30 sets of keywords (with three keywords in each set) and generated 30 quatrains for each system according to two lengths, namely 5-char and 7-char. Overall, we obtained ratings for 300 ($5\times30\times2$) poems.

5 Results

The results of our perplexity evaluation are summarized in Table 3. We compare our RNN-based poem generator (RNNPG) against Mikolov's (2010) recurrent neural network language model (RNNLM) and a 5-gram language model with Kneser-Ney smoothing (KN5). All models were trained on QTRAIN and tuned on QVALID. The perplexities were computed on QTEST. Note that

[3] 27 participants were professional or amateur poets and three were Chinese literature students who had taken at least one class on Chinese poetry composition.

Models	1 → 2		2 → 3		3 → 4		Average	
	5-char	7-char	5-char	7-char	5-char	7-char	5-char	7-char
SMT	0.0559	0.0906	0.0410	0.1837	0.0547	**0.1804**	0.0505	0.1516
RNNPG	**0.0561**	**0.1868**	**0.0515**	**0.2102**	**0.0572**	0.1800	**0.0549**	**0.1923**

Table 4: BLEU-2 scores on 5-char and 7-char quatrains. Given i goldstandard lines, BLEU-2 scores are computed for the next $(i+1)$th lines.

Models	Fluency		Coherence		Meaning		Poeticness		Rank	
	5-char	7-char	5-char	7-char	5-char	7-char	5-char	7-char	5-char	7-char
Random	2.52	2.18	2.22	2.16	2.02	1.93	1.77	1.71	0.31	0.26
SUM	1.97	1.91	2.08	2.33	1.84	1.98	1.66	1.73	0.25	0.22
SMT	2.81	3.01	2.47	2.76	2.33	2.73	2.08	2.36	0.43	0.53
RNNPG	$\mathbf{4.01}^{**}$	$\mathbf{3.44}^{*}$	$\mathbf{3.18}^{**}$	$\mathbf{3.12}^{*}$	$\mathbf{3.20}^{**}$	**3.02**	$\mathbf{2.80}^{**}$	$\mathbf{2.68}^{*}$	$\mathbf{0.73}^{**}$	$\mathbf{0.64}^{*}$
Human	4.31^{+}	4.19^{++}	3.81^{++}	4.00^{++}	3.61^{+}	3.91^{++}	3.29^{++}	3.49^{++}	0.79	0.84^{++}

Table 5: Mean ratings elicited by humans on 5-char and 7-char quatrains. Diacritics ** ($p < 0.01$) and * ($p < 0.05$) indicate our model (RNNPG) is significantly better than all other systems except Human. Diacritics $^{++}$ ($p < 0.01$) and $^{+}$ ($p < 0.05$) indicate Human is significantly better than all other systems.

the RNNPG estimates the probability of a poem line given at least one previous line. Therefore, the probability of a quatrain assigned by the RNNPG is the probability of the last three lines. For a fair comparison, RNNLM and KN5 only leverage the last three lines of each poem during training, validation and testing. The results in Table 3 indicate that the generation ability of the RNNPG is better than KN5 and RNNLM. Note that this perplexity-style evaluation is not possible for models which cannot produce probabilities for gold standard poems. For this reason, other related poem generators (Yan et al., 2013; He et al., 2012) are not included in the table.

The results of our evaluation using BLEU-2 are summarized in Table 4. Here, we compare our system against the SMT-based poem generation model of He et al. (2012).[4] Their system is a linear combination of two translation models (one with five features and another one with six). Our model uses three of their features, namely the inverted phrase translation model feature, the lexical weight feature, and a Kneser-Ney trigram feature. Unfortunately, it is not possible to evaluate Yan et al.'s (2013) summarization-based system with BLEU, as it creates poems as a whole and there is no obvious way to generate next lines with their algorithm. The BLEU scores in Table 4 indicate that, given the same context lines, the RNNPG is better than SMT at generating what to say next. BLEU scores should be, however, viewed with some degree of caution. Aside from being an approximation of human judgment (Callison-Burch et al., 2012), BLEU might be unnecessarily conservative for poem composition which by its very nature is a creative endeavor.

The results of our human evaluation study are shown in Table 5. Each column reports mean ratings for a different dimension (e.g., fluency, coherence). Ratings for 5-char and 7-char quatrains are shown separately. The last column reports rank scores for each system (Callison-Burch et al., 2012). In a ranked list of N items ($N = 5$ here), the score of the ith ranked item is $\frac{(N-i)}{(N-1)}$. The numerator indicates how many times a systems won in pairwise comparisons, while the denominator normalizes the score.

With respect to 5-char quatrains, RNNPG is significantly better than Random, SUM and SMT on fluency, coherence, meaningfulness, poeticness and ranking scores (using a t-test). On all dimensions, human-authored poems are rated as significantly better than machine-generated ones, with the exception of overall ranking. Here, the difference between RNNPG and Human is not significant. We obtain similar results with 7-char quatrains. In general, RNNPG seems to perform better on the shorter poems. The mean ratings

[4] Our re-implementation of their system delivered very similar scores to He et al. (2012). For example, we obtained an average BLEU-1 of 0.167 for 5-char quatrains and 0.428 for 7-char quatrains compared to their reported scores of 0.141 and 0.380, respectively.

白鹭窥鱼立, Egrets stood, peeping fishes. 青山照水开. Water was still, reflecting mountains. 夜来风不动, The wind went down by nightfall, 明月见楼台. as the moon came up by the tower.	满怀风月一枝春, Budding branches are full of romance. 未见梅花亦可人. Plum blossoms are invisible but adorable. 不为东风无此客, With the east wind comes Spring. 世间何处是前身. Where on earth do I come from?

Table 6: Example output produced by our model (RNNPG).

are higher and the improvements over other systems are larger. Also notice, that the score margins between the human- and machine-written poems become larger for 7-char quatrains. This indicates that the composition of 7-char quatrains is more difficult compared to 5-char quatrains. Table 6 shows two example poems (5-char and 7-char) produced by our model which received high scores with respect to poeticness.

Interestingly, poems generated by SUM[5] are given ratings similar to Random. In fact SUM is slightly worse (although not significantly) than Random on all dimensions, with the exception of coherence. In the human study reported in Yan et al. (2013), SUM is slightly better than SMT. There are several reasons for this discrepancy. We used a more balanced experimental design: all systems generated poems from the same keywords which were randomly chosen. We used a larger dataset to train the SMT model compared to Yan et al. (284,899 poems vs 61,960). The Random baseline is not a straw-man; it selects phrases from a taxonomy of meaningful clusters edited by humans and closely related to the input keywords.

6 Conclusions

In this paper we have presented a model for Chinese poem generation based on recurrent neural networks. Our model jointly performs content selection and surface realization by learning representations of individual characters and their combinations within and across poem lines. Previous work on poetry generation has mostly leveraged contextual information of limited length (e.g., one sentence). In contrast, we introduced two recurrent neural networks (the recurrent context model and recurrent generation model) which naturally capture multi-sentential content. Experimental results show that our model yields high quality poems compared to the state of the art. Perhaps unsurprisingly, our human evaluation study revealed that machine-generated poems lag behind human-generated ones. It is worth bearing in mind that poetry composition is a formidable task for humans, let alone machines. And that the poems against which our output was compared have been written by some of the most famous poets in Chinese history!

Avenues for future work are many and varied. We would like to generate poems across different languages and genres (e.g., Engish sonnets or Japanese haiku). We would also like to make the model more sensitive to line-to-line transitions and stylistic conventions by changing its training objective to a combination of cross-entropy error and BLEU score. Finally, we hope that some of the work described here might be of relevance to other generation tasks such as summarization, concept-to-text generation, and machine translation.

[5]We made a good-faith effort to re-implement their poem generation system. We are grateful to Rui Yan for his help and technical advice.

Acknowledgments

We would like to thank Eva Halser for valuable discussions on the machine translation baseline. We are grateful to the 30 Chinese poetry experts for participating in our rating study. Thanks to Gujing Lu, Chu Liu, and Yibo Wang for their help with translating the poems in Table 6 and Table 1.

References

Manex Agirrezabal, Bertol Arrieta, Aitzol Astigarraga, and Mans Hulden. 2013. POS-Tag Based Poetry Generation with WordNet. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 162–166, Sofia, Bulgaria.

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint Language and Translation

Modeling with Recurrent Neural Networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, Washington, USA.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of the 7th Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada.

Simon Colton, Jacob Goodwin, and Tony Veale. 2012. Full-FACE Poetry Generation. In *Proceedings of the International Conference on Computational Creativity*, pages 95–102, Dublin, Ireland.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic Analysis of Rhythmic Poetry with Applications to Generation and Translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 524–533, Cambridge, MA.

Jing He, Ming Zhou, and Long Jiang. 2012. Generating Chinese Classical Poems with Statistical Machine Translation Models. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1650–1656, Toronto, Canada.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.

Long Jiang and Ming Zhou. 2008. Generating Chinese Couplets using a Statistical MT Approach. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 377–384, Manchester, UK, August.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54, Edmonton, Canada.

Wenwei Liu. 1735. *ShiXueHanYing*.

Ruli Manurung, Graeme Ritchie, and Henry Thompson. 2012. Using Genetic Algorithms to Create Meaningful Poetic Text. *Journal of Experimental Theoretical Artificial Intelligence*, 24(1):43–64.

Ruli Manurung. 2003. *An Evolutionary Algorithm Approach to Poetry Generation*. Ph.D. thesis, University of Edinburgh.

Tomas Mikolov and Geoffrey Zweig. 2012. Context Dependent Recurrent Neural Network Language Model. In *Proceedings of 2012 IEEE Workshop on Spoken Language Technology*, pages 234–239, Miami, Florida.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent Neural Network based Language Model. In *Proceedings of INTERSPEECH*, pages 1045–1048, Makuhari, Japan.

Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky. 2011a. Strategies for Training Large Scale Neural Network Language Models. In *Proceedings of ASRU 2011*, pages 196–201, Hilton Waikoloa Village, Big Island, Hawaii, US.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, JH Cernocky, and Sanjeev Khudanpur. 2011b. Extensions of Recurrent Neural Network Language Model. In *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5528–5531, Prague, Czech Republic.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, Lake Tahoe, Nevada, United States.

Jeff Mitchell and Mirella Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388–1439.

Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. 2009. Gaiku: Generating Haiku with Word Associations Norms. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 32–39, Boulder, Colorado.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Hugo Gonçalo Oliveira. 2012. PoeTryMe: a Versatile Platform for Poetry Generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21.

David Rumelhart, Geoffrey Hinton, and Ronald Williams. 1988. *Learning Representations by Back-propagating Errors*. MIT Press, Cambridge, MA, USA.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea.

Naoko Tosa, Hideto Obara, and Michihiko Minoh. 2008. Hitch Haiku: An Interactive Supporting System for Composing Haiku Poem How I Learned to Love the Bomb: Defcon and the Ethics of Computer Games. In *Proceedings of the 7th International Conference on Entertainment Computing*, pages 209–216, Pittsburgh, PA.

Li Wang. 2002. *A Summary of Rhyming Constraints of Chinese Poems (Shi Ci Ge Lv Gai Yao)*. Beijing Press, 2002.

Xiaofeng Wu, Naoko Tosa, and Ryohei Nakatsu. 2009. New Hitch Haiku: An Interactive Renku Poem Composition Supporting Tool Applied for Sightseeing Navigation System. In *Proceedings of the 8th International Conference on Entertainment Computing*, pages 191–196, Paris, France.

Rui Yan, Han Jiang, Mirella Lapata, Shou-De Lin, Xueqiang Lv, and Xiaoming Li. 2013. I, Poet: Automatic Chinese Poetry Composition Through a Generative Summarization Framework Under Constrained Optimization. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 2197–2203, Beijing, China.

Cheng-Le Zhou, Wei You, and Xiaojun Ding. 2010. Genetic Algorithm and its Implementation of Automatic Generation of Chinese SongCi. *Journal of Software*, pages 427–437.

Geoffrey Zweig and Konstantin Makarychev. 2013. Speed Regularization and Optimality in Word Classing. In *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 8237–8241, Florence, Italy.

Fear the REAPER: A System for Automatic Multi-Document Summarization with Reinforcement Learning

Cody Rioux
University of Lethbridge
Lethbridge, AB, Canada
cody.rioux@uleth.ca

Sadid A. Hasan
Philips Research North America
Briarcliff Manor, NY, USA
sadid.hasan@philips.com

Yllias Chali
University of Lethbridge
Lethbridge, AB, Canada
chali@cs.uleth.ca

Abstract

This paper explores alternate algorithms, reward functions and feature sets for performing multi-document summarization using reinforcement learning with a high focus on reproducibility. We show that ROUGE results can be improved using a unigram and bigram similarity metric when training a learner to select sentences for summarization. Learners are trained to summarize document clusters based on various algorithms and reward functions and then evaluated using ROUGE. Our experiments show a statistically significant improvement of 1.33%, 1.58%, and 2.25% for ROUGE-1, ROUGE-2 and ROUGE-L scores, respectively, when compared with the performance of the state of the art in automatic summarization with reinforcement learning on the DUC2004 dataset. Furthermore query focused extensions of our approach show an improvement of 1.37% and 2.31% for ROUGE-2 and ROUGE-SU4 respectively over query focused extensions of the state of the art with reinforcement learning on the DUC2006 dataset.

1 Introduction

The multi-document summarization problem has received much attention recently (Lyngbaek, 2013; Sood, 2013; Qian and Liu, 2013) due to its ability to reduce large quantities of text to a human processable amount as well as its application in other fields such as question answering (Liu et al., 2008; Chali et al., 2009a; Chali et al., 2009b; Chali et al., 2011b). We expect this trend to further increase as the amount of linguistic data on the web from sources such as social media, wikipedia, and online newswire increases. This paper focuses specifically on utilizing reinforcement learning (Sutton and Barto, 1998; Szepesv, 2009) to create a policy for summarizing clusters of multiple documents related to the same topic.

The task of extractive automated multi-document summarization (Mani, 2001) is to select a subset of textual units, in this case sentences, from the source document cluster to form a summary of the cluster in question. This extractive approach allows the learner to construct a summary without concern for the linguistic quality of the sentences generated, as the source documents are assumed to be of a certain linguistic quality. This paper aims to expand on the techniques used in Ryang and Abekawa (2012) which uses a reinforcement learner, specifically $TD(\lambda)$, to create summaries of document clusters. We achieve this through introducing a new algorithm, varying the feature space and utilizing alternate reward functions.

The $TD(\lambda)$ learner used in Ryang and Abekawa (2012) is a very early reinforcement learning implementation. We explore the option of leveraging more recent research in reinforcement learning algorithms to improve results. To this end we explore the use of $SARSA$ which is a derivative of $TD(\lambda)$ that models the action space in addition to the state space modelled by $TD(\lambda)$. Furthermore we explore the use of an algorithm not based on temporal difference methods, but instead on policy iteration techniques. Approximate Policy Iteration (Lagoudakis and Parr, 2003) generates a policy, then evaluates and iterates until convergence.

The reward function in Ryang and Abekawa (2012) is a delayed reward based on $tf*idf$ values. We further explore the reward space by introducing similarity metric calculations used in ROUGE (Lin, 2004) and base our ideas on Saggion et al. (2010). The difference between immediate rewards and delayed rewards is that the learner re-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 681–690,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

ceives immediate feedback at every action in the former and feedback only at the end of the episode in the latter. We explore the performance difference of both reward types. Finally we develop query focused extensions to both reward functions and present their results on more recent Document Understanding Conference (DUC) datasets which ran a query focused task.

We first evaluate our systems using the DUC2004 dataset for comparison with the results in Ryang and Abekawa (2012). We then present the results of query focused reward functions against the DUC2006 dataset to provide reference with a more recent dataset and a more recent task, specifically a query-focused summarization task. Evaluations are performed using ROUGE for ROUGE-1, ROUGE-2 and ROUGE-L values for general summarization, while ROUGE-2 and ROUGE-SU4 is used for query-focused summarization. Furthermore we selected a small subset of query focused summaries to be subjected to human evaluations and present the results.

Our implementation is named REAPER (**R**elatedness-focused **E**xtractive **A**utomatic summary **P**reparation **E**xploiting **R**einfocement learning) thusly for its ability to harvest a document cluster for ideal sentences for performing the automatic summarization task. REAPER is not just a reward function and feature set, it is a full framework for implementing summarization tasks using reinforcement learning and is available online for experimentation.[1] The primary contributions of our experiments are as follows:

- Exploration of $TD(\lambda)$, $SARSA$ and Approximate Policy Iteration.
- Alternate REAPER reward function.
- Alternate REAPER feature set.
- Query focused extensions of automatic summarization using reinforcement learning.

2 Previous Work and Motivation

Previous work using reinforcement learning for natural language processing tasks (Branavan et al., 2009; Wan, 2007; Ryang and Abekawa, 2012; Chali et al., 2011a; Chali et al., 2012) inspired us to use a similar approach in our experiments. Ryang and Abekawa (2012) implemented a reinforcement learning approach to multi-document summarization which they named Automatic Summarization using Reinforcement Learning (ASRL). ASRL uses TD(λ) to learn and then execute a policy for summarizing a cluster of documents. The algorithm performs N summarizations from a blank state to termination, updating a set of state-value predictions as it does so. From these N episodes a policy is created using the estimated state-value pairs, this policy greedily selects the best action until the summary enters its terminal state. This summary produced is the output of ASRL and is evaluated using ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004). The results segment of the paper indicates that ASRL outperforms greedy and integer linear programming (ILP) techniques for the same task.

There are two notable details that provide the motivation for our experiments; TD(λ) is relatively old as far as reinforcement learning (RL) algorithms are concerned, and the optimal ILP did not outperform ASRL using the same reward function. The intuition gathered from this is that if the optimal ILP algorithm did not outperform the suboptimal ASRL on the ROUGE evaluation, using the same reward function, then there is clearly room for improvement in the reward function's ability to accurately model values in the state space. Furthermore one may expect to achieve a performance boost exploiting more recent research by utilizing an algorithm that intends to improve upon the concepts on which $TD(\lambda)$ is based. These provide the motivation for the remainder of the research preformed.

Query focused multi-document summarization (Li and Li, 2013; Chali and Hasan, 2012b; Yin et al., 2012; Wang et al., 2013) has recently gained much attention due to increasing amounts of textual data, as well as increasingly specific user demands for extracting information from said data. This is reflected in the query focused tasks run in the Document Understanding Conference (DUC) and Text Analysis Conference (TAC) over the past decade. This has motivated us to design and implement query focused extensions to these reinforcement learning approaches to summarization.

There has been some research into the effects of sentence compression on the output of automatic summarization systems (Chali and Hasan, 2012a), specifically the evaluation results garnered from compressing sentences before evaluation (Qian and Liu, 2013; Lin and Rey, 2003; Ryang and

[1] https://github.com/codyrioux/REAPER

Abekawa, 2012). However Ryang and Abekawa (2012) found this technique to be ineffective in improving ROUGE metrics using a similar reinforcement learning approach to this paper, as a result we will not perform any further exploration into the effects of sentence compression.

3 Problem Definition

We use an identical problem definition to Ryang and Abekawa (2012). Assume the given cluster of documents is represented as a set of textual units $D = \{x_1, x_2, \cdots, x_n\}$ where $|D| = n$ and x_i represents a single textual unit. Textual units for the purposes of this experiment are the individual sentences in the document cluster, that is $D = D_1 \cup D_2 \cup \cdots \cup D_m$ where m is the number of documents in the cluster and each D_i represents a document.

The next necessary component is the score function, which is to be used as the reward for the learner. The function $score(s)$ can be applied to any $s \subset D$. s is a summary of the given document or cluster of documents.

Given these parameters, and a length limitation k we can define an optimal summary s^* as:

$$s^* = argmax\ score(s) \quad where\ s \subset D\ and\ length(s) \leq k \quad (1)$$

It is the objective of our learner to create a policy that produces the optimal summary for its provided document cluster D. Henceforth the length limitations used for general summarization will be 665 bytes, and query focused summarization will use 250 words. These limitations on summary length match those set by the Document Understanding Conferences associated with the dataset utilized in the respective experiments.

4 Algorithms

$TD(\lambda)$ and $SARSA$ (Sutton and Barto, 1998) are temporal difference methods in which the primary difference is that $TD(\lambda)$ models state value predictions, and $SARSA$ models state-action value predictions. Approximate Policy Iteration (API) follows a different paradigm by iteratively improving a policy for a markov decision process until the policy converges.

4.1 TD(λ)

In the ASRL implementation of $TD(\lambda)$ the learning rate α_k and temperature τ_k decay as learning progresses with the following equations with k set to the number of learning episodes that had taken place.

$$\alpha_k = 0.001 \cdot 101/(100 + k^{1.1}) \quad (2)$$

$$\tau_k = 1.0 * 0.987^{k-1} \quad (3)$$

One can infer from the decreasing values of α_k that as the number of elapsed episodes increases the learner adjusts itself at a smaller rate. Similarly as the temperature τ_k decreases the action selection policy becomes greedier and thus performs less exploration, this is evident in (5) below.

Note that unlike traditional $TD(\lambda)$ implementations the eligibility trace e resets on every episode. The reasons for this will become evident in the experiments section of the paper in which $\lambda = 1, \gamma = 1$ and thus there is no decay during an episode and complete decay after an episode. The same holds true for $SARSA$ below.

The action-value estimation $Q(s, a)$ is approximated as:

$$Q(s, a) = r + \gamma V(s') \quad (4)$$

The policy is implemented as such:

$$policy(a|s; \theta; \tau) = \frac{e^{Q(s,a)/\tau}}{\sum_{a \in A} e^{Q(s,a)/\tau}} \quad (5)$$

Actions are selected probabilistically using softmax selection (Sutton and Barto, 1998) from a Boltzmann distribution. As the value of τ approaches 0 the distribution becomes greedier.

4.2 $SARSA$

$SARSA$ is implemented in a very similar manner and shares α_k, τ_k, $\phi(s)$, m, and $policy(s)$ with the $TD(\lambda)$ implementation above. $SARSA$ is also a temporal difference algorithm and thus behaves similarly to $TD(\lambda)$ with the exception that values are estimated not only on the state s but a state-action pair $[s, a]$.

4.3 Approximate Policy Iteration

The third algorithm in our experiment uses Approximate Policy Iteration (Lagoudakis and Parr, 2003) to implement a reinforcement learner. The

novelty introduced by (Lagoudakis and Parr, 2003) is that they eschew standard representations for a policy and instead use a classifier to represent the current policy π. Further details on the algorithm can be obtained from Lagoudakis and Parr (2003).

5 Experiments

Our state space S is represented simply as a three-tuple $[s\ a\ f]$ in which s is the set of textual units (sentences) that have been added to the summary, a is a sequence of actions that have been performed on the summary and f is a boolean with value 0 representing non-terminal states and 1 representing a summary that has been terminated.

The individual units in our action space are defined as [:insert x_i] where x_i is a textual unit as described earlier, let us define D_i as the set [:insert x_i] for all $x_i \in D$ where D is the document set. We also have one additional action [:finish] and thus we can define our action space.

$$A = D_i \cup \{[:\ finish]\} \tag{6}$$

The actions eligible to be executed on any given state s is defined by a function $actions(A, s)$:

$$actions(A, s) = \begin{cases} [:\ finish] & \text{if } length(s) > k \\ A - a_t & otherwise \end{cases} \tag{7}$$

The state-action transitions are defined below:

$$[s_t\ , a_t\ , 0] \xrightarrow{a=insert x_i} [s_t \cup x_i\ , a_t \cup a\ , 0] \tag{8}$$

$$[s_t\ , a_t\ , 0] \xrightarrow{:finish} [s_t\ , a_t\ , 1] \tag{9}$$

$$[s_t\ , a_{t+1}\ , 1] \xrightarrow{any} [s_t\ , a_t\ , 1] \tag{10}$$

Insertion adds both the content of the textual unit x_i to the set s as well as the action itself to set a. Conversely finishing does not alter s or a but it flips the f bit to on. Notice from (10) that once a state is terminal any further actions have no effect.

5.1 Feature Space

We present an alternate feature set called REAPER feature set based on the ideas presented in Ryang and Abekawa (2012). Our proposed feature set follows a similar format to the previous one but depends on the presence of top **bigrams** instead of $tf * idf$ words.

- One bit $b \in 0, 1$ for each of the top n **bigrams** (Manning and Schütze, 1999) present in the summary.
- Coverage ratio calculated as the sum of the bits in the previous feature divided by n.
- Redundancy Ratio calculated as the number of redundant times a bit in the first feature is flipped on, divided by n.
- Length Ratio calculated as $length(s)/k$ where k is the length limit.
- Longest common subsequence length.
- Length violation bit. Set to 1 if $length(s) > k$

Summaries which exceed the length limitation k are subject to the same reduction as the ASRL feature set (Ryang and Abekawa, 2012) to an all zero vector with the final bit set to one.

5.2 Reward Function

Our reward function (termed as REAPER reward) is based on the n-gram concurrence score metric, and the longest-common-subsequence recall metric contained within ROUGE (Lin, 2004).

$$reward(s) = \begin{cases} -1, & \text{if } length(s) > k \\ score(s) & \text{if } s \text{ is terminal} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Where $score$ is defined identically to ASRL, with the exception that Sim is a new equation based on ROUGE metrics.

$$score(s) = \sum_{x_i \in S} \lambda_s Rel(x_i) - \sum_{x_i, x_j \in S, i<j} (1 - \lambda_s) Red(x_i, x_j) \tag{12}$$

$$Rel(x_i) = Sim(x_i, D) + Pos(x_i)^{-1} \tag{13}$$

$$Red(x_i, x_j) = Sim(x_i, x_j) \tag{14}$$

Ryang and Abekawa (2012) experimentally determined a value of 0.9 for the λ_s parameter. That value is used herein unless otherwise specified. Sim has been redefined as:

$$\begin{aligned} Sim(s) =& 1 * ngco(1, D, s) + \\ & 4 * ngco(2, D, s) + \\ & 1 * ngco(3, D, s) + \\ & 1 * ngco(4, D, s) + \\ & 1 * rlcs(D, s) \end{aligned} \quad (15)$$

and $ngco$ is the ngram co-occurence score metric as defined by Lin (2004).

$$ngco(n, D, s) = \frac{\sum_{r \in D} \sum_{ngram \in r} Count_{match}(ngram)}{\sum_{S_r \in D} \sum_{ngram \in r} Count(ngram)} \quad (16)$$

Where n is the n-gram count for example 2 for bigrams, D is the set of documents, and s is the summary in question. $Count_{match}$ is the maximum number of times the ngram occurred in either D or s.

The $rlcs(R, S)$ is also a recall oriented measure based on longest common subsequence (Hirschberg, 1977). Recall was selected as DUC2004 tasks favoured a β value for F-Measure (Lin, 2004) high enough that only recall would be considered. lcs is the longest common subsequence, and $length(D)$ is the total number of tokens in the reference set D.

$$rlcs(D, s) = \frac{lcs(D, s)}{length(D)} \quad (17)$$

We are measuring similarity between sentences and our entire reference set, and thusly our D is the set of documents defined in section 3. This is also a delayed reward as the provided reward is zero until the summary is terminal.

5.2.1 Query Focused Rewards

We have proposed an extension to both reward functions to allow for query focused (QF) summarization. We define a function $score'$ which aims to balance the summarization abilities of the reward with a preference for selecting textual units related to the provided query q. Both ASRL and REAPER $score$ functions have been extended in the following manner where Sim is the same similarity functions used in equation (13) and (15).

$$score'(q, s) = \beta Sim(q, s) + (1 - \beta) score(s) \quad (18)$$

The parameter β is a balancing factor between query similarity and overall summary score in which $0 <= \beta <= 1$, we used an arbitrarily chosen value of 0.9 in these experiments. In the case of ASRL the parameter q is the vectorized version of the query function with $tf * idf$ values, and for Sim q is a sequence of tokens which make up the query, stemmed and stop-words removed.

5.2.2 Immediate Rewards

Finally we also employ immediate versions of the reward functions which behave similarly to their delayed counterparts with the exception that the score is always provided to the caller regardless of the terminal status of state s.

$$reward(s) = \begin{cases} -1, & \text{if } length(s) > k \\ score(s) & \text{otherwise} \end{cases} \quad (19)$$

6 Results

We first present results[2] of our experiments, specifying parameters, and withholding discussion until the following section. We establish a benchmark using ASRL and other top-scoring summarization systems compared with REAPER using ROUGE. For generic multi-document summarization we run experiments on all 50 document clusters, each containing 10 documents, of DUC2004 task 2 with parameters for REAPER and ASRL fixed at $\lambda = 1$, $\gamma = 1$, and $k = 665$. Sentences were stemmed using a Porter Stemmer (Porter, 1980) and had the ROUGE stop word set removed. All summaries were processed in this manner and then projected back into their original (unstemmed, with stop-words) state and output to disk.

Config	R-1	R-2	R-L
REAPER	0.40339	0.11397	0.36574
ASRL	0.39013	0.09479	0.33769
MCKP	0.39033	0.09613	0.34225
PEER65	0.38279	0.09217	0.33099
ILP	0.34712	0.07528	0.31241
GREEDY	0.30618	0.06400	0.27507

Table 1: Experimental results with ROUGE-1, ROUGE-2 and ROUGE-L scores on DUC2004.

[2]ROUGE-1.5.5 run with -m -s -p 0

Table 1 presents results for REAPER, ASRL (Ryang and Abekawa, 2012), MCKP (Takamura and Okumura, 2009), PEER65 (Conroy et al., 2004) , and GREEDY (Ryang and Abekawa, 2012) algorithms on the same task. This allows us to make a direct comparison with the results of Ryang and Abekawa (2012).

REAPER results are shown using the $TD(\lambda)$ algorithm, REAPER reward function, and ASRL feature set. This is to establish the validity of the reward function holding all other factors constant. REAPER results for ROUGE-1, ROUGE-2 and ROUGE-L are statistically significant compared to the result set presented in Table 2 of Ryang and Abekawa (2012) using $p < 0.01$.

Run	R-1	R-2	R-L
REAPER 0	0.39536	0.10679	0.35654
REAPER 1	0.40176	0.11048	0.36450
REAPER 2	0.39272	0.11171	0.35766
REAPER 3	0.39505	0.11021	0.35972
REAPER 4	0.40259	0.11396	0.36539
REAPER 5	0.40184	0.11306	0.36391
REAPER 6	0.39311	0.10873	0.35481
REAPER 7	0.39814	0.11001	0.35786
REAPER 8	0.39443	0.10740	0.35586
REAPER 9	0.40233	0.11397	0.36483
Average	0.39773	0.11063	0.36018

Table 2: REAPER run 10 times on the DUC2004.

We present the results of 10 runs of REAPER, with REAPER feature set.. As with ASRL, REAPER does not converge on a stable solution which is attributable to the random elements of $TD(\lambda)$. Results in all three metrics are again statistically significant compared to ASRL results presented in the Ryang and Abekawa (2012) paper. All further REAPER experiments use the bigram oriented feature space.

Reward	R-1	R-2	R-L
$Delayed$	0.39773	0.11397	0.36018
$Immediate$	0.32981	0.07709	0.30003

Table 3: REAPER with delayed and immediate rewards on DUC2004.

Table 3 shows the performance difference of REAPER when using a delayed and immediate reward. The immediate version of REAPER provides feedback on every learning step, unlike the delayed version which only provides score at the end of the episode.

Features	R-1	R-2	R-L
$ASRL$	0.40339	0.11397	0.36574
$REAPER$	0.40259	0.11396	0.36539

Table 4: REAPER with alternate feature spaces on DUC2004.

We can observe the results of using REAPER with various feature sets in Table 4. Experiments were run using REAPER reward, $TD(\lambda)$, and the specified feature set.

Algorithm	R-1	R-2	R-L
$TD(\lambda)$	0.39773	0.11063	0.36018
$SARSA$	0.28287	0.04858	0.26172
API	0.29163	0.06570	0.26542

Table 5: REAPER with alternate algorithms on DUC2004.

Table 5 displays the performance of REAPER with alternate algorithms. $TD(\lambda)$ and $SARSA$ are run using the delayed reward feature, while API requires an immediate reward and was thus run with the immediate reward.

System	R-2	R-SU4
$REAPER$	0.07008	0.11689
$ASRL$	0.05639	0.09379
$S24$	0.09505	0.15464
Baseline	0.04947	0.09788

Table 6: QF-REAPER on DUC2006.

For query-focused multi-document summarization we experimented with the DUC2006 system task, which contained 50 document clusters consisting of 25 documents each. Parameters were fixed to $\lambda = 1$, $\gamma = 1$ and $k = 250$ **words**. In Table 6 we can observe the results[3] of our query focused systems against DUC2006's top scorer (S24) for ROUGE-2, and a baseline. The baseline was generated by taking the most recent document in the cluster and outputting the first 250 words.

Human Evaluations: We had three native English-speaking human annotators evaluate a set of four randomly chosen summaries produced by REAPER on the DUC2004 dataset.

[3]ROUGE-1.5.5 run with -n 2 -x -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0 -l 250

Metric	A1	A2	A3	AVG
Grammaticality	3.00	4.00	4.00	**3.67**
Redundancy	4.75	4.25	2.75	**3.92**
Referential Clarity	4.00	4.50	3.50	**4.00**
Focus	4.50	3.50	2.25	**3.42**
Structure	3.50	4.00	3.00	**3.50**
Responsiveness	4.25	3.75	3.00	**3.67**

Table 7: Human evaluation scores on DUC2004.

Table 7 shows the evaluation results according to the DUC2006 human evaluation guidelines. The first five metrics are related entirely to the linguistic quality of the summary in question and the final metric, Responsiveness, rates the summary on its relevance to the source documents. Columns represent the average provided by a given annotator over the four summaries, and the AVG column represents the average score for all three annotators over all four summaries. Score values are an integer between 1 and 5 inclusive.

7 Discussion

First we present a sample of a summary generated from a randomly selected cluster. The following summary was generated from cluster $D30017$ of the DUC 2004 dataset using REAPER reward with $TD(\lambda)$ and REAPER feature space.

A congressman who visited remote parts of North Korea last week said Saturday that the food and health situation there was desperate and deteriorating, and that millions of North Koreans might have starved to death in the last few years. North Korea is entering its fourth winter of chronic food shortages with its people malnourished and at risk of dying from normally curable illnesses, senior Red Cross officials said Tuesday. More than five years of severe food shortages and a near-total breakdown in the public health system have led to devastating malnutrition in North Korea and probably left an entire generation of children physically and mentally impaired, a new study by international aid groups has found. Years of food shortages have stunted the growth of millions of North Korean children, with two-thirds of children under age seven suffering malnourishment, U.N. experts said Wednesday. The founder of South Korea's largest conglomerate plans to visit his native North Korea again next week with a gift of 501 cattle, company officials said Thursday. "There is enough rice.

We can observe that the summary is both syntactically sound, and elegantly summarizes the source documents.

Our baseline results table (Table 1) shows REAPER outperforming ASRL in a statistically significant manner on all three ROUGE metrics in question. However we can see from the absolute differences in score that very few additional important words were extracted (ROUGE-1) however REAPER showed a significant improvement in the structuring and ordering of those words (ROUGE-2, and ROUGE-L).

The balancing factors used in the REAPER reward function are responsible for the behaviour of the reward function, and thus largely responsible for the behaviour of the reinforcement learner. In equation 15 we can see balance numbers of 1, 4, 1, 1, 1 for 1-grams, 2-grams, 3-grams, 4-grams, and LCS respectively. In adjusting these values a user can express a preference for a single metric or a specific mixture of these metrics. Given that the magnitude of scores for n-grams decrease as n increases and given that the magnitude of scores for 1-grams is generally 3 to 4 times larger, in our experience, we can see that this specific reward function favours bigram similarity over unigram similarity. These balance values can be adjusted to suit the specific needs of a given situation, however we leave exploration of this concept for future work.

We can observe in Figure 1 that ASRL does not converge on a stable value, and dips towards the 300^{th} episode while in Figure 2 REAPER does not take nearly such a dramatic dip. These figures display average normalized reward for all 50 document clusters on a single run. Furthermore we can observe that ASRL reaches it's peak reward around episode 225 while REAPER does so around episode 175 suggesting that REAPER converges faster.

7.1 Delayed vs. Immediate Rewards

The delayed vs. immediate rewards results in Table 3 clearly show that delaying the reward provides a significant improvement in globally optimizing the summary for ROUGE score. This can be attributed to the $\lambda = 1$ and $\gamma = 1$ parameter values being suboptimal for the immediate reward situation. This has the added benefit of being much more performant computationally as far fewer reward calculations need be done.

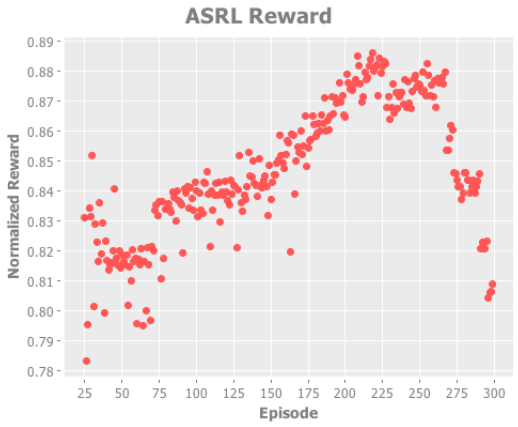


Figure 1: ASRL normalized reward.

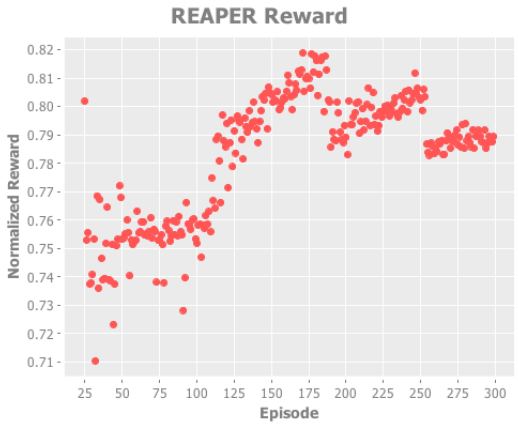


Figure 2: REAPER normalized reward.

7.2 Feature Space

The feature space experiments in Table 4 seem to imply that REAPER performs similarly with both feature sets. We are confident that an improvement could be made through further experimentation. Feature engineering, however, is a very broad field and we plan to pursue this topic in depth in the future.

7.3 Algorithms

$TD(\lambda)$ significantly outperformed both $SARSA$ and API in the algorithm comparison. Ryang and Abekawa (2012) conclude that the feature space is largely responsible for the algorithm performance. This is due to the fact that poor states such as those that are too long, or those that contain few important words will reduce to the same feature set and receive negative rewards collectively. $SARSA$ loses this benefit as a result of its modelling of state-action pairs.

API on the other hand may have suffered a performance loss due to its requirements of an immediate reward, this is because when using a delayed reward if the trajectory of a rollout does not reach a terminal state the algorithm will not be able to make any estimations about the value of the state in question. We propose altering the policy iteration algorithm to use a trajectory length of one episode instead of a fixed number of actions in order to counter the need for an immediate reward function.

7.4 Query Focused Rewards

From the ROUGE results in Table 6 we can infer that while REAPER outperformed ASRL on the query focused task, however it is notable that both systems under performed when compared to the top system from the DUC2006 conference.

We can gather from these results that it is not enough to simply naively calculate similarity with the provided query in order to produce a query-focused result. Given that the results produced by the generic summarization task is rather acceptable according to our human evaluations we suggest that further research be focused on a proper similarity metric between the query and summary to improve the reward function's overall ability to score summaries in a query-focused setting.

8 Conclusion and Future Work

We have explored alternate reward functions, feature sets, and algorithms for the task of automatic summarization using reinforcement learning. We have shown that REAPER outperforms ASRL on both generic summarization and the query focused tasks. This suggests the effectiveness of our reward function and feature space. Our results also confirm that $TD(\lambda)$ performs best for this task compared to $SARSA$ and API.

Due to the acceptable human evaluation scores on the general summarization task it is clear that the algorithm produces acceptable summaries of newswire data. Given that we have a framework for generating general summaries, and the current popularity of the query-focused summarization task, we propose that the bulk of future work in this area be focused on the query-focused task specifically in assessing the relevance of a summary to a provided query. Therefore we intend to pursue future research in utilizing word-sense disambiguation and synonyms, as well as other techniques for furthering REAPER's query similarity

metrics in order to improve its ROUGE and human evaluation scores on query-focused tasks.

Acknowledgments

We would like to thank the anonymous reviewers for their useful comments. The research reported in this paper was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada - discovery grant and the University of Lethbridge. This work was done when the second author was at the University of Lethbridge.

References

S . Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay. 2009. Reinforcement Learning for Mapping Instructions to Actions. *In Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 82–90.

Y. Chali and S. A. Hasan. 2012a. On the Effectiveness of Using Sentence Compression Models for Query-Focused Multi-Document Summarization. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 457–474. Mumbai, India.

Y. Chali and S. A. Hasan. 2012b. Query-focused Multi-document Summarization: Automatic Data Annotations and Supervised Learning Approaches. *Journal of Natural Language Engineering*, 18(1):109–145.

Y. Chali, S. A. Hasan, and S. R. Joty. 2009a. Do Automatic Annotation Techniques Have Any Impact on Supervised Complex Question Answering? *Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP 2009)*, pages 329–332.

Y. Chali, S. R. Joty, and S. A. Hasan. 2009b. Complex Question Answering: Unsupervised Learning Approaches and Experiments. *Journal of Artificial Intelligence Research*, 35:1–47.

Y. Chali, S. A. Hasan, and K. Imam. 2011a. A Reinforcement Learning Framework for Answering Complex Questions. In *Proceedings of the 16th International Conference on Intelligent User Interfaces*, pages 307–310. ACM, Palo Alto, CA, USA.

Y. Chali, S. A. Hasan, and S. R. Joty. 2011b. Improving Graph-based Random Walks for Complex Question Answering Using Syntactic, Shallow Semantic and Extended String Subsequence Kernels. *Information Processing and Management (IPM), Special Issue on Question Answering*, 47(6):843–855.

Y. Chali, S. A. Hasan, and K. Imam. 2012. Improving the Performance of the Reinforcement Learning Model for Answering Complex Questions. In *Proceedings of the 21st ACM Conference on Information and Knowledge Management (CIKM 2012)*, pages 2499–2502. ACM, Maui, Hawaii, USA.

J. Conroy, J. Goldstein, and D. Leary. 2004. Left-Brain / Right-Brain Multi-Document Summarization. *In Proceedings of the Document Un- derstanding Conference (DUC 2004)*.

D. S. Hirschberg. 1977. Algorithms for the Longest Common Subsequence Problem. *In Journal of the ACM*, 24(4):664–675, October.

M. Lagoudakis and R. Parr. 2003. Reinforcement learning as classification: Leveraging modern classifiers. *In Proceedings of the Twentieth International Conference on Machine Learning*, 20(1):424.

J. Li and S. Li. 2013. A Novel Feature-based Bayesian Model for Query Focused Multi-document Summarization. *In Transactions of the Association for Computational Linguistics*, 1:89–98.

C. Lin and M. Rey. 2003. Improving Summarization Performance by Sentence Compression A Pilot Study. *In Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages*.

C. Lin. 2004. ROUGE : A Package for Automatic Evaluation of Summaries. *In Information Sciences*, 16(1):25–26.

Y. Liu, S. Li, Y. Cao, C. Lin, D. Han, and Y. Yu. 2008. Understanding and Summarizing Answers in Community-Based Question Answering Services. *In COLING '08 Proceedings of the 22nd International Conference on Computational Linguistics*, 1(August):497–504.

S. Lyngbaek. 2013. *SPORK: A Summarization Pipeline for Online Repositories of Knowledge*. M.sc. thesis, California Polytechnic State University.

I. Mani. 2001. *Automatic Summarization*. John Benjamins Publishing.

C. D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*, volume 26 of . MIT Press.

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

X. Qian and Y. Liu. 2013. Fast Joint Compression and Summarization via Graph Cuts. *In Conference on Empirical Methods in Natural Language Processing*.

S. Ryang and T. Abekawa. 2012. Framework of Automatic Text Summarization Using Reinforcement Learning. *In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 1:256–265.

H. Saggion, C. Iria, T. M. Juan-Manuel, and E. San-Juan. 2010. Multilingual Summarization Evaluation without Human Models. *In COLING '10 Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, 1:1059–1067.

A. Sood. 2013. *Towards Summarization of Written Text Conversations*. M.sc. thesis, International Institute of Information Technology, Hyderabad, India.

R. S. Sutton and A. G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

C. A. Szepesv. 2009. *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers.

H. Takamura and M. Okumura. 2009. Text Summarization Model based on Maximum Coverage Problem and its Variant. *In EACL '09 Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, (April):781–789.

X Wan. 2007. Towards an Iterative Reinforcement Approach for Simultaneous Document Summarization and Keyword Extraction. *In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 552–559.

L. Wang, H. Raghavan, V. Castelli, R. Florian, and C. Cardie. 2013. A Sentence Compression Based Framework to Query-Focused. *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 1:1384–1394.

W. Yin, Y. Pei, F. Zhang, and L. Huang. 2012. Query-focused multi-document summarization based on query-sensitive feature space. *In Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12*, page 1652.

Improving Multi-documents Summarization by Sentence Compression based on Expanded Constituent Parse Trees

Chen Li[1], **Yang Liu**[1], **Fei Liu**[2], **Lin Zhao**[3], **Fuliang Weng**[3]
[1] Computer Science Department, The University of Texas at Dallas
Richardson, TX 75080, USA
[2] School of Computer Science, Carnegie Mellon University
Pittsburgh, PA 15213, USA
[3] Research and Technology Center, Robert Bosch LLC
Palo Alto, California 94304, USA
{chenli,yangl@hlt.utdallas.edu}
{feiliu@cs.cmu.edu}
{lin.zhao,fuliang.weng@us.bosch.com}

Abstract

In this paper, we focus on the problem of using sentence compression techniques to improve multi-document summarization. We propose an innovative sentence compression method by considering every node in the constituent parse tree and deciding its status – remove or retain. Integer liner programming with discriminative training is used to solve the problem. Under this model, we incorporate various constraints to improve the linguistic quality of the compressed sentences. Then we utilize a pipeline summarization framework where sentences are first compressed by our proposed compression model to obtain top-n candidates and then a sentence selection module is used to generate the final summary. Compared with state-of-the-art algorithms, our model has similar ROUGE-2 scores but better linguistic quality on TAC data.

1 Introduction

Automatic summarization can be broadly divided into two categories: extractive and abstractive summarization. Extractive summarization focuses on selecting salient sentences from the document collection and concatenating them to form a summary; while abstractive summarization is generally considered more difficult, involving sophisticated techniques for meaning representation, content planning, surface realization, etc.

There has been a surge of interest in recent years on generating compressed document summaries as a viable step towards abstractive summarization. These compressive summaries often contain more information than sentence-based extractive summaries since they can remove insignificant sentence constituents and make space for more salient information that is otherwise dropped due to the summary length constraint. Two general strategies have been used for compressive summarization. One is a pipeline approach, where sentence-based extractive summarization is followed or proceeded by sentence compression (Lin, 2003; Zajic et al., 2007; Vanderwende et al., 2007; Wang et al., 2013). Another line of work uses joint compression and summarization. Such methods have been shown to achieve promising performance (Daumé, 2006; Chali and Hasan, 2012; Almeida and Martins, 2013; Qian and Liu, 2013), but they are typically computationally expensive.

In this study, we propose an innovative sentence compression model based on expanded constituent parse trees. Our model uses integer linear programming (ILP) to search the entire space of compression, and is discriminatively trained. It is built based on the discriminative sentence compression model from (McDonald, 2006) and (Clarke and Lapata, 2008), but our method uses an expanded constituent parse tree rather than only the leaf nodes in previous work. Therefore we can extract rich features for every node in the constituent parser tree. This is an advantage of tree-based compression technique (Knight and Marcu, 2000; Galley and McKeown, 2007; Wang et al., 2013). Similar to (Li et al., 2013a), we use a pipeline summarization framework where multiple compression candidates are generated for each pre-selected important sentence, and then an ILP-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 691–701,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

based summarization model is used to select the final compressed sentences. We evaluate our proposed method on the TAC 2008 and 2011 data sets using the standard ROUGE metric (Lin, 2004) and human evaluation of the linguistic quality. Our results show that using our proposed sentence compression model in the summarization system can yield significant performance gain in linguistic quality, without losing much performance on the ROUGE metric.

2 Related Work

Summarization research has seen great development over the last fifty years (Nenkova and McKeown, 2011). Compared to the abstractive counterpart, extractive summarization has received considerable attention due to its clear problem formulation: to extract a set of salient and non-redundant sentences from the given document set. Both unsupervised and supervised approaches have been explored for sentence selection. Supervised approaches include the Bayesian classifier (Kupiec et al., 1995), maximum entropy (Osborne, 2002), skip-chain CRF (Galley, 2006), discriminative reranking (Aker et al., 2010), among others. The extractive summary sentence selection problem can also be formulated in an optimization framework. Previous methods include using integer linear programming (ILP) and submodular functions to solve the optimization problem (Gillick et al., 2009; Li et al., 2013b; Lin and Bilmes, 2010).

Compressive summarization receives increasing attention in recent years, since it offers a viable step towards abstractive summarization. The compressed summaries can be generated through a joint model of the sentence selection and compression processes, or through a pipeline approach that integrates a sentence compression model with a summary sentence pre-selection or post-selection step.

Many studies have explored the joint sentence compression and selection setting. Martins and Smith (2009) jointly performed sentence extraction and compression by solving an ILP problem. Berg-Kirkpatrick et al. (2011) proposed an approach to score the candidate summaries according to a combined linear model of extractive sentence selection and compression. They trained the model using a margin-based objective whose loss captures the final summary quality. Woodsend and Lapata (2012) presented another method where the summary's informativeness, succinctness, and grammaticality are learned separately from data but optimized jointly using an ILP setup. Yoshikawa et al. (2012) incorporated semantic role information in the ILP model.

Our work is closely related with the pipeline approach, where sentence-based extractive summarization is followed or proceeded by sentence compression. There have been many studies on sentence compression, independent of the summarization task. McDonald (2006) firstly introduced a discriminative sentence compression model to directly optimize the quality of the compressed sentences produced. Clarke and Lapata (2008) improved the above discriminative model by using ILP in decoding, making it convenient to add constraints to preserve grammatical structure. Nomoto (2007) treated the compression task as a sequence labeling problem and used CRF for it. Thadani and McKeown (2013) presented an approach for discriminative sentence compression that jointly produces sequential and syntactic representations for output text. Filippova and Altun (2013) presented a method to automatically build a sentence compression corpus with hundreds of thousands of instances on which deletion-based compression algorithms can be trained.

In addition to the work on sentence compression as a stand-alone task, prior studies have also investigated compression for the summarization task. Knight and Marcu (2000) utilized the noisy channel and decision tree method to perform sentence compression in the summarization task. Lin (2003) showed that pure syntactic-based compression may not significantly improve the summarization performance. Zajic et al. (2007) compared two sentence compression approaches for multi-document summarization, including a 'parse-and-trim' and a noisy-channel approach. Galanis and Androutsopoulos (2010) used the maximum entropy model to generate the candidate compressions by removing branches from the source sentences. Woodsend and Lapata (2010) presented a joint content selection and compression model for single-document summarization. They operated over a phrase-based representation of the source document which they obtained by merging information from PCFG parse trees and dependency graphs. Liu and Liu (2013) adopted the CRF-based sentence compression approach for summa-

rizing spoken documents. Unlike the word-based operation, some of these models e.g (Knight and Marcu, 2000; Siddharthan et al., 2004; Turner and Charniak, 2005; Galanis and Androutsopoulos, 2010; Wang et al., 2013), are tree-based approaches that operate on the parse trees and thus the compression decision can be made for a constituent, instead of a single word.

3 Sentence Compression Method

Sentence compression is a task of producing a summary for a single sentence. The compressed sentence should be shorter, contain important content from the original sentence, and be grammatical. In some sense, sentence compression can be described as a 'scaled down version of the text summarization problem' (Knight and Marcu, 2002). Here similar to much previous work on sentence compression, we just focus on how to remove/select words in the original sentence without using operation like rewriting sentence.

3.1 Discriminative Compression Model by ILP

McDonald (2006) presented a discriminative compression model, and Clarke and Lapata (2008) improved it by using ILP for decoding. Since our proposed method is based upon this model, in the following we briefly describe it first. Details can be found in (Clarke and Lapata, 2008). In this model, the following score function is used to evaluate each compression candidate:

$$s(\mathbf{x}, \mathbf{y}) = \sum_{j=2}^{|\mathbf{y}|} s(\mathbf{x}, L(y_{j-1}), L(y_j)) \tag{1}$$

where $\mathbf{x} = x_1 x_2, ..., x_n$ represents an original sentence and $\mathbf{y} = y_1 y_2, ..., y_m$ denotes a compressed sentence. Because the sentence compression problem is defined as a word deletion task, y_j must occur in $\mathbf{x}$. Function $L(y_i) \in [1...n]$ maps word y_i in the compression to the word index in the original sentence $\mathbf{x}$. Note that $L(y_i) < L(y_{i+1})$ is required, that is, each word in x can only occur at most once in compression y. In this model, a first order Markov assumption is used for the score function. Decoding this model is to find the combination of bigrams that maximizes the score function in Eq (1). Clarke and Lapata (2008) introduced the following variables and used ILP to solve it:

$$\delta_i = \begin{cases} 1 & if\ x_i\ is\ in\ the\ compression \\ 0 & otherwise \end{cases} \quad \forall i \in [1..n]$$

$$\alpha_i = \begin{cases} 1 & if\ x_i\ starts\ the\ compression \\ 0 & otherwise \end{cases} \quad \forall i \in [1..n]$$

$$\beta_i = \begin{cases} 1 & if\ x_i\ ends\ the\ compression \\ 0 & otherwise \end{cases} \quad \forall i \in [1..n]$$

$$\gamma_{ij} = \begin{cases} 1 & if\ x_i, x_j\ are\ in\ the\ compression \\ 0 & otherwise \end{cases} \quad \forall i \in [1..n-1] \forall j \in [i+1..n]$$

Using these variables, the objective function can be defined as:

$$\begin{aligned} \max z \quad &= \sum_{i=1}^{n} \alpha_i \cdot s(\mathbf{x}, 0, i) \\ &+ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \gamma_{ij} \cdot s(\mathbf{x}, i, j) \\ &+ \sum_{i=1}^{n} \beta_i \cdot s(\mathbf{x}, i, n+1) \end{aligned} \tag{2}$$

The following four basic constraints are used to make the compressed result reasonable:

$$\sum_{i=1}^{n} \alpha_i = 1 \tag{3}$$

$$\delta_j - \alpha_j - \sum_{i=1}^{j} \gamma_{ij} = 0 \quad \forall j \in [1..n] \tag{4}$$

$$\delta_i - \sum_{j=i+1}^{n} \gamma_{ij} - \beta_i = 0 \quad \forall i \in [1..n] \tag{5}$$

$$\sum_{i=1}^{n} \beta_i = 1 \tag{6}$$

Formula (3) and (6) denote that exactly one word can begin or end a sentence. Formula (4) means if a word is in the compressed sentence, it must either start the compression or follow another word; formula (5) represents if a word is in the

compressed sentence, it must either end the sentence or be followed by another word.

Furthermore, discriminative models are used for the score function:

$$s(\mathbf{x}, \mathbf{y}) = \sum_{j=2}^{|\mathbf{y}|} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, L(y_{j-1}), L(y_j)) \quad (7)$$

High dimensional features are used and their corresponding weights are trained discriminatively.

Above is the basic supervised ILP formulation for sentence compression. Linguistically and semantically motivated constraints can be added in the ILP model to ensure the correct grammar structure in the compressed sentence. For example, Clarke and Lapata (2008) forced the introducing term of prepositional phrases and subordinate clauses to be included in the compression if any word from within that syntactic constituent is also included, and vice versa.

3.2 Compression Model based on Expanded Constituent Parse Tree

In the above ILP model, variables are defined for each word in the sentence, and the task is to predict each word's status. In this paper, we propose to adopt the above ILP framework, but operate directly on the nodes in the constituent parse tree, rather than just the words (leaf nodes in the tree). This way we can remove or retain a chunk of the sentence rather than isolated words, which we expect can improve the readability and grammar correctness of the compressed sentences.

The top part of Fig1 is a standard constituent parse tree. For some levels of the tree, the nodes at that same level can not represent a sentence. We extend the parse tree by duplicating non-POS constituents so that leaf nodes (words and their corresponding POS tags) are aligned at the bottom level as shown in bottom of as Fig1. In the example tree, the solid lines represent relationship of nodes from the original parse tree, the long dot lines denote the extension of the duplication nodes from the upper level to the lower level, and the nodes at the same level are connected (arrowed lines) to represent that is a sequence. Based on this expanded constituent parse tree, we can consider every level as a 'sentence' and the tokens are POS tags and parse tree labels. We apply the above compression model in Section 3.1 on every level to decide every node's status in the final compressed sentence. In order to make the compressed parsed tree reasonable, we model the relationship of nodes between

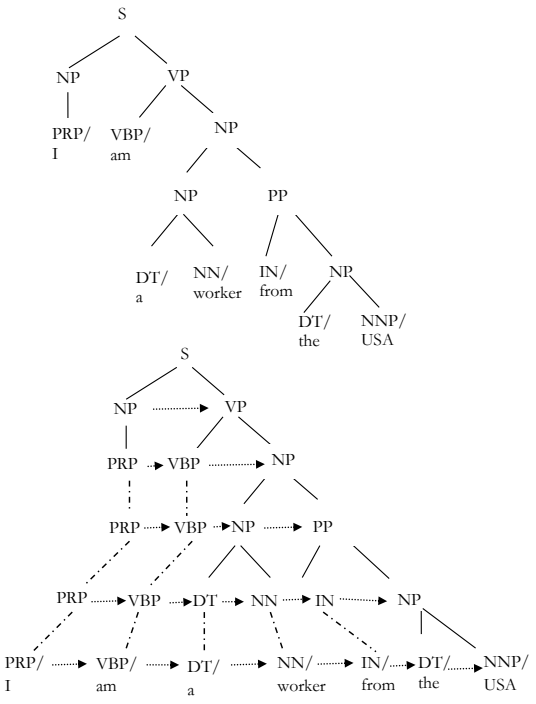


Figure 1: A regular constituent parse tree and its Expanded constituent tree.

adjacent levels as following: if the parent node is labeled as removed, all of its children will be removed; one node will retain if at least one of its children is kept.

Therefore, the objective function in the new ILP formulation is:

$$\begin{aligned} \max z \quad = & \sum_{l=1}^{height} \Big(\sum_{i=1}^{n_l} \alpha_i^l \cdot s(\mathbf{x}, 0, l_i) \\ & + \sum_{i=1}^{n_l - 1} \sum_{j=i+1}^{n_l} \gamma_{ij}^l \cdot s(\mathbf{x}, l_i, l_j) \\ & + \sum_{i=1}^{n_l} \beta_i^l \cdot s(\mathbf{x}, l_i, n_l + 1) \Big) \end{aligned} \quad (8)$$

where height is the depth for a parse tree (starting from level 1 for the tree), and n_l means the length of level l (for example, $n_5 = 6$ in the example in Fig1). Then every level will have a set of parameters $\delta_i^l, \alpha_i^l, \beta_i^l$, and γ_{ij}^l, and the corresponding constraints as shown in Formula (3) to (6). The relationship between nodes from adjacent levels can be expressed as:

$$\delta_i^l \geq \delta_j^{(l+1)} \quad (9)$$

$$\delta_i^l \leq \sum \delta_j^{(l+1)} \quad (10)$$

in which node j at level $(l+1)$ is the child of node

i at level l. In addition, $1 \leq l \leq height - 1$, $1 \leq i \leq n_l$ and $1 \leq j \leq n_{l+1}$.

3.3 Linguistically Motivated Constraints

In our proposed model, we can jointly decide the status of every node in the constituent parse tree at the same time. One advantage is that we can add constraints based on internal nodes or relationship in the parse tree, rather than only using the relationship based on words. In addition to the constraints proposed in (Clarke and Lapata, 2008), we introduce more linguistically motivated constraints to keep the compressed sentence more grammatically correct. The following describes the constraints we used based on the constituent parse tree.

- If a node's label is 'SBAR', its parent's label is 'NP' and its first child's label is 'WHNP' or 'WHPP' or 'IN', then if we can find a noun in the left siblings of 'SBAR', this subordinate clause could be an attributive clause or appositive clause. Therefore the found noun node should be included in the compression if the 'SBAR' is also included, because the node 'SBAR' decorates the noun. For example, the top part of Fig 2 is part of expanded constituent parse tree of sentence 'Those who knew David were all dead.' The nodes in ellipse should share the same status.

- If a node's label is 'SBAR', its parent's label is 'VP' and its first child's label is 'WHNP', then if we can find a verb in the left siblings of 'SBAR', this subordinate clause could be an objective clause. Therefore, the found verb node should be included in the compression if the 'SBAR' node is also included, because the node 'SBAR' is the object of that verb. An example is shown in the bottom part of Fig 2. The nodes in ellipse should share the same status.

- If a node's label is 'SBAR', its parent's label is 'VP' and its first child's label is 'WHADVP', then if the first leaf for this node is a wh-word (e.g., 'where, when, why') or 'how', this clause may be an objective clause (when the word is 'why, how, where') or attributive clause (when the word is 'where') or adverbial clause (when the word is 'when'). Therefore, similar to above, if a verb or noun is found in the left siblings of 'SBAR', the found verb or noun node should be included in the compression if the 'SBAR' node is also included.

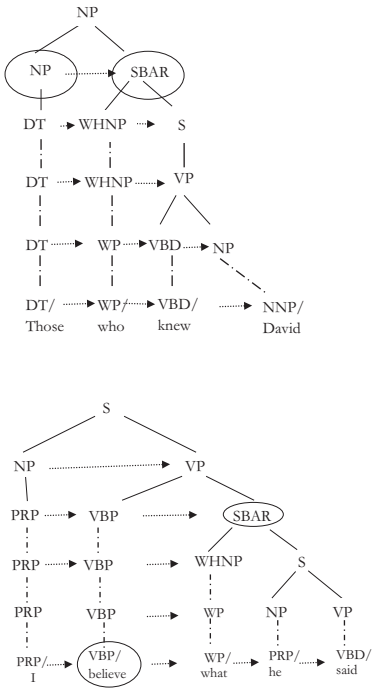


Figure 2: Expanded constituent parse tree for examples.

- If a node's label is 'SBAR' and its parent's label is 'ADJP', then if we can find a 'JJ', 'JJR', or 'JJS' in the left siblings of 'SBAR', the 'SBAR' node should be included in the compression if the found 'JJ', 'JJR' or 'JJS' node is also included because the node 'SBAR' is decorated by the adjective.

- The node with a label of 'PRN' can be removed without other constraints.

We also include some other constraints based on the Stanford dependency parse tree. Table 1 lists the dependency relations we considered.

- For type I relations, the parent and child node with those relationships should have the same value in the compressed result (both are kept or removed).

- For type II relations, if the child node in those relations is retained in the compressed sentence, the parent node should be also retained.

	Dependency Relation	Example
I	prt: phrase verb particle	They shut down the station. prt(shut,down)
	prep: prepositional modifier	He lives in a small village. prep(lives,in)
	pobj: object of a preposition	I sat on the chair. pobj(on,chair)
	nsubj: nominal subject	The boy is cute. nsubj(cute,boy)
	cop: copula	Bill is big. cop(big,is)
II	partmod: participial modifier	Truffles picked during the spring are tasty. partmod(truffles,picked)
	nn: noun compound modifier	Oil price futures. nn(futures,oil)
	acomp: adjectival complement	She looks very beautiful. acomp(looks,beautiful)
III	pcomp: prepositional complement	He felt sad after learning that tragedy. pcomp(after,learning)
	ccomp: clausal complement	I am certain that he did it. ccomp(certain,did)
	tmod: temporal modifier	Last night I swam in the pool. tmod(swam,night)

Table 1: Some dependency relations used for extra constraints. All the examples are from (Marneffe and Manning, 2002)

- For type III relations, if the parent node in these relations is retained, the child node should be kept as well.

3.4 Features

So far we have defined the decoding process and related constraints used in decoding. These all rely on the score function $s(\mathbf{x}, \mathbf{y}) = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, L(y_{j-1}), L(y_j))$ for every level in the constituent parse tree. We included all the features introduced in (Clarke and Lapata, 2008) (those features are designed for leaves). Table 2 lists the additional features we used in our system.

General Features for Every Node
1. individual node label and concatenation of a pair of nodes
2. distance of two nodes at the same level
3. is the node at beginning or end at that level?
4. do the two nodes have the same parent?
5. if two nodes do not have the same parent, then is the left node the rightmost child of its parent? is the right node the leftmost child of its parent?
6. combination of parent label if the node pair are not under the same parent
7. number of node's children: 1/0/>1
8. depth of nodes in the parse tree
Extra Features for Leaf nodes
1. word itself and concatenation of two words
2. POS and concatenation of two words' POS
3. whether the word is a stopword
4. node's named entity tag
5. dependency relationship between two leaves

Table 2: Features used in our system besides those used in (Clarke and Lapata, 2008).

3.5 Learning

To learn the feature weights during training, we perform ILP decoding on every sentence in the training set, to find the best hypothesis for each node in the expanded constituent parse tree. If the hypothesis is incorrect, we update the feature weights using the structured perceptron learning strategy (Collins, 2002). The reference label for every node in the expanded constituent parse tree is obtained automatically from the bottom to the top of the tree. Since every leaf node (word) is human annotated (removed or retain), we annotate the internal nodes as removed if all of its children are removed. Otherwise, the node is annotated as retained.

During perceptron training, a fixed learning rate is used and parameters are averaged to prevent overfitting. In our experiment, we observe stable convergence using the held-out development corpus, with best performance usually obtained around 10-20 epochs.

4 Summarization System

Similar to (Li et al., 2013a), our summarization system is , which consists of three key components: an initial sentence pre-selection module to select some important sentence candidates; the above compression model to generate n-best compressions for each sentence; and then an ILP summarization method to select the best summary sentences from the multiple compressed sentences.

The sentence pre-selection model is a simple supervised support vector regression (SVR) model that predicts a salience score for each sentence and selects the top ranked sentences for further processing (compression and summarization). The target value for each sentence during training is the ROUGE-2 score between the sentence and the human written abstracts. We use three common features: (1) sentence position in the document; (2) sentence length; and (3) interpolated n-gram document frequency as introduced in (Ng et al., 2012).

The final sentence selection process follows the

ILP method introduced in (Gillick et al., 2009). Word bi-grams are used as concepts, and their document frequency is used as weights. Since we use multiple compressions for one sentence, an additional constraint is used: for each sentence, only one of its n-best compressions may be included in the summary.

For the compression module, using the ILP method described above only finds the best compression result for a given sentence. To generate n-best compression candidates, we use an iterative approach – we add one more constraints to prevent it from generating the same answer every time after getting one solution.

5 Experimental Results

5.1 Experimental Setup

Summarization Data For summarization experiments, we use the standard TAC data sets[1], which have been used in the NIST competitions. In particular, we used the TAC 2010 data set as training data for the SVR sentence pre-selection model, TAC 2009 data set as development set for parameter tuning, and the TAC 2008 and 2011 data as the test set for reporting the final summarization results. The training data for the sentence compression module in the summarization system is summary guided compression corpus annotated by (Li et al., 2013a) using TAC2010 data. In the compression module, for each word we also used its document level feature.[2]

Compression Data We also evaluate our compression model using the data set from (Clarke and Lapata, 2008). It includes 82 newswire articles with manually produced compression for each sentence. We use the same partitions as (Martins and Smith, 2009), i.e., 1,188 sentences for training and 441 for testing.

Data Processing We use Stanford CoreNLP toolkit[3] to tokenize the sentences, extract name entity tags, and generate the dependency parse tree. Berkeley Parser (Petrov et al., 2006) is adopted to obtain the constituent parse tree for every sentence and POS tag for every token. We use Pocket CRF[4] to implement the CRF sentence compression model. SVMlight[5] is used for the summary sentence pre-selection model. Gurobi ILP solver[6] does all ILP decoding.

5.2 Summarization Results

We compare our summarization system against four recent studies, which have reported some of the highest published results on this task. Berg-Kirkpatrick et al. (2011) introduced a joint model for sentence extraction and compression. Woodsend and Lapata (2012) learned individual summary aspects from data, e.g., informativeness, succinctness, grammaticalness, stylistic writing conventions, and jointly optimized the outcome in an ILP framework. Ng et al. (2012) exploited category-specific information for multi-document summarization. Almeida and Martins (2013) proposed compressive summarization method by dual decomposition and multi-task learning. Our summarization framework is the same as (Li et al., 2013a), except they used a CRF-based compression model. In addition to the four previous studies, we also report the best achieved results in the TAC competitions.

Table 3 shows the summarization results of our method and others. The top part contains the results for TAC 2008 data and bottom part is for TAC 2011 data. We use the ROUGE evaluation metrics (Lin, 2004), with R-2 measuring the bigram overlap between the system and reference summaries and R-SU4 measuring the skip-bigram with the maximum gap length of 4. In addition, we evaluate the linguistic quality (LQ) of the summaries for our system and (Li et al., 2013a).[7] The linguistic quality consists of two parts. One evaluates the grammar quality within a sentence. For this, annotators marked if a compressed sentence is grammatically correct. Typical grammar errors include lack of verb or subordinate clause. The other evaluates the coherence between sentences, including the order of sentences and irrelevant sentences. We invited 3 English native speakers to do this evaluation. They gave every compressed sentence a grammar score and a coherence score for

[1] http://www.nist.gov/tac/data/index.html

[2] Document level features for a word include information such as the word's document frequency in a topic. These features cannot be extracted from a single sentence, as in the standard sentence compression task, and are related to the document summarization task.

[3] http://nlp.stanford.edu/software/corenlp.shtml

[4] http://sourceforge.net/projects/pocket-crf-1/

[5] http://svmlight.joachims.org/

[6] http://www.gurobi.com

[7] We chose to evaluate the linguistic quality for this system because of two reasons: one is that we have an implementation of that method; the other more important one is that it has the highest reported ROUGE results among the compared methods.

System	R-2	R-SU4	Gram	Cohere
TAC'08 Best System	11.03	13.96	n/a	n/a
(Berg-Kirk et al., 2011)	11.70	14.38	n/a	n/a
(Woodsend et al., 2012)	11.37	14.47	n/a	n/a
(Almeida et al.,2013)	12.30	15.18	n/a	n/a
(Li et al., 2013a)	**12.35**	15.27	3.81	3.41
Our System	12.23	**15.47**	**4.29**	**4.11**
TAC'11 Best System	13.44	16.51	n/a	n/a
(Ng et al., 2012)	13.93	16.83	n/a	n/a
(Li et al., 2013a)	**14.40**	**16.89**	3.67	3.32
Our System	14.04	16.67	**4.18**	**4.07**

Table 3: Summarization results on the TAC 2008 and 2011 data sets.

each topic. The score is scaled and ranges from 1 (bad) to 5 (good). Therefore, in table 3, the grammar score is the average score for each sentence and coherence score is the average for each topic. We measure annotators' agreement in the following way: we consider the scores from each annotator as a distribution and we find that these three distributions are not statistically significantly different each other ($p > 0.05$ based on paired t-test).

We can see from the table that in general, our system achieves better ROUGE results than most previous work except (Li et al., 2013a) on both TAC 2008 and TAC 2011 data. However, our system's linguistic quality is better than (Li et al., 2013a). The CRF-based compression model used in (Li et al., 2013a) can not well model the grammar. Particularly, our results (ROUGE-2) are statistically significantly ($p < 0.05$) higher than TAC08 Best system, but are not statistically significant compared with (Li et al., 2013a) ($p > 0.05$). The pattern is similar in TAC 2011 data. Our result (R-2) is statistically significantly ($p < 0.05$) better than TAC11 Best system, but not statistically ($p > 0.05$) significantly different from (Li et al., 2013a). However, for the grammar and coherence score, our results are statistically significantly ($p < 0.05$) than (Li et al., 2013a). All the above statistics are based on paired t-test.

5.3 Compression Results

The results above show that our summarization system is competitive. In this section we focus on the evaluation of our proposed compression method. We compare our compression system against four other models. HedgeTrimmer in Dorr et al. (2003) applied a variety of linguistically-motivated heuristics to guide the sentences compression; McDonald (2006) used the output of two parsers as features in a discriminative model that decomposes over pairs of consecutive words; Martins and Smith (2009) built the compression model in the dependency parse and utilized the relationship between the head and modifier to preserve the grammar relationship; Wang et al. (2013) developed a novel beam search decoder using the tree-based compression model on the constituent parse tree, which could find the most probable compression efficiently.

System	C Rate (%)	Uni-F1	Rel-F1
HedgeTrimmer	57.64	0.64	0.50
McDonald (2006)	70.95	0.77	0.55
Martins (2009)	71.35	0.77	0.56
Wang (2013)	68.06	0.79	0.59
Our System	71.19	0.77	0.58

Table 4: Sentence compression results. The human compression rate of the test set is 69%.

Table 4 shows the compression results of various systems, along with the compression ratio (C Rate) of the system output. We adopt the compression metrics as used in (Martins and Smith, 2009) that measures the macro F-measure for the retained unigrams (Uni-F1), and the one used in (Clarke and Lapata, 2008) that calculates the F1 score of the grammatical relations labeled by (Briscoe and Carroll, 2002) (Rel-F1). We can see that our proposed compression method performs well, similar to the state-of-the-art systems.

To evaluate the power of using the expanded parse tree in our model, we conducted another experiment where we only consider the bottom level of the constituent parse tree. In some sense, this could be considered as the system in (Clarke and Lapata, 2008). Furthermore, we use two different setups: one uses the lexical features (about the words) and the other does not. Table 5 shows the results using the data in (Clarke and Lapata, 2008). For a comparison, we also include the results using the CRF-based compression model (the one used in (Nomoto, 2007; Li et al., 2013a)). We report results using both the automatically calculated compression metrics and the linguistic quality score. Three English native speaker annotators were asked to judge two aspects of the compressed sentence compared with the gold result: one is the content that looks at whether the important words are kept and the other is the grammar score which evaluates the sentence's readability. Each of these

two scores ranges from 1(bad) to 5(good).

Table 5 shows that when using lexical features, our system has statistically significantly ($p < 0.05$) higher Grammar value and content importance value than the CRF and the leaves only system. When no lexical features are used, default system can achieve statistically significantly ($p < 0.01$) higher results than the CRF and the leaves only system.

We can see that using the expanded parse tree performs better than using the leaves only, especially when lexical features are not used. In addition, we observe that our proposed compression method is more generalizable than the CRF-based model. When our system does not use lexical features in the leaves, it achieves better performance than the CRF-based model. This is important since such a model is more robust and may be used in multiple domains, whereas a model relying on lexical information may suffer more from domain mismatch. From the table we can see our proposed tree based compression method consistently has better linguistic quality. On the other hand, the CRF compression model is the most computationally efficient one among these three compression methods. It is about 200 times faster than our model using the expanded parse tree. Table 6 shows some examples using different methods.

System	C Rate(%)	Uni-F1	Rel-F1	Gram	Imp
Using lexical features					
CRF	79.98	0.80	0.51	3.9	4.0
ILP(I)	80.54	0.79	0.57	4.0	4.2
ILP(II)	79.90	0.80	0.57	4.2	4.4
No lexical features					
CRF	77.75	0.78	0.51	3.35	3.5
ILP(I)	77.77	0.78	0.56	3.7	3.9
ILP(II)	77.78	0.80	0.58	4.1	4.2

Table 5: Sentence compression results: effect of lexical features and expanded parse tree. ILP(I) represents the system using only bottom nodes in constituent parse tree. ILP(II) is our system. Imp means the content importance value.

6 Conclusion

In this paper, we propose a discriminative ILP sentence compression model based on the expanded constituent parse tree, which aims to improve the linguistic quality of the compressed sentences in the summarization task. Linguistically motivated constraints are incorporated to improve the sentence quality. We conduct experiments on the TAC 2008 and 2011 summarization data sets and show that by incorporating this sentence compression model, our summarization system can yield significant performance gain in linguistic quality without losing much ROUGE results. The analysis of the compression module also demonstrates its competitiveness, in particular the better linguistic quality and less reliance on lexical cues.

Using lexical features
Source:
Apart from drugs, detectives believe money is laundered from a variety of black market deals involving arms and high technology.
Human compress:
detectives believe money is laundered from a variety of black market deals.
CRF result :
Apart from drugs detectives believe money is laundered from a black market deals involving arms and technology.
ILP(I) Result:
detectives believe money is laundered from a variety of black deals involving arms.
ILP(II) Result:
detectives believe money is laundered from black market deals.
No lexical features
Source:
Mrs Allan's son disappeared in May 1989, after a party during his back packing trip across North America.
Human compress:
Mrs Allan's son disappeared in 1989, after a party during his trip across North America.
CRF result :
Mrs Allan's son disappeared May 1989, after during his packing trip across North America.
ILP(I) Result:
Mrs Allan's son disappeared in May, 1989, after a party during his packing trip across North America .
ILP(II) Result:
Mrs Allan's son disappeared in May 1989, after a party during his trip.

Table 6: Examples of original sentences and their compressed sentences from different systems.

Acknowledgments

We thank the anonymous reviewers for their detailed and insightful comments on earlier drafts of this paper. The work is also partially supported by NSF award IIS-0845484 and DARPA Contract No. FA8750-13-2-0041. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the funding agencies.

References

Ahmet Aker, Trevor Cohn, and Robert Gaizauskas. 2010. Multi-document summarization using a* search and discriminative training. In *Proceedings of EMNLP*.

Miguel B. Almeida and Andre F. T. Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of ACL*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of ACL*.

Ted Briscoe and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of LREC*.

Yllias Chali and Sadid A. Hasan. 2012. On the effectiveness of using sentence compression models for query-focused multi-document summarization. In *Proceedings of COLING*.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression an integer linear programming approach. *Journal of Artificial Intelligence Research*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Hal Daumé. 2006. Practical structured learning techniques for natural language processing. *Ph.D. thesis, University of Southern California*.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of NAACL*.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of EMNLP*.

Dimitrios Galanis and Ion Androutsopoulos. 2010. An extractive supervised two-stage method for sentence compression. In *Proceedings of NAACL*.

Michel Galley and Kathleen McKeown. 2007. Lexicalized markov grammars for sentence compression. In *Processings of NAACL*.

Michel Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proceedings of EMNLP*.

Dan Gillick, Benoit Favre, Dilek Hakkani-Tur, Berndt Bohnet, Yang Liu, and Shasha Xie. 2009. The icsi/utd summarization system at tac 2009. In *Proceedings of TAC*.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proceedings of AAAI*.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of SIGIR*.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013a. Document summarization via guided sentence compression. In *Proceedings of the EMNLP*.

Chen Li, Xian Qian, and Yang Liu. 2013b. Using supervised bigram-based ilp for extractive summarization. In *Proceedings of ACL*.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of NAACL*.

Chin-Yew Lin. 2003. Improving summarization performance by sentence compression - A pilot study. In *Proceeding of the Sixth International Workshop on Information Retrieval with Asian Language*.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Proceedings of ACL*.

Fei Liu and Yang Liu. 2013. Towards abstractive speech summarization: Exploring unsupervised and supervised approaches for spoken utterance compression. *IEEE Transactions on Audio, Speech, and Language Processing*.

Marie-Catherine de Marneffe and Christopher D Manning. 2002. Stanford typed dependencies manual.

Andre F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the ACL Workshop on Integer Linear Programming for Natural Language Processing*.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of EACL*.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*.

Jun-Ping Ng, Praveen Bysani, Ziheng Lin, Min-Yen Kan, and Chew-Lim Tan. 2012. Exploiting category-specific information for multi-document summarization. In *Proceedings of COLING*.

Tadashi Nomoto. 2007. Discriminative sentence compression with conditional random fields. *Information Processing and Management*.

Miles Osborne. 2002. Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL*.

Xian Qian and Yang Liu. 2013. Fast joint compression and summarization via graph cuts. In *Proceedings of EMNLP*.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2004. Syntactic simplification for improving content selection in multi-document summarization. In *Proceedings of Coling*.

Kapil Thadani and Kathleen McKeown. 2013. Sentence compression with joint structural inference. In *Proceedings of CoNLL*.

Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of ACL*.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing & Management*.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of ACL*.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Proceedings of ACL*.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proceedings of EMNLP-CoNLL*.

Katsumasa Yoshikawa, Tsutomu Hirao, Ryu Iida, and Manabu Okumura. 2012. Sentence compression with semantic role constraints. In *Proceedings of ACL*.

David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. In *Information Processing and Management*.

Analyzing Stemming Approaches for Turkish Multi-Document Summarization

Muhammed Yavuz Nuzumlalı **Arzucan Özgür**
Department of Computer Engineering
Boğaziçi University
TR-34342, Bebek, İstanbul, Turkey
{yavuz.nuzumlali,arzucan.ozgur}@boun.edu.tr

Abstract

In this study, we analyzed the effects of applying different levels of stemming approaches such as fixed-length word truncation and morphological analysis for multi-document summarization (MDS) on Turkish, which is an agglutinative and morphologically rich language. We constructed a manually annotated MDS data set, and to our best knowledge, reported the first results on Turkish MDS. Our results show that a simple fixed-length word truncation approach performs slightly better than no stemming, whereas applying complex morphological analysis does not improve Turkish MDS.

1 Introduction

Automatic text summarization has gained more importance with the enormous growth and easy availability of the Internet. It is now possible to reach extensive and continuously growing amount of resources. However, this situation brings its own challenges such as finding the relevant documents, and absorbing a large quantity of relevant information (Gupta and Lehal, 2010). The goal of multi-document summarization (MDS) is to automatically create a summary of a set of documents about the same topic without losing the important information. Several approaches for MDS have been proposed in the last decade. However, most of them have only been applied to a relatively small set of languages, mostly English, and recently also to languages like Chinese, Romanian, Arabic, and Spanish (Giannakopoulos, 2013).

Previous studies have shown that methods proposed for languages like English do not generally work well for morphologically rich languages like Finnish, Turkish, and Czech, and additional methods considering the morphological structures of these languages are needed (Eryiğit et al., 2008). For instance, Turkish is an agglutinative language where root words can take many derivational and inflectional affixes. This feature results in a very high number of different word surface forms, and eventually leads to the data sparseness problem. Hakkani-Tür et al. (2000) analyzed the number of unique terms for Turkish and English and showed that the term count for Turkish is three times more than English for a corpus of 1M words.

There are only a few studies for text summarization on Turkish, all of which are about single-document summarization (Altan, 2004; Çığır et al., 2009; Özsoy et al., 2010; Güran et al., 2010; Güran et al., 2011). Some of these studies applied morphological analysis methods, but none of them analyzed their effects in detail.

To our best knowledge, this paper reports the first multi-document summarization study for Turkish. We used LexRank as the main summarization algorithm (Erkan and Radev, 2004), applied and analyzed different levels of stemming methods such as complex morphological analysis and fixed-length word truncation. We also created the first manually annotated MDS data set for Turkish, which has been made publicly available for future studies.

The rest of the paper is organized as follows. Section 2 presents the related work on MDS, as well as on the applications of morphological analysis on Turkish for different Natural Language Processing (NLP) and Information Retrieval (IR) problems. In Section 3, we provide a very brief introduction to the Turkish morphology and present the stemming methods that we evaluated. The details about the created data set and our experimental setup are presented in Section 4. We present and discuss the results in Section 5, and conclude in Section 6.

2 Related Work

A large number of methods have been proposed for multi-document summarization in the last 10-15 years (e.g. (Erkan and Radev, 2004; Shen and Li, 2010; Christensen et al., 2013)). While most of these approaches have only been applied to English, summarization data sets and systems for other languages like Chinese, Romanian, and Arabic have also been proposed in the recent years (Giannakopoulos, 2013).

Previous studies on automatic summarization for Turkish only tackled the problem of single-document summarization (SDS). Altan (2004) and Çığır et al. (2009) proposed feature-based approaches for Turkish SDS, whereas Özsoy et al. (2010) and Güran et al. (2010) used Latent Semantic Analysis (LSA) based methods. Güran et al. (2011) applied non-negative ma-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 702–706,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Word	Analysis
gören *(the one who sees)*	gör+en(DB)
görülen *(the one which is seen)*	gör+ül(DB)+en(DB)
görüş *(opinion)*	gör+üş(DB)
görüşün *(your opinion)*	gör+üş(DB)+ün
görüşler *(opinions)*	gör+üş(DB)+ler
görüşme *(negotiation)*	gör+üş(DB)+me(DB)
görüşmelerin *(of negotiations)*	gör+üş(DB)+me(DB)+ler+in

Table 1: Different word forms and their morphological analysis for the stem "gör" (to see). The derivational boundaries are marked with (DB).

trix factorization (NMF) and used consecutive words detection as a preprocessing step.

The effect of morphological analysis for Turkish was analyzed in detail for Information Retrieval (Can et al., 2008) and Text Categorization (Akkuş and Çakıcı, 2013). Can et al. (2008) showed that using fixed-length truncation methods perform similarly to lemmatization-based stemming for information retrieval. Akkuş and Çakıcı (2013) obtained better results for text categorization with fixed-length word truncation rather than complex morphological analysis, but the difference was not significant. For other morphologically rich languages, there is a case study on Greek by Galiotou et al. (2013). They applied different stemming algorithms and showed that stemming on Greek texts improves the summarization performance.

3 Methodology

This section contains detailed information about the application of different levels of morphological features during the summarization process. Before diving into the details, we provide a very brief description of the morphological structure of the Turkish language.

3.1 Turkish Morphology

Turkish is an agglutinative language with a productive morphology. Root words can take one or more derivational and inflectional affixes; therefore, a root can be seen in a large number of different word forms. Another issue is the morphological ambiguity, where a word can have more than one morphological parse.

Table 1 shows an example list of different word forms for the stem "gör" (to see). All the words in the table have the same root, but the different suffixes lead to different surface forms which may have similar or different meanings. When the surface forms of these words are used in a summarization system, they will be regarded as totally different words. However, if a morphological analysis method is applied to the sentences before giving them to the summarization system, words with similar meanings can match during the sentence similarity calculations.

3.2 Stemming Policies

In this section, we explain the different stemming methods that we investigated.

Raw: In this method, we take the surface forms of words, without applying any stemming.

Root: This method takes the most simple unit of a word, namely the root form. For example, in Table 1, the words "gören", "görüşün", and "görüşmelerin" have the same root (gör), so they will match during sentence similarity calculations.

Deriv: Using the Root method may oversimplify words because some words that are derived from the same root may have irrelevant meanings. In the above example, "görüşler" and "gören" have different meanings, but they have the same root (gör). In order to solve this oversimplification issue, we propose to preserve derivational affixes, and only remove the inflectional affixes from the words. In this method, "görüşler" and "gören" will not match because when we remove only the inflectional affixes, they become "görüş" and "gören". On the other hand, the words "görüşler" and "görüşün" will match because their Deriv forms are the same, which is "görüş".

Prefix: In Turkish, affixes almost always occur as suffixes, not prefixes. Additionally, applying morphological analysis methods is a time consuming process, and may become an overhead for online applications. Therefore, a fixed-length simplification method is also tried, since it is both a fast method and can help match similar words by taking the first n characters of words which have lengths larger than n.

As the summarization algorithm, we used LexRank (Erkan and Radev, 2004), which is a salient graph-based method that achieves promising results for MDS. In LexRank, first a sentence connectivity graph is constructed based on the cosine similarities between sentences, and then the PageRank (Page et al., 1999) algorithm is used to find the most important sentences.

4 Experimental Setup

4.1 Data Set

One of the greatest challenges for MDS studies in Turkish is that there does not exist a manually annotated data set. In this study, we have collected and manually

annotated a Turkish MDS data set, which is publicly available for future studies[1].

In order to match the standards for MDS data sets, we tried to follow the specifications of the DUC 2004 data set. Our data set consists of 21 clusters, each consisting of around 10 documents. We selected 21 different topics from different domains (e.g., politics, economics, sports, social, daily, and technology), and selected 10 documents on average for each topic. The documents were obtained from the websites of various news sources. The average number of words per document is 337, and the average number of letters in a word is 6.84.

For manual annotation, we divided the 21 clusters into three groups and sent them to three annotators different from the authors. We required the human summaries not to exceed 120 words for the summary of each cluster.

4.2 Tools

4.2.1 Turkish Morphological Analysis

In order to perform different levels of morphological analysis on documents, we used a two-level morphological analyzer (Oflazer, 1994) and a perceptron-based morphological disambiguator (Sak et al., 2007), which is trained with a corpus of about $750,000$ tokens from news articles. The accuracy of the disambiguator has been reported as 96% (Sak et al., 2007). The Root and Deriv forms of words were generated from the disambiguator output.

4.2.2 MEAD Summarization Toolkit

We used MEAD (Radev et al., 2004), which is an open-source toolkit created for extractive MDS, in our experiments. MEAD handles all the necessary processes to generate a summary document (e.g., sentence ranking, selection, re-ordering, and etc.).

We used the LexRank implementation that comes with MEAD as a feature, together with the Centroid and Position features (each feature is equally weighted). We forced the generated summaries not to exceed 120 words. However, we define the following exception in order to preserve the readability and the grammaticality of the generated summary. For a candidate sentence S having n words, if the absolute difference between the threshold (which is 120) and the summary length including sentence S (say N_w) is less than the absolute difference between the threshold and the summary length excluding sentence S (say N_{wo}), and if N_w is less than 132 (which is $120*1.1$), we allow the summary to exceed the threshold and add sentence S as the last summary sentence.

We used term frequency (tf) based cosine similarity as the similarity measure during the sentence selection step. We also required sentence length to be between 6 and 50 words (which we found empirically) in order to increase the readability of the summaries. The reason behind applying this filtering is that very short sentences generally do not contain much information to become a summary sentence, whereas very long sentences decrease the readability and fill a significant percentage of the summary limit.

4.2.3 ROUGE

For evaluation, we used ROUGE, which is a standard metric for automated evaluation of summaries based on n-gram co-occurrence. We used ROUGE-1 (based on uni-grams), ROUGE-2 (based on bi-grams), and ROUGE-W (based on longest common sub-sequence weighted by length) in our experiments. Among these, ROUGE-1 has been shown to agree with human judges the most (Lin and Hovy, 2003), so we give importance to it while interpreting the results.

5 Evaluation and Results

We ran MEAD with the proposed stemming policies using different levels of cosine similarity threshold values to analyze the effect of the similarity threshold on the summarization performance. After the sentences are ranked using the LexRank method, the similarity threshold is used to decide whether to include a sentence to the summary. A sentence is not included to the summary, if its similarity to a previously picked sentence is larger than the similarity threshold.

In our preliminary experiments, we used the default similarity threshold 0.7, which was found empirically by the MEAD developers for English. However, it produced poor results on the Turkish data set.

Policy	ROUGE-1	ROUGE-2	ROUGE-W
Prefix10	**0.438**	0.194	**0.197**
Prefix12	0.433	**0.197**	0.195
Prefix9	0.432	0.194	0.194
Prefix4	0.432	0.178	0.190
Prefix7	0.431	0.189	0.190
Prefix5	0.431	0.183	0.190
Prefix6	0.430	0.185	0.189
Raw	0.428	0.189	0.191
Deriv	0.428	0.178	0.188
Prefix8	0.427	0.187	0.188
Prefix11	0.427	0.190	0.193
Root	0.420	0.186	0.185

Table 2: Best scores for different policies

Figure 1 shows the F-1 scores for the ROUGE-1 metric for policies with different thresholds. After the threshold exceeds 0.5, the performances for all policies start to decrease, so we don't report the values here to make the chart more readable. In general, Raw and Prefix10 (taking the first 10 letters of the words) achieve better performances with lower threshold values, whereas Root and Deriv operate better with relatively higher threshold values. As we stated earlier, in Turkish, words with similar meanings can occur in

[1]The data set can be retrieved from the following github repository: https://github.com/manuyavuz/TurkishMDSDataSet_alpha

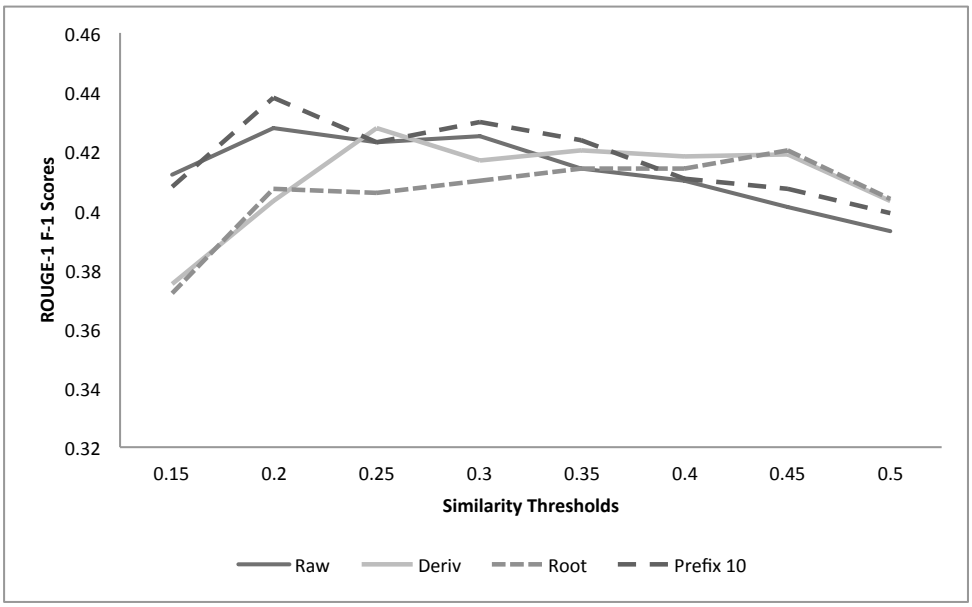


Figure 1: F-1 scores for different similarity threshold values

text with different surface forms due to their inflections. Such words can not be matched during similarity computation if morphological analysis is not performed. Therefore, using higher similarity threshold values cause very similar sentences to occur together in the summaries, and eventually, result in poor scores.

Table 2 shows the best scores obtained by each policy. The Prefix policy generally outperforms the Raw policy. The Prefix10 policy achieves the best ROUGE-1 score. On the other hand, the policies that apply complex morphological analysis (i.e. Root and Deriv) are not able to outperform the simple Prefix and Raw policies. The Deriv policy performs similarly to the Raw and Prefix policies, whereas the Root policy obtains the lowest ROUGE-1 score.

5.1 Discussion

The results show that using a simple fixed-length prefix policy outperforms all other methods, and applying complex morphological analysis does not improve Turkish MDS. The poor performance of the Root policy is somewhat expected due to the fact that, if we preserve only the roots of the words, we lose the semantic differences among the surface forms provided by the derivational affixes. On the other hand, the reason behind the observation that Deriv and Raw obtain similar performances is not obvious.

In order to further analyze this observation, we used an entropy based measure, which is calculated as shown below, to quantify the homogeneity of the clusters in the data set in terms of the variety of the surface forms corresponding to the Deriv forms of each word in the cluster. We first compute the entropy for each Deriv form in a cluster. The entropy of a Deriv form is lower, if it occurs with fewer different surface forms in the cluster. The entropy of a cluster is computed by summing the entropies of the Deriv forms in the cluster and dividing the sum by the number of words in the cluster (i.e. N).

$$D_{Deriv_i} = \{t \mid t\ inflected\ from\ Deriv\ i\}$$

$$H(Deriv_i) = \sum_{t \in D_{Deriv_i}} p(t) \log p(t)$$

$$H(C) = \sum_i \frac{H(Deriv_i)}{N}$$

To compare with the data set clusters, we generated random document clusters by randomly selecting 10 different clusters and then randomly selecting one document from each selected cluster. The average entropy value for the data set clusters and the random clusters were 4.99 and 7.58, respectively. Due to this significant difference, we can hypothesize that the documents about the same topic show a more homogeneous structure. In other words, a Deriv form is usually seen in the same surface form in a cluster of documents which are about the same topic. Therefore, the Deriv policy and the Raw policy achieve similar results for summarizing documents about the same topic.

During evaluation, we ran ROUGE with the Deriv versions of the human summaries and the system summaries in order to match semantically similar words having different surface forms. We also experimented with ROUGE using the Raw versions, but the results followed very similar patterns, so those results were not reported.

6 Conclusion

In this paper, we reported the first steps for a multi-document summarization system for Turkish. A manually annotated data set has been constructed from news

articles, and made publicly available for future studies. We utilized the LexRank summarization algorithm, and analyzed the effects of different stemming policies for Turkish MDS. Our results show that simple fixed-length truncation methods with high limits (such as taking the first 10 letters) improves summarization scores. In contrast to our expectation, using morphological analysis does not enhance Turkish MDS, possibly due to the homogeneousness of the documents in a cluster to be summarized. As future work, we plan to extend the data set with more clusters and more reference summaries, as well as to develop sentence compression methods for Turkish MDS.

Acknowledgments

We would like to thank Ferhat Aydın for his contributions during the data set corpus collection and annotation process. We would also like to thank Burak Sivrikaya and Serkan Bugur for their help in generating the human summaries for the data set.

References

Burak Kerim Akkuş and Ruket Çakıcı. 2013. Categorization of turkish news documents with morphological analysis. In *ACL (Student Research Workshop)*, pages 1–8. The Association for Computer Linguistics.

Zeynep Altan. 2004. A turkish automatic text summarization system. In *Proceedings of the IASTED International Conference Artificial Intelligence and Applications*, pages 74–83.

Fazlı Can, Seyit Koçberber, Erman Balçık, Cihan Kaynak, H Cağdaş Öcalan, and Onur M Vursavaş. 2008. Information retrieval on turkish texts. *Journal of the American Society for Information Science and Technology*, 59(3):407–421.

Janara Christensen, Stephen Soderland Mausam, and Oren Etzioni. 2013. Towards coherent multi-document summarization. In *Proceedings of NAACL-HLT*, pages 1163–1173.

Güneş Erkan and Dragomir R. Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. In *EMNLP*, pages 365–371. ACL.

Gülşen Eryiğit, Joakim Nivre, and Kemal Oflazer. 2008. Dependency parsing of turkish. *Computational Linguistics*, 34(3):357–389.

Eleni Galiotou, Nikitas Karanikolas, and Christodoulos Tsoulloftas. 2013. On the effect of stemming algorithms on extractive summarization: a case study. In Panayiotis H. Ketikidis, Konstantinos G. Margaritis, Ioannis P. Vlahavas, Alexander Chatzigeorgiou, George Eleftherakis, and Ioannis Stamelos, editors, *Panhellenic Conference on Informatics*, pages 300–304. ACM.

George Giannakopoulos. 2013. Multi-document multilingual summarization and evaluation tracks in acl 2013 multiling workshop. *MultiLing 2013*, page 20.

Vishal Gupta and Gurpreet Singh Lehal. 2010. A survey of text summarization extractive techniques. *Journal of Emerging Technologies in Web Intelligence*, 2(3).

Aysun Güran, Eren Bekar, and S Akyokuş. 2010. A comparison of feature and semantic-based summarization algorithms for turkish. In *International Symposium on Innovations in Intelligent Systems and Applications*. Citeseer.

A Güran, NG Bayazıt, and E Bekar. 2011. Automatic summarization of turkish documents using non-negative matrix factorization. In *Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on*, pages 480–484. IEEE.

Dilek Z. Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2000. Statistical morphological disambiguation for agglutinative languages. In *COLING*, pages 285–291. Morgan Kaufmann.

Chin-Yew Lin and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *HLT-NAACL*.

Kemal Oflazer. 1994. Two-level description of turkish morphology. *Literary and linguistic computing*, 9(2):137–148.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Stanford InfoLab.

Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Celebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, et al. 2004. Mead-a platform for multidocument multilingual text summarization. Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004).

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2007. Morphological disambiguation of turkish text with perceptron algorithm. In Alexander F. Gelbukh, editor, *CICLing*, volume 4394 of *Lecture Notes in Computer Science*, pages 107–118. Springer.

Chao Shen and Tao Li. 2010. Multi-document summarization via the minimum dominating set. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 984–992. Association for Computational Linguistics.

Celal Çığır, Mücahid Kutlu, and İlyas Çiçekli. 2009. Generic text summarization for turkish. In *ISCIS*, pages 224–229. IEEE.

Makbule Gülçin Özsoy, İlyas Çiçekli, and Ferda Nur Alpaslan. 2010. Text summarization of turkish texts using latent semantic analysis. In Chu-Ren Huang and Dan Jurafsky, editors, *COLING*, pages 869–876. Tsinghua University Press.

Learning from Rational* Behavior

Thorsten Joachims
Department of Computer Science and
Department of Information Science
Cornell University
tj@cs.cornell.edu

Abstract

The ability to learn from user interactions can give systems access to unprecedented amounts of knowledge. This is evident in search engines, recommender systems, and electronic commerce, and it can be the key to solving other knowledge intensive tasks. However, extracting the knowledge conveyed by user interactions is less straightforward than standard machine learning, since it requires learning systems that explicitly account for human decision making, human motivation, and human abilities.

In this talk, I argue that the design space of such interactive learning systems encompasses not only the machine learning algorithm itself, but also the design of the interaction under an appropriate model of user behavior. To this effect, the talk explores how integrating microeconomic models of human behavior into the learning process leads to new interaction models and their associated learning algorithms, leading to systems that have provable guarantees and that perform robustly in practice.

About the Speaker

Thorsten Joachims is a Professor in the Department of Computer Science and the Department of Information Science at Cornell University. His research interests center on a synthesis of theory and system building in machine learning, with applications in information access, language technology, and recommendation. His past research focused on support vector machines, text classification, structured output prediction, convex optimization, learning to rank, learning with preferences, and learning from implicit feedback. In 2001, he finished his dissertation advised by Prof. Katharina Morik at the University of Dortmund. From there he also received his Diploma in Computer Science in 1997. Between 2000 and 2001 he worked as a PostDoc at the GMD Institute for Autonomous Intelligent Systems. From 1994 to 1996 he was a visiting scholar with Prof. Tom Mitchell at Carnegie Mellon University.

* Restrictions apply. Some modeling is required.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), page 707,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Evaluating Neural Word Representations in Tensor-Based Compositional Settings

Dmitrijs Milajevs[1] **Dimitri Kartsaklis**[2] **Mehrnoosh Sadrzadeh**[1] **Matthew Purver**[1]

[1]Queen Mary University of London
School of Electronic Engineering
and Computer Science
Mile End Road, London, UK
{d.milajevs,m.sadrzadeh,m.purver}@qmul.ac.uk

[2]University of Oxford
Department of Computer Science
Parks Road, Oxford, UK
dimitri.kartsaklis@cs.ox.ac.uk

Abstract

We provide a comparative study between neural word representations and traditional vector spaces based on co-occurrence counts, in a number of compositional tasks. We use three different semantic spaces and implement seven tensor-based compositional models, which we then test (together with simpler additive and multiplicative approaches) in tasks involving verb disambiguation and sentence similarity. To check their scalability, we additionally evaluate the spaces using simple compositional methods on larger-scale tasks with less constrained language: paraphrase detection and dialogue act tagging. In the more constrained tasks, co-occurrence vectors are competitive, although choice of compositional method is important; on the larger-scale tasks, they are outperformed by neural word embeddings, which show robust, stable performance across the tasks.

1 Introduction

Neural word embeddings (Bengio et al., 2006; Collobert and Weston, 2008; Mikolov et al., 2013a) have received much attention in the distributional semantics community, and have shown state-of-the-art performance in many natural language processing tasks. While they have been compared with co-occurrence based models in simple similarity tasks at the word level (Levy et al., 2014; Baroni et al., 2014), we are aware of only one work that attempts a comparison of the two approaches in compositional settings (Blacoe and Lapata, 2012), and this is limited to additive and multiplicative composition, compared against composition via a neural autoencoder.

The purpose of this paper is to provide a more complete picture regarding the potential of neural word embeddings in compositional tasks, and meaningfully compare them with the traditional distributional approach based on co-occurrence counts. We are especially interested in investigating the performance of neural word vectors in compositional models involving general mathematical composition operators, rather than in the more task- or domain-specific deep-learning compositional settings they have generally been used with so far (for example, by Socher et al. (2012), Kalchbrenner and Blunsom (2013) and many others).

In particular, this is the first large-scale study to date that applies neural word representations in tensor-based compositional distributional models of meaning similar to those formalized by Coecke et al. (2010). We test a range of implementations based on this framework, together with additive and multiplicative approaches (Mitchell and Lapata, 2008), in a variety of different tasks. Specifically, we use the verb disambiguation task of Grefenstette and Sadrzadeh (2011a) and the transitive sentence similarity task of Kartsaklis and Sadrzadeh (2014) as small-scale focused experiments on pre-defined sentence structures. Additionally, we evaluate our vector spaces on paraphrase detection (using the Microsoft Research Paraphrase Corpus of Dolan et al. (2005)) and dialogue act tagging using the Switchboard Corpus (see e.g. (Stolcke et al., 2000)).

In all of the above tasks, we compare the neural word embeddings of Mikolov et al. (2013a) with two vector spaces both based on co-occurrence counts and produced by standard distributional techniques, as described in detail below. The general picture we get from the results is that in almost all cases the neural vectors are more effective than the traditional approaches.

We proceed as follows: Section 2 provides a concise introduction to distributional word representations in natural language processing. Section

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 708–719,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

3 takes a closer look to the subject of compositionality in vector space models of meaning and describes the range of compositional operators examined here. In Section 4 we provide details about the vector spaces used in the experiments. Our experimental work is described in detail in Section 5, and the results are discussed in Section 6. Finally, Section 7 provides conclusions.

2 Meaning representation

There are several approaches to the representation of word, phrase and sentence meaning. As natural languages are highly creative and it is very rare to see the same sentence twice, any practical approach dealing with large text segments must be *compositional*, constructing the meaning of phrases and sentences from their constituent parts. The ideal method would therefore express not only the similarity in meaning between those constituent parts, but also between the results of their composition, and do this in ways which fit with linguistic structure and generalisations thereof.

Formal semantics Formal approaches to the semantics of natural language have long built upon the classical idea of compositionality – that the meaning of a sentence is a function of the meanings of its parts (Frege, 1892). In compositional type-logical approaches, predicate-argument structures representing phrases and sentences are built from their constituent parts by β-reduction within the lambda calculus framework (Montague, 1970): for example, given a representation of *John* as $john'$ and *sleeps* as $\lambda x.sleep'(x)$, the meaning of the sentence "John sleeps" can be constructed as $\lambda x.sleep'(x)(john') = sleep'(john')$. Given a suitable pairing between words and semantic representations of them, this method can produce structured sentential representations with broad coverage and good generalisability (see e.g. (Bos, 2008)). The above logical approach is extremely powerful because it can capture complex aspects of meaning such as quantifiers and their interaction (see e.g. (Copestake et al., 2005)), and enables inference using well studied and developed logical methods (see e.g. (Bos and Gabsdil, 2000)).

Distributional hypothesis However, such formal approaches are less able to express *similarity* in meaning. We would like to capture the intuition that while *John* and *Mary* are distinct, they are rather similar to each other (both of them are humans) and dissimilar to words such as *dog*, *pavement* or *idea*. The same applies at the phrase and sentence level: "dogs chase cats" is similar in meaning to "hounds pursue kittens", but less so to "cats chase dogs" (despite the lexical overlap).

Distributional methods provide a way to address this problem. By representing words and phrases as vectors or tensors in a (usually highly dimensional) vector space, one can express similarity in meaning via a suitable distance metric within that space (usually cosine distance); furthermore, composition can be modelled via suitable linear-algebraic operations.

Co-occurrence-based word representations One way to produce such vectorial representations is to directly exploit Harris (1954)'s intuition that semantically similar words tend to appear in similar contexts. We can construct a vector space in which the dimensions correspond to contexts, usually taken to be words as well. The word vector components can then be calculated from the frequency with which a word has co-occurred with the corresponding contexts in a window of words, with a predefined length.

Table 1 shows 5 3-dimensional vectors for the words *Mary*, *John*, *girl*, *boy* and *idea*. The words *philosophy*, *book* and *school* signify vector space dimensions. As the vector for *John* is closer to *Mary* than it is to *idea* in the vector space—a direct consequence of the fact that *John*'s contexts are similar to *Mary*'s and dissimilar to *idea*'s—we can infer that *John* is semantically more similar to *Mary* than to *idea*.

Many variants of this approach exist: performance on word similarity tasks has been shown to be improved by replacing raw counts with weighted values (e.g. mutual information)—see (Turney et al., 2010) and below for discussion, and (Kiela and Clark, 2014) for a detailed comparison.

	philosophy	book	school
Mary	0	10	22
John	4	60	59
girl	0	19	93
boy	0	12	164
idea	10	47	39

Table 1: Word co-occurrence frequencies extracted from the BNC (Leech et al., 1994).

Neural word embeddings Deep learning techniques exploit the distributional hypothesis differently. Instead of relying on observed co-occurrence frequencies, a neural language model is trained to maximise some objective function related to e.g. the probability of observing the surrounding words in some context (Mikolov et al., 2013b):

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \qquad (1)$$

Optimizing the above function, for example, produces vectors which maximise the conditional probability of observing words in a context around the target word w_t, where c is the size of the training window, and $w_1 w_2, \cdots w_T$ a sequence of words forming a training instance. Therefore, the resulting vectors will capture the distributional intuition and can express degrees of lexical similarity.

This method has an obvious advantage compared to co-occurrence method: since now the context is *predicted*, the model in principle can be much more robust in data sparsity problems, which is always an important issue for co-occurrence word spaces. Additionally, neural vectors have also proven successful in other tasks (Mikolov et al., 2013c), since they seem to encode not only attributional similarity (the degree to which similar words are close to each other), but also relational similarity (Turney, 2006). For example, it is possible to extract the singular:plural relation (*apple*:*apples*, *car*:*cars*) using vector subtraction:

$$\overrightarrow{apple} - \overrightarrow{apples} \approx \overrightarrow{car} - \overrightarrow{cars}$$

Perhaps even more importantly, semantic relationships are preserved in a very intuitive way:

$$\overrightarrow{king} - \overrightarrow{man} \approx \overrightarrow{queen} - \overrightarrow{woman}$$

allowing the formation of analogy queries similar to $\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} = ?$, obtaining $\overrightarrow{queen}$ as the result.[1]

Both neural and co-occurrence-based approaches have advantages over classical formal approaches in their ability to capture lexical semantics and degrees of similarity; their success at extending this to the sentence level and to more complex semantic phenomena, though, depends on their applicability within compositional models, which is the subject of the next section.

3 Compositional models

Compositional distributional models represent meaning of a sequence of words by a vector, obtained by combining meaning vectors of the words within the sequence using some vector composition operation. In a general classification of these models, one can distinguish between three broad cases: simplistic models which combine word vectors irrespective of their order or relation to one another, models which exploit linear word order, and models which use grammatical structure.

The first approach combines word vectors by vector addition or point-wise multiplication (Mitchell and Lapata, 2008)—as this is independent of word order, it cannot capture the difference between the two sentences "dogs chase cats" and "cats chase dogs". The second approach has generally been implemented using some form of deep learning, and captures word order, but not by necessarily caring about the grammatical structure of the sentence. Here, one works by recursively building and combining vectors for subsequences of words within the sentence using e.g. autoencoders (Socher et al., 2012) or convolutional filters (Kalchbrenner et al., 2014). We do not consider this approach in this paper. This is because, as mentioned in the introduction, their vectors and composition operators are task-specific. These are trained directly to achieve specific objectives in certain pre-determined tasks. We are interested in vector and composition operators that work for *any* compositional task, and which can be combined with results in linguistics and formal semantics to provide generalisable models that can canonically extend to complex semantic phenomena. The third (i.e. the grammatical) approach promises a way to achieve this, and has been instantiated in various ways in the work of Baroni and Zamparelli (2010),Grefenstette and Sadrzadeh (2011a), and Kartsaklis et al. (2012).

General framework Formally, we can specify the vector representation of a word sequence $w_1 w_2 \cdots w_n$ as the vector $\overrightarrow{s} = \overrightarrow{w_1} \star \overrightarrow{w_2} \star \cdots \star \overrightarrow{w_n}$, where $\star$ is a vector operator, such as addition $+$, point-wise multiplication $\odot$, tensor product $\otimes$, or matrix multiplication $\times$.

[1] Levy et al. (2014) improved Mikolov et al. (2013c)'s method of retrieving relational similarities by changing the underlying objective function.

In the simplest compositional models (the first approach described above), $\star$ is $+$ or $\odot$, e.g. see (Mitchell and Lapata, 2008). Grammar-based compositional models (the third approach) are based on a generalisation of the notion of vectors, known as *tensors*. Whereas a vector $\overrightarrow{v}$ is an element of an atomic vector space V, a tensor $\overline{z}$ is an element of a tensor space $V \otimes W \otimes \cdots \otimes Z$. The number of tensored spaces is referred to by the *order* of the space. Using a general duality theorem from multi-linear algebra (Bourbaki, 1989), it follows that tensors are in one-one correspondence with multi-linear maps, that is we have:

$$\overline{z} \in V \otimes W \otimes \cdots \otimes Z \;\cong\; f_{\overline{z}} \colon V \to W \to \cdots \to Z$$

In such a tensor-based formalism, meanings of nouns are vectors and meanings of predicates such as adjectives and verbs are tensors. Meaning of a string of words is obtained by applying the compositions of multi-linear map duals of the tensors to the vectors. For the sake of demonstration, take the case of an intransitive sentence "Sbj Verb"; the meaning of the subject is a vector $\overrightarrow{\text{Sbj}} \in V$ and the meaning of the intransitive verb is a tensor $\overline{\text{Verb}} \in V \otimes W$. Meaning of the sentence is obtained by applying $f_{\overline{Verb}}$ to $\overrightarrow{\text{Sbj}}$, as follows:

$$\overrightarrow{\text{Sbj Verb}} = f_{\overline{Verb}}(\overrightarrow{\text{Sbj}})$$

By tensor-map duality, the above becomes equivalent to the following, where composition has now become the familiar notion of matrix multiplication, that is $\star$ is $\times$:

$$\overline{\text{Verb}} \times \overrightarrow{\text{Sbj}}$$

In general and for words with tensors of order higher than two, $\star$ becomes a generalisation of $\times$, referred to by *tensor contraction*, see e.g. Kartsaklis and Sadrzadeh (2013). Since the creation and manipulation of tensors of order higher than 2 is difficult, one can work with simplified versions of tensors, faithful to their underlying mathematical basis; these have found intuitive interpretations, e.g. see Grefenstette and Sadrzadeh (2011a), Kartsaklis and Sadrzadeh (2014). In such cases, $\star$ becomes a combination of a range of operations such as $\times$, $\otimes$, $\odot$, and $+$.

Specific models In the current paper we will experiment with a variety of models. In Table 2, we present these models in terms of their composition operators and a reference to the main paper in which each model was introduced. For the simple compositional models the sentence is a string of any number of words; for the grammar-based models, we consider simple transitive sentences "Sbj Verb Obj" and introduce the following abbreviations for the concrete method used to build a tensor for the verb:

1. $\overline{\text{Verb}}$ is a verb matrix computed using the formula $\sum_i \overrightarrow{\text{Sbj}_i} \otimes \overrightarrow{\text{Obj}_i}$, where $\overrightarrow{\text{Sbj}_i}$ and $\overrightarrow{\text{Obj}_i}$ are the subjects and objects of the verb across the corpus. These models are referred to by *relational* (Grefenstette and Sadrzadeh, 2011a); they are generalisations of predicate semantics of transitive verbs, from pairs of individuals to pairs of vectors. The models reduce the order 3 tensor of a transitive verb to an order 2 tensor (i.e. a matrix).

2. $\widetilde{\text{Verb}}$ is a verb matrix computed using the formula $\overrightarrow{\text{Verb}} \otimes \overrightarrow{\text{Verb}}$, where $\overrightarrow{\text{Verb}}$ is the distributional vector of the verb. These models are referred to by *Kronecker*, which is the term sometimes used to denote the outer product of tensors (Grefenstette and Sadrzadeh, 2011b). This models also reduces the order 3 tensor of a transitive verb to an order 2 tensor.

3. The models of the last five lines of the table use the so-called *Frobenius* operators from categorical compositional distributional semantics (Kartsaklis et al., 2012) to expand the relational matrices of verbs from order 2 to order 3. The expansion is obtained by either copying the dimension of the subject into the space provided by the third tensor, hence referred to by *Copy-Sbj*, or copying the dimension of the object in that space, hence referred to by *Copy-Obj*; furthermore, we can take addition, multiplication, or outer product of these, which are referred to by *Frobenius-Add*, *Frobenius-Mult*, and *Frobenius-Outer* (Kartsaklis and Sadrzadeh, 2014).

4 Semantic word spaces

Co-occurrence-based vector space instantiations have received a lot of attention from the scientific community (refer to (Kiela and Clark, 2014; Polajnar and Clark, 2014) for recent studies). We instantiate two co-occurrence-based vectors spaces with different underlying corpora and weighting schemes.

Method	Sentence	Linear algebraic formula	Reference
Addition	$w_1 w_2 \cdots w_n$	$\overrightarrow{w_1} + \overrightarrow{w_2} + \cdots + \overrightarrow{w_n}$	Mitchell and Lapata (2008)
Multiplication	$w_1 w_2 \cdots w_n$	$\overrightarrow{w_1} \odot \overrightarrow{w_2} \odot \cdots \odot \overrightarrow{w_n}$	Mitchell and Lapata (2008)
Relational	Sbj Verb Obj	$\overline{\text{Verb}} \odot (\overrightarrow{\text{Sbj}} \otimes \overrightarrow{\text{Obj}})$	Grefenstette and Sadrzadeh (2011a)
Kronecker	Sbj Verb Obj	$\widetilde{\text{Verb}} \odot (\overrightarrow{\text{Sbj}} \otimes \overrightarrow{\text{Obj}})$	Grefenstette and Sadrzadeh (2011b)
Copy object	Sbj Verb Obj	$\overrightarrow{\text{Sbj}} \odot (\overline{\text{Verb}} \times \overrightarrow{\text{Obj}})$	Kartsaklis et al. (2012)
Copy subject	Sbj Verb Obj	$\overrightarrow{\text{Obj}} \odot (\overline{\text{Verb}}^{\mathsf{T}} \times \overrightarrow{\text{Sbj}})$	Kartsaklis et al. (2012)
Frob. add.	Sbj Verb Obj	$(\overrightarrow{\text{Sbj}} \odot (\overline{\text{Verb}} \times \overrightarrow{\text{Obj}})) + (\overrightarrow{\text{Obj}} \odot (\overline{\text{Verb}}^{\mathsf{T}} \times \overrightarrow{\text{Sbj}}))$	Kartsaklis and Sadrzadeh (2014)
Frob. mult.	Sbj Verb Obj	$(\overrightarrow{\text{Sbj}} \odot (\overline{\text{Verb}} \times \overrightarrow{\text{Obj}})) \odot (\overrightarrow{\text{Obj}} \odot (\overline{\text{Verb}}^{\mathsf{T}} \times \overrightarrow{\text{Sbj}}))$	Kartsaklis and Sadrzadeh (2014)
Frob. outer	Sbj Verb Obj	$(\overrightarrow{\text{Sbj}} \odot (\overline{\text{Verb}} \times \overrightarrow{\text{Obj}})) \otimes (\overrightarrow{\text{Obj}} \odot (\overline{\text{Verb}}^{\mathsf{T}} \times \overrightarrow{\text{Sbj}}))$	Kartsaklis and Sadrzadeh (2014)

Table 2: Compositional methods.

GS11 Our first word space is based on a typical configuration that has been used in the past extensively for compositional distributional models (see below for details), so it will serve as a useful baseline for the current work. In this vector space, the co-occurrence counts are extracted from the British National Corpus (BNC) (Leech et al., 1994). As basis words, we use the most frequent nouns, verbs, adjectives and adverbs (POS tags `SUBST`, `VERB`, `ADJ` and `ADV` in the BNC XML distribution[2]). The vector space is lemmatized, that is, it contains only "canonical" forms of words.

In order to weight the raw co-occurrence counts, we use positive point-wise mutual information (PPMI). The component value for a target word t and a context word c is given by:

$$\text{PPMI}(t, c) = \max\left(0, \log \frac{p(c|t)}{p(c)}\right)$$

where $p(c|t)$ is the probability of word c given t in a symmetric window of length 5 and $p(c)$ is the probability of c overall.

Vector spaces based on point-wise mutual information (or variants thereof) have been successfully applied in various distributional and compositional tasks; see e.g. Grefenstette and Sadrzadeh (2011a), Mitchell and Lapata (2008), Levy et al. (2014) for details. PPMI has been shown to achieve state-of-the-art results (Levy et al., 2014) and is suggested by the review of Kiela and Clark (2014). Our use here of the BNC as a corpus and the window length of 5 is based on previous use and better performance of these parameters in a number of compositional experiments (Grefenstette and Sadrzadeh, 2011a; Grefenstette and Sadrzadeh, 2011b; Mitchell and Lapata, 2008; Kartsaklis et al., 2012).

KS14 In this variation, we train a vector space from the ukWaC corpus[3] (Ferraresi et al., 2008), originally using as a basis the 2,000 content words with the highest frequency (but excluding a list of stop words as well as the 50 most frequent content words since they exhibit low information content). The vector space is again lemmatized. As context we consider a 5-word window from either side of the target word, while as our weighting scheme we use local mutual information (i.e. point-wise mutual information multiplied by raw counts). In a further step, the vector space was normalized and projected onto a 300-dimensional space using singular value decomposition (SVD).

In general, dimensionality reduction produces more compact word representations that are robust against potential noise in the corpus (Landauer and Dumais, 1997; Schütze, 1997). SVD has been shown to perform well on a variety of tasks similar to ours (Baroni and Zamparelli, 2010; Kartsaklis and Sadrzadeh, 2014).

Neural word embeddings (NWE) For our neural setting, we used the skip-gram model of Mikolov et al. (2013b) trained with negative sampling. The specific implementation that was tested in our experiments was a 300-dimensional vector space learned from the Google News corpus and provided by the `word2vec`[4] toolkit. Furthermore, the `gensim` library (Řehůřek and Sojka, 2010) was used for accessing the vectors. On the contrary with the previously described co-

[2] `http://www.natcorp.ox.ac.uk/`

[3] `http://wacky.sslmit.unibo.it/`

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

occurrence vector spaces, this version is *not* lemmatized.

The negative sampling method improves the objective function of Equation 1 by introducing negative examples to the training algorithm. Assume that the probability of a specific (c, t) pair of words (where t is a target word and c another word in the same context with t), coming from the training data, is denoted as $p(D = 1|c, t)$. The objective function is then expressed as follows:

$$\prod_{(c,t)\in D} p(D = 1|c, t) \quad (2)$$

That is, the goal is to set the model parameters in a way that maximizes the probability of all observations coming from the training data. Assume now that D' is a set of randomly selected incorrect (c', t') pairs that do not occur in D, then Equation 2 above can be recasted in the following way:

$$\prod_{(c,t)\in D} p(D = 1|c, t) \prod_{(c',t')\in D'} p(D = 0|c', t') \quad (3)$$

In other words, the model tries to distinguish a target word t from random draws that come from a noise distribution. In the implementation we used for our experiments, c is always selected from a 5-word window around t. More details about the negative sampling approach can be found in (Mikolov et al., 2013b); the note of Goldberg and Levy (2014) also provides an intuitive explanation of the underlying setting.

5 Experiments

Our experiments explore the use of the vector spaces above, together with the compositional operators described in Section 3, in a range of tasks all of which require semantic composition: verb sense disambiguation; sentence similarity; paraphrasing; and dialogue act tagging.

5.1 Disambiguation

We use the transitive verb disambiguation dataset described in Grefenstette and Sadrzadeh (2011a)[5]. This dataset consists of ambiguous transitive verbs together with their arguments, landmark verbs that identify one of the verb senses, and human judgements that specify how similar is the disambiguated sense of the verb in the given context to one of the landmarks. This is similar to the intransitive dataset described in (Mitchell and Lapata, 2008). Consider the sentence "system meets specification"; here, *meets* is the ambiguous transitive verb, and *system* and *specification* are its arguments in this context. Possible landmarks for *meet* are *satisfy* and *visit*; for this sentence, the human judgements show that the disambiguated meaning of the verb is more similar to the landmark *satisfy* and less similar to *visit*.

The task is to estimate the similarity of the sense of a verb in a context with a given landmark. To get our similarity measures, we compose the verb with its arguments using one of our compositional models; we do the same for the landmark and then compute the cosine similarity of the two vectors. We evaluate the performance by averaging the human judgements for the same verb, argument and landmark entries, and calculating the Spearman's correlation between the average values and the cosine scores. As a baseline, we compare this with the correlation produced by using only the verb vector, without composing it with its arguments.

Table 3 shows the results of the experiment. NWE *copy-object* composition yields the best correlation with the human judgements, and top performance across all vector spaces and models with a Spearman ρ of 0.456. For the KS14 space, the best result comes from *Frobenius outer* (0.350),

Method	GS11	KS14	NWE
Verb only	0.212	0.325	0.107
Addition	0.103	0.275	0.149
Multiplication	0.348	0.041	0.095
Kronecker	0.304	0.176	0.117
Relational	0.285	0.341	0.362
Copy subject	0.089	0.317	0.131
Copy object	0.334	0.331	**0.456**
Frobenius add.	0.261	0.344	0.359
Frobenius mult.	0.233	0.341	0.239
Frobenius outer	0.284	0.350	0.375

Table 3: Spearman ρ correlations of models with human judgements for the word sense disambiguation task. The best result (NWE Copy object) outperforms the nearest co-occurrence-based competitor (KS14 Frobenius outer) with a statistically significant difference ($p < 0.05$, t-test).

[5]This and the sentence similarity dataset are available at `http://www.cs.ox.ac.uk/activities/compdistmeaning/`

while the best operator for the GS11 space is *point-wise multiplication* (0.348).

For simple point-wise composition, only multiplicative GS11 and additive NWE improve over their corresponding verb-only baselines (but both perform worse than the KS14 baseline). With tensor-based composition in co-occurrence based spaces, *copy subject* yields lower results than the corresponding baselines. Other composition methods, except *Kronecker* for KS14, improve over the verb-only baselines. Finally we should note that, despite the small training corpus, the GS11 vector space performs comparatively well: for instance, *Kronecker* model improves the previously reported score of 0.28 (Grefenstette and Sadrzadeh, 2011b).

5.2 Sentence similarity

In this experiment we use the transitive sentence similarity dataset described in Kartsaklis and Sadrzadeh (2014). The dataset consists of transitive sentence pairs and a human similarity judgement[6]. The task is to estimate a similarity measure between two sentences. As in the disambiguation task, we first compose word vectors to obtain sentence vectors, then compute cosine similarity of them. We average the human judgements for identical sentence pairs to compute a correlation with cosine scores.

Table 4 shows the results. Again, the best performing vector space is KS14, but this time with *addition*: the Spearman ρ correlation score with averaged human judgements is 0.732. Addition was the means for the other vector spaces to achieve top performance as well: GS11 and NWE got 0.682 and 0.689 respectively.

None of the models in tensor-based composition outperformed addition. KS14 performs worse with tensor-based methods here than in the other vector spaces. However, GS11 and NWE, except *copy subject* for both of them and *Frobenius multiplication* for NWE, improved over their verb-only baselines.

5.3 Paraphrasing

In this experiment we evaluate our vector spaces on a mainstream paraphrase detection task.

[6]The textual content of this dataset is the same as that of (Kartsaklis and Sadrzadeh, 2013), the difference is that the dataset of (Kartsaklis and Sadrzadeh, 2014) has updated human judgements whereas the previous dataset used the original annotations of the intransitive dataset of (Mitchell and Lapata, 2010).

Method	**GS11**	**KS14**	**NWE**
Verb only	0.491	0.602	0.561
Addition	0.682	**0.732**	0.689
Multiplication	0.597	0.321	0.341
Kronecker	0.581	0.408	0.561
Relational	0.558	0.437	0.618
Copy subject	0.370	0.448	0.405
Copy object	0.571	0.306	0.655
Frobenius add.	0.566	0.460	0.585
Frobenius mult.	0.525	0.226	0.387
Frobenius outer	0.560	0.439	0.622

Table 4: Results for sentence similarity. There is no statistically significant difference between KS14 addition and NWE addition (the second best result).

Specifically, we get classification results on the Microsoft Research Paraphrase Corpus paraphrase corpus (Dolan et al., 2005) working in the following way: we construct vectors for the sentences of each pair; if the cosine similarity between the two sentence vectors exceeds a certain threshold, the pair is classified as a paraphrase, otherwise as not a paraphrase. For this experiment and that of Section 5.4 below, we investigate only the addition and point-wise multiplication compositional models, since at their current stage of development tensor-based models can only efficiently handle sentences of fixed structure. Nevertheless, the simple point-wise compositional models still allow for a direct comparison of the vector spaces, which is the main goal of this paper.

For each vector space and model, a number of different thresholds were tested on the first 2000 pairs of the training set, which we used as a development set; in each case, the best-performed threshold was selected for a *single* run of our "classifier" on the test set (1726 pairs). Additionally, we evaluate the NWE model with a lemmatized version of the corpus, so that the experimental setup is maximally similar for all vector spaces. The results are shown in the first part of Table 5.

Additive NWE gives the highest performance, with both lemmatized and un-lemmatized versions outperforming the GS11 and KS14 spaces. In the un-lemmatized case, the accuracy of our simple "classifier" (0.73) is close to state-of-the-art range. The state-of-the art result (0.77 accuracy

	Baseline		Co-occurrence				Neural word embeddings			
			GS11		KS14		Unlemmatized		Lemmatized	
Model	Accuracy	F-Score	Accuracy	F-Score	Accuracy	F-Score	Accuracy	F-Score	Accuracy	F-Score
MSR addition	0.65	0.75	0.62	0.79	0.70	0.80	**0.73**	**0.82**	0.72	0.81
MSR multiplication			0.52	0.58	**0.66**	**0.80**	0.42	0.34	0.41	0.36
SWDA addition	0.60	0.58	0.35	0.35	0.40	0.35	**0.63**	**0.60**	0.44	0.40
SWDA multiplication			0.32	0.16	0.39	0.33	**0.58**	**0.53**	0.43	0.38

Table 5: Results for paraphrase detection (MSR) and dialog act tagging (SWDA) tasks. All top results significantly outperform corresponding nearest competitors (for accuracy): $p < 0.05$, χ^2 test.

and 0.84 F-score[7]) by the time of this writing has been obtained using 8 machine translation metrics and three constituent classifiers (Madnani et al., 2012).

The multiplicative model gives lower results than the additive model across all vector spaces. The KS14 vector space shows the steadiest performance, with a drop in accuracy of only 0.04 and no drop in F-score, while for the GS11 and NWE spaces both accuracy and F-score experienced drops by more than 0.20.

5.4 Dialogue act tagging

As our last experiment, we evaluate the word spaces on a dialogue act tagging task (Stolcke et al., 2000) over the Switchboard corpus (Godfrey et al., 1992). Switchboard is a collection of approximately 2500 dialogs over a telephone line by 500 speakers from the U.S. on predefined topics.[8]

The experiment pipeline follows (Milajevs and Purver, 2014). The input utterances are preprocessed so that the parts of interrupted utterances are concatenated (Webb et al., 2005). Disfluency markers and commas are removed from the utterance raw texts. For GS11 and KS14 the utterance tokens are POS-tagged and lemmatized; for NWE, we test the vectors in both a lemmatized and an un-lemmatized version of the corpus.[9] We split the training and testing utterances as suggested by Stolcke et al. (2000). Utterance vectors are then obtained as in the previous experiments; they are reduced to 50 dimensions using SVD and a k-nearest-neighbour classifier is trained on these reduced utterance vectors (the 5 closest neighbours by Euclidean distance are retrieved to make a classification decision). The results are shown in the second part of Table 5.

Un-lemmatized NWE *addition* gave the best accuracy (0.63) and F-score (0.60) (averaged over tag classes), i.e. similar results to (Milajevs and Purver, 2014)—although note that the dimensionality of our NWE vectors is 10 times lower than theirs. *Multiplicative* NWE outperformed the corresponding model in (Milajevs and Purver, 2014). In general, addition consistently outperforms multiplication for all the models. Lemmatization dramatically lowers tagging accuracy: the lemmatized GS11, KS14 and NWE models perform much worse than un-lemmatized NWE, suggesting that morphological features are important for this task.

6 Discussion

Previous comparisons of co-occurrence-based and neural word vector representations vary widely in their conclusions. While Baroni et al. (2014) conclude that "context-predicting models obtain a thorough and resounding victory against their count-based counterparts", this seems to contradict, at least at the first consideration, the more conservative conclusion of Levy et al. (2014) that "analogy recovery is not restricted to neural word embeddings [...] a similar amount of relational similarities can be recovered from traditional distributional word representations" and the findings of Blacoe and Lapata (2012) that "shallow approaches are as good as more computationally intensive alternatives" on phrase similarity and paraphrase detection tasks.

It seems clear that neural word embeddings have an advantage when used in tasks for which they have been trained; our main questions here are whether they outperform co-occurrence based alternatives across the board; and which approach lends itself better to composition using general mathematical operators. To partially an-

[7] F-scores use the standard definition $F = 2(precision * recall)/(precision + recall)$.

[8] The dataset and a Python interface to it are available at `http://compprag.christopherpotts.net/swda.html`

[9] We use `WordNetLemmatizer` of the NLTK library (Bird, 2006).

swer this question, we can compare model behaviour against the baselines in *isolation*.

For the disambiguation and sentence similarity tasks the baseline is the similarity between verbs only, ignoring the context—see above. For the paraphrase task, we take the global vector-based similarity reported in (Mihalcea et al., 2006): 0.65 accuracy and 0.75 F-score. For the dialogue act tagging task the baseline is the accuracy of the bag-of-unigrams model in (Milajevs and Purver, 2014): 0.60.

Sections 5.1 and 5.2 show that although the best choice of vector representation might vary, for small-scale tasks all methods give fairly competitive results. The choice of compositional operator seems to be more important and more task-specific: while a tensor-based operation (Frobenius copy-object) performs best for verb disambiguation, the best result for sentence similarity is achieved by a simple additive model, with all other compositional methods behaving worse than the verb-only baseline in the KS14 case. GS11 and NWE, on the other hand, outperform their baselines with a number of compositional methods, although both of them achieve lower performance than KS14 overall.

Based on only small-scale experiment results, one could conclude that there is little significant difference between the two ways of obtaining vectors. GS11 and NWE show similar behaviour in comparison to their baselines, while it is possible to tune a co-occurrence based vector space (KS14) and obtain the best result. Large scale tasks reveal another pattern: the GS11 vector space, which behaves stably on the small scale, drags behind the KS14 and NWE spaces in the paraphrase detection task. In addition, NWE consistently yields best results. Finally, only the NWE space was able to provide adequate results on the dialogue act tagging task. Table 6 summarizes model performance with regard to baselines.

Task	**GS11**	**KS14**	**NWE**
Disambiguation	+	+	**+**
Sentence similarity	+	**−**	+
Paraphrase	−	+	**+**
Dialog act tagging	−	−	**+**

Table 6: Summary of vector space performance against baselines. General improvement (cases where more than a half of the models perform better) and decrease with regard to a corresponding baseline is respectively marked by + and −. A bold value means that the model gave the best result in the task.

7 Conclusion

In this work we compared the performance of two co-occurrence-based semantic spaces with vectors learned by a neural network in compositional settings. We carried out two small-scale tasks (word sense disambiguation and sentence similarity) and two large-scale tasks (paraphrase detection and dialogue act tagging).

On small-scale tasks, where the sentence structures are predefined and relatively constrained, NWE gives better or similar results to count-based vectors. Tensor-based composition does not always outperform simple compositional operators, but for most of the cases gives results within the same range.

On large-scale tasks, neural vectors are more successful than the co-occurrence based alternatives. However, this study does not reveal whether this is because of their neural nature, or just because they are trained on a larger amount of data.

The question of whether neural vectors outperform co-occurrence vectors therefore requires further detailed comparison to be entirely resolved; our experiments suggest that this is indeed the case in large-scale tasks, but the difference in size and nature of the original corpora may be a confounding factor. In any case, it is clear that the neural vectors of `word2vec` package perform steadily off-the-shelf across a large variety of tasks. The size of the vector space (3 million words) and the available code-base that simplifies the access to the vectors, makes this set a good and safe choice for experiments in the future. Of course, even better performances can be achieved by training neural language models specifically for a given task (see e.g. Kalchbrenner et al. (2014)).

The choice of compositional operator (tensor-based or a simple point-wise operation) depends strongly on the task and dataset: tensor-based composition performed best with the verb disambiguation task, where the verb senses depend strongly on the arguments of the verb. However, it seems to depend less on the nature of the vectors itself: in the disambiguation task, tensor-based

composition proved best for both co-occurrence-based and neural vectors; in the sentence similarity task, where point-wise operators proved best, this was again true across vector spaces.

Acknowledgements

We would like to thank the three anonymous reviewers for their fruitful comments. Support by EPSRC grant EP/F042728/1 is gratefully acknowledged by Milajevs, Kartsaklis and Sadrzadeh. Purver is partly supported by ConCreTe: the project ConCreTe acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET grant number 611733.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

Steven Bird. 2006. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics.

Johan Bos and Malte Gabsdil. 2000. First-order inference and the interpretation of questions and answers. *Proceedings of Gotelog*, pages 43–50.

Johan Bos. 2008. Wide-coverage semantic analysis with boxer. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, Research in Computational Semantics, pages 277–286. College Publications.

N. Bourbaki. 1989. *Commutative Algebra: Chapters 1-7*. Srpinger Verlag, Berlin/New York.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *CoRR*, abs/1003.4394.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(2-3):281–332.

Bill Dolan, Chris Brockett, and Chris Quirk. 2005. Microsoft research paraphrase corpus. *Retrieved May*, 29:2013.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukWaC, a very large web-derived corpus of English. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54.

Gottlob Frege. 1892. On sense and reference. *Ludlow (1997)*, pages 563–584.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 517–520. IEEE.

Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011a. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404. Association for Computational Linguistics.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011b. Experimenting with transitive verbs in a DisCoCat. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 62–66, Edinburgh, UK, July. Association for Computational Linguistics.

Z.S. Harris. 1954. Distributional structure. *Word*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 119–126, Sofia, Bulgaria, August. Association for Computational Linguistics.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, June.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2013. Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNL)*, pages 1590–1601, Seattle, USA, October. Association for Computational Linguistics.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2014. A study of entanglement in a categorical framework of natural language. In *Proceedings of the 11th Workshop on Quantum Physics and Logic (QPL)*, Kyoto, Japan, June.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2012. A unified sentence space for categorical distributional-compositional semantics: Theory and experiments. In *Proceedings of COLING 2012: Posters*, pages 549–558, Mumbai, India, December. The COLING 2012 Organizing Committee.

Douwe Kiela and Stephen Clark. 2014. A systematic study of semantic vector space model parameters. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 21–30, Gothenburg, Sweden, April. Association for Computational Linguistics.

T. Landauer and S. Dumais. 1997. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquision, Induction, and Representation of Knowledge. *Psychological Review*.

Geoffrey Leech, Roger Garside, and Michael Bryant. 1994. Claws4: the tagging of the british national corpus. In *Proceedings of the 15th conference on Computational linguistics-Volume 1*, pages 622–628. Association for Computational Linguistics.

Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, Baltimore, Maryland, USA, June. Association for Computational Linguistics*.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190. Association for Computational Linguistics.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780.

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

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

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

Dmitrijs Milajevs and Matthew Purver. 2014. Investigating the contribution of distributional semantic information for dialogue act classification. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 40–47, Gothenburg, Sweden, April. Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244. Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1439.

Richard Montague. 1970. Universal grammar. *Theoria*, 36(3):373–398.

Tamara Polajnar and Stephen Clark. 2014. Improving distributional semantic vectors through context selection and normalisation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 230–238, Gothenburg, Sweden, April. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Hinrich Schütze. 1997. Ambiguity resolution in natural language learning. csli. *Stanford, CA*, 4:12–36.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Carol Van Ess-Dykema, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373.

Peter D Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Peter D Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Nick Webb, Mark Hepple, and Yorick Wilks. 2005. Dialogue act classification based on intra-utterance features. In *Proceedings of the AAAI Workshop on Spoken Language Understanding*. Citeseer.

Opinion Mining with Deep Recurrent Neural Networks

Ozan İrsoy and **Claire Cardie**
Department of Computer Science
Cornell University
Ithaca, NY, 14853, USA
oirsoy, cardie@cs.cornell.edu

Abstract

Recurrent neural networks (RNNs) are connectionist models of sequential data that are naturally applicable to the analysis of natural language. Recently, "depth in space" — as an orthogonal notion to "depth in time" — in RNNs has been investigated by stacking multiple layers of RNNs and shown empirically to bring a temporal hierarchy to the architecture. In this work we apply these *deep RNNs* to the task of opinion expression extraction formulated as a token-level sequence-labeling task. Experimental results show that deep, narrow RNNs outperform traditional shallow, wide RNNs with the same number of parameters. Furthermore, our approach outperforms previous CRF-based baselines, including the state-of-the-art semi-Markov CRF model, and does so without access to the powerful opinion lexicons and syntactic features relied upon by the semi-CRF, as well as without the standard layer-by-layer pre-training typically required of RNN architectures.

1 Introduction

Fine-grained opinion analysis aims to detect the subjective expressions in a text (e.g. "hate") and to characterize their intensity (e.g. strong) and sentiment (e.g. negative) as well as to identify the opinion holder (the entity expressing the opinion) and the target, or topic, of the opinion (i.e. what the opinion is about) (Wiebe et al., 2005). Fine-grained opinion analysis is important for a variety of NLP tasks including opinion-oriented question answering and opinion summarization. As a result, it has been studied extensively in recent years.

In this work, we focus on the detection of opinion expressions — both *direct subjective expressions* (DSEs) and *expressive subjective expressions* (ESEs) as defined in Wiebe et al. (2005). DSEs consist of explicit mentions of private states or speech events expressing private states; and ESEs consist of expressions that indicate sentiment, emotion, etc., without explicitly conveying them. An example sentence shown in Table 1 in which the DSE "has refused to make any statements" explicitly expresses an opinion holder's attitude and the ESE "as usual" indirectly expresses the attitude of the writer.

The	committee	,	as	usual	,	has
O	O	O	B_ESE	I_ESE	O	B_DSE
refused	to	make	any	statements	.	
I_DSE	I_DSE	I_DSE	I_DSE	I_DSE	O	

Table 1: An example sentence with labels

Opinion extraction has often been tackled as a sequence labeling problem in previous work (e.g. Choi et al. (2005)). This approach views a sentence as a sequence of tokens labeled using the conventional BIO tagging scheme: B indicates the beginning of an opinion-related expression, I is used for tokens inside the opinion-related expression, and O indicates tokens outside any opinion-related class. The example sentence in Table 1 shows the appropriate tags in the BIO scheme. For instance, the ESE "as usual" results in the tags B_ESE for "as" and I_ESE for "usual".

Variants of **conditional random field (CRF)** approaches have been successfully applied to opinion expression extraction using this token-based view (Choi et al., 2005; Breck et al., 2007): the state-of-the-art approach is the semiCRF, which relaxes the Markovian assumption inherent to CRFs and operates at the phrase level rather than the token level, allowing the incorporation of phrase-level features (Yang and Cardie, 2012). The success of the CRF- and semiCRF-based approaches, however, hinges critically on access to an appropriate feature set, typically based on constituent and dependency parse trees, manually crafted opinion lexicons, named entity taggers and other preprocessing components (see Yang and Cardie (2012) for an up-to-date list).

Distributed representation learners provide a different approach to learning in which latent features are modeled as distributed dense vectors of hidden layers. A **recurrent neural network (RNN)** is one such learner that can operate on sequential data of variable length, which means it can also be applied as a sequence labeler. Moreover, **bidirectional RNNs** incorporate information from preceding as well as following tokens (Schuster and Paliwal, 1997) while recent advances in word embedding induction (Collobert and Weston, 2008; Mnih and Hinton, 2007; Mikolov et

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 720–728,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

al., 2013; Turian et al., 2010) have enabled more effective training of RNNs by allowing a lower dimensional dense input representation and hence, more compact networks (Mikolov et al., 2010; Mesnil et al., 2013). Finally, **deep recurrent networks**, a type of RNN with multiple stacked hidden layers, are shown to naturally employ a temporal hierarchy with multiple layers operating at different time scales (Hermans and Schrauwen, 2013): lower levels capture short term interactions among words; higher layers reflect interpretations aggregated over longer spans of text. When applied to natural language sentences, such hierarchies might better model the multi-scale language effects that are emblematic of natural languages, as suggested by previous results (Hermans and Schrauwen, 2013).

Motivated by the recent success of deep architectures in general and deep recurrent networks in particular, we explore an application of **deep bidirectional RNNs** — henceforth **deep RNNs** — to the task of opinion expression extraction. For both DSE and ESE detection, we show that such models outperform conventional, shallow (uni- and bidirectional) RNNs as well as previous CRF-based state-of-the-art baselines, including the semiCRF model.

In the rest of the paper we discuss related work (Section 2) and describe the architecture and training methods for recurrent neural networks (RNNs), bidirectional RNNs, and deep (bidirectional) RNNs (Section 3). We present experiments using a standard corpus for fine-grained opinion extraction in Section 4.

2 Related Work

Opinion extraction. Early work on fine-grained opinion extraction focused on recognizing subjective phrases (Wilson et al., 2005; Munson et al., 2005). Breck et al. (2007), for example, formulated the problem as a token-level sequence-labeling problem and apply a CRF-based approach, which significantly outperformed previous baselines. Choi et al. (2005) extended the sequential prediction approach to jointly identify opinion holders; Choi and Cardie (2010) jointly detected polarity and intensity along with the opinion expression. Reranking approaches have also been explored to improve the performance of a single sequence labeler (Johansson and Moschitti, 2010; Johansson and Moschitti, 2011). More recent work relaxes the Markovian assumption of CRFs to capture phrase-level interactions, significantly improving upon the token-level labeling approach (Yang and Cardie, 2012). In particular, Yang and Cardie (2013) propose a joint inference model to jointly detect opinion expressions, opinion holders and targets, as well as the relations among them, outperforming previous pipelined approaches.

Deep learning. Recurrent neural networks (Elman, 1990) constitute one important class of naturally deep architecture that has been applied to many sequential prediction tasks. In the context of NLP, recurrent neural networks view a sentence as a sequence of tokens and have been successfully applied to tasks such as language modeling (Mikolov et al., 2011) and spoken language understanding (Mesnil et al., 2013). Since classical recurrent neural networks only incorporate information from the past (i.e. preceding tokens), **bidirectional** variants have been proposed to incorporate information from both the past and the future (i.e. subsequent tokens) (Schuster and Paliwal, 1997). Bidirectionality is especially useful for NLP tasks, since information provided by the following tokens is generally helpful (and sometimes essential) when making a decision on the current token.

Stacked recurrent neural networks have been proposed as a way of constructing *deep* RNNs (Schmidhuber, 1992; El Hihi and Bengio, 1995). Careful empirical investigation of this architecture showed that multiple layers in the stack can operate at different time scales (Hermans and Schrauwen, 2013). Pascanu et al. (2013) explore other ways of constructing deep RNNs that are orthogonal to the concept of stacking layers on top of each other. In this work, we focus on the *stacking* notion of depth.

3 Methodology

This section describes the architecture and training methods for the deep bidirectional recurrent networks that we propose for the task of opinion expression mining. Recurrent neural networks are presented in 3.1, bidirectionality is introduced in 3.2, and deep bidirectional RNNs, in 3.3.

3.1 Recurrent Neural Networks

A recurrent neural network (Elman, 1990) is a class of neural network that has recurrent connections, which allow a form of memory. This makes them applicable for sequential prediction tasks with arbitrary spatio-temporal dimensions. Thus, their structure fits many NLP tasks, when the interpretation of a single sentence is viewed as analyzing a sequence of tokens. In this work, we focus our attention on only Elman-type networks (Elman, 1990).

In an **Elman-type network**, the hidden layer h_t at time step t is computed from a nonlinear transformation of the current input layer x_t and the previous hidden layer h_{t-1}. Then, the final output y_t is computed using the hidden layer h_t. One can interpret h_t as an intermediate representation summarizing the past, which is used to make a final decision on the current input.

More formally, given a sequence of vectors $\{x_t\}_{t=1..T}$, an Elman-type RNN operates by computing the following memory and output sequences:

$$h_t = f(Wx_t + Vh_{t-1} + b) \quad (1)$$

$$y_t = g(Uh_t + c) \quad (2)$$

where f is a nonlinear function, such as the sigmoid function and g is the output nonlinearity, such as the

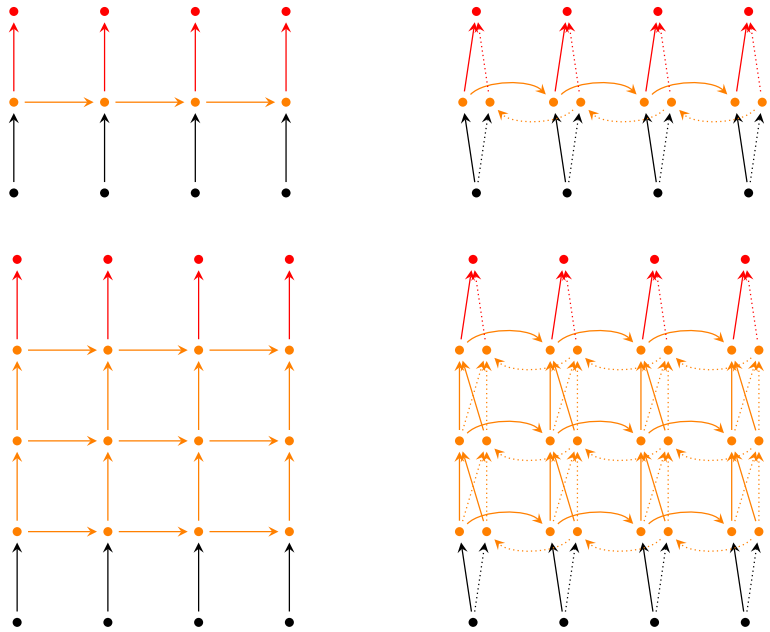


Figure 1: Recurrent neural networks. Each black, orange and red node denotes an input, hidden or output layer, respectively. Solid and dotted lines denote the connections of forward and backward layers, respectively. Top: Shallow unidirectional (left) and bidirectional (right) RNN. Bottom: 3-layer deep unidirectional (left) and bidirectional (right) RNN.

softmax function. W and V are weight matrices between the input and hidden layer, and among the hidden units themselves (connecting the previous intermediate representation to the current one), respectively, while U is the output weight matrix. b and c are bias vectors connected to hidden and output units, respectively. As a base case for the recursion in Equation 1, h_0 is assumed to be 0.

Training an RNN can be done by optimizing a discriminative objective (e.g. the cross entropy for classification tasks) with a gradient-based method. Backpropagation through time can be used to efficiently compute the gradients (Werbos, 1990). This method is essentially equivalent to unfolding the network in time and using backpropagation as in feedforward neural networks, while sharing the connection weights across different time steps. The Elman-style RNN is shown in Figure 1, top left.

3.2 Bidirectionality

Observe that with the above definition of RNNs, we have information only about the past, when making a decision on x_t. This is limiting for most NLP tasks. As a simple example, consider the two sentences: "I did not accept his suggestion" and "I did not go to the rodeo". The first has a DSE phrase ("did not accept") and the second does not. However, any such RNN will assign the same labels for the words "did" and "not" in both sentences, since the preceding sequences (past) are the same: the Elman-style unidirectional RNNs lack the representational power to model this task. A simple way to work around this problem is to include a fixed-size future context around a single input vector (token). However, this approach requires tuning the context size, and ignores future information from outside of the context window. Another way to incorporate information about the future is to add bidirectionality to the architecture, referred as the **bidirectional RNN** (Schuster and Paliwal, 1997):

$$\overrightarrow{h}_t = f(\overrightarrow{W}x_t + \overrightarrow{V}\overrightarrow{h}_{t-1} + \overrightarrow{b}) \quad (3)$$

$$\overleftarrow{h}_t = f(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b}) \quad (4)$$

$$y_t = g(U_{\rightarrow}\overrightarrow{h}_t + U_{\leftarrow}\overleftarrow{h}_t + c) \quad (5)$$

where $\overrightarrow{W}$, $\overrightarrow{V}$ and $\overrightarrow{b}$ are the forward weight matrices and bias vector as before; $\overleftarrow{W}$, $\overleftarrow{V}$ and $\overleftarrow{b}$ are their backward counterparts; $U_{\rightarrow}$, $U_{\leftarrow}$ are the output matrices; and c is the output bias.[1] Again, we assume $\overrightarrow{h}_0 = \overleftarrow{h}_{T+1} = 0$. In this setting $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ can be interpreted as a summary of the past, and the future, respectively, around the time step t. When we make a decision on an input vector, we employ the two intermediate representations $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ of the past and

[1] As a convention, we adopt the following notation throughout the paper: Superscript arrows for vectors disambiguate between forward and backward representations. Superscript arrows for matrices denote the resulting vector representations (connection outputs), and subscript arrows for matrices denote incoming vector representations (connection inputs). We omit subscripts when there is no ambiguity.

the future. (See Figure 1, top right.) Therefore in the bidirectional case, we have perfect information about the sequence (ignoring the practical difficulties about capturing long term dependencies, caused by vanishing gradients), whereas the classical Elman-type network uses only partial information as described above.

Note that the forward and backward parts of the network are independent of each other until the output layer when they are combined. This means that during training, after backpropagating the error terms from the output layer to the forward and backward hidden layers, the two parts can be thought of as separate, and each trained with the classical backpropagation through time (Werbos, 1990).

3.3 Depth in Space

Recurrent neural networks are often characterized as having *depth in time*: when unfolded, they are equivalent to feedforward neural networks with as many hidden layers as the number tokens in the input sequence (with shared connections across multiple layers of time). However, this notion of depth likely does not involve hierarchical processing of the data: across different time steps, we repeatedly apply the same transformation to compute the memory contribution of the input (W), to compute the response value from the current memory (U) and to compute the next memory vector from the previous one (V). Therefore, assuming the input vectors $\{x_t\}$ together lie in the same representation space, as do the output vectors $\{y_t\}$, hidden representations $\{h_t\}$ lie in the same space as well. As a result, they do not necessarily become more and more abstract, hierarchical representations of one another as we traverse in time. However in the more conventional, *stacked* deep learners (e.g. deep feedforward nets), an important benefit of depth is the hierarchy among hidden representations: every hidden layer conceptually lies in a different representation space, and constitutes a more abstract and higher-level representation of the input (Bengio, 2009).

In order to address these concerns, we investigate deep RNNs, which are constructed by stacking Elman-type RNNs on top of each other (Hermans and Schrauwen, 2013). Intuitively, every layer of the deep RNN treats the memory sequence of the previous layer as the input sequence, and computes its own memory representation.

More formally, we have:

$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}_{\rightarrow}^{(i)} \overrightarrow{h}_t^{(i-1)} + \overrightarrow{W}_{\leftarrow}^{(i)} \overleftarrow{h}_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)}) \quad (6)$$

$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}_{\rightarrow}^{(i)} \overrightarrow{h}_t^{(i-1)} + \overleftarrow{W}_{\leftarrow}^{(i)} \overleftarrow{h}_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)}) \quad (7)$$

when $i > 1$ and

$$\overrightarrow{h}_t^{(1)} = f(\overrightarrow{W}^{(1)} x_t + \overrightarrow{V}^{(1)} \overrightarrow{h}_{t-1}^{(1)} + \overrightarrow{b}^{(1)}) \quad (8)$$

$$\overleftarrow{h}_t^{(1)} = f(\overleftarrow{W}^{(1)} x_t + \overleftarrow{V}^{(1)} \overleftarrow{h}_{t+1}^{(1)} + \overleftarrow{b}^{(1)}) \quad (9)$$

Importantly, note that both forward and backward representations are employed when computing the forward and backward memory of the next layer.

Two alternatives for the output layer computations are to employ all memory layers or only the last. In this work we adopt the second approach:

$$y_t = g(U_{\rightarrow} \overrightarrow{h}_t^{(L)} + U_{\leftarrow} \overleftarrow{h}_t^{(L)} + c) \quad (10)$$

where L is the number of layers. Intuitively, connecting the output layer to only the last hidden layer forces the architecture to capture enough high-level information at the final layer for producing the appropriate output-layer decision.

Training a deep RNN can be conceptualized as interleaved applications of the conventional backpropagation across multiple layers, and backpropagation through time within a single layer.

The unidirectional and bidirectional deep RNNs are depicted in the bottom half of Figure 1.

Hypotheses. In general, we expected that the deep RNNs would show the most improvement over shallow RNNS for ESEs — phrases that implicitly convey subjectivity. Existing research has shown that these are harder to identify than direct expressions of subjectivity (DSEs): they are variable in length and involve terms that, in many (or most) contexts, are neutral with respect to sentiment and subjectivity. As a result, models that do a better job interpreting the context should be better at disambiguating subjective vs. non-subjective uses of phrases involving common words (e.g. "as usual", "in fact"). Whether or not deep RNNs would be powerful enough to outperform the state-of-the-art semiCRF was unclear, especially if the semiCRF is given access to the distributed word representations (embeddings) employed by the deep RNNs. In addition, the semiCRF has access to parse tree information and opinion lexicons, neither of which is available to the deep RNNs.

4 Experiments

Activation Units. We employ the standard softmax activation for the output layer: $g(x) = e^{x_i} / \sum_j e^{x_j}$. For the hidden layers we use the rectifier linear activation: $f(x) = \max\{0, x\}$. Experimentally, rectifier activation gives better performance, faster convergence, and sparse representations. Previous work also reported good results when training deep neural networks using rectifiers, without a pretraining step (Glorot et al., 2011).

Data. We use the MPQA 1.2 corpus (Wiebe et al., 2005) (535 news articles, 11,111 sentences) that is manually annotated with both DSEs and ESEs at the phrase level. As in previous work, we separate 135 documents as a development set and employ 10-fold CV over the remaining 400 documents. The development set is used during cross validation to do model selection.

Layers	$\|h\|$	Precision		Recall		F1	
		Prop.	Bin.	Prop.	Bin.	Prop	Bin.
Shallow	36	62.24	65.90	65.63*	73.89*	63.83	69.62
Deep 2	29	63.85*	67.23*	65.70*	**74.23***	64.70*	70.52*
Deep 3	25	63.53*	67.67*	65.95*	73.87*	64.57*	70.55*
Deep 4	22	**64.19***	**68.05***	**66.01***	73.76*	**64.96***	**70.69***
Deep 5	21	60.65	61.67	56.83	69.01	58.60	65.06
Shallow	200	62.78	66.28	65.66*	74.00*	64.09	69.85
Deep 2	125	62.92*	66.71*	66.45*	**74.70***	64.47	70.36
Deep 3	100	**65.56***	**69.12***	**66.73***	74.69*	**66.01***	**71.72***
Deep 4	86	61.76	65.64	63.52	72.88*	62.56	69.01
Deep 5	77	61.64	64.90	62.37	72.10	61.93	68.25

Table 2: Experimental evaluation of RNNs for DSE extraction

Layers	$\|h\|$	Precision		Recall		F1	
		Prop.	Bin.	Prop.	Bin.	Prop	Bin.
Shallow	36	51.34	59.54	57.60	72.89*	54.22	65.44
Deep 2	29	51.13	59.94	**61.20***	**75.37***	**55.63***	**66.64***
Deep 3	25	**53.14***	**61.46***	58.01	72.50	55.40*	66.36*
Deep 4	22	51.48	60.59*	59.25*	73.22	54.94	66.15*
Deep 5	21	49.67	58.42	48.98	65.36	49.25	61.61
Shallow	200	**52.20***	60.42*	58.11	72.64	54.75	65.75
Deep 2	125	51.75*	60.75*	60.69*	74.39*	55.77*	66.79*
Deep 3	100	52.04*	**60.50***	**61.71***	**76.02***	**56.26***	**67.18***
Deep 4	86	50.62*	58.41*	53.55	69.99	51.98	63.60
Deep 5	77	49.90*	57.82	52.37	69.13	51.01	62.89

Table 3: Experimental evaluation of RNNs for ESE extraction

Evaluation Metrics. We use precision, recall and F-measure for performance evaluation. Since the boundaries of expressions are hard to define even for human annotators (Wiebe et al., 2005), we use two soft notions of the measures: *Binary Overlap* counts every overlapping match between a predicted and true expression as correct (Breck et al., 2007; Yang and Cardie, 2012), and *Proportional Overlap* imparts a partial correctness, proportional to the overlapping amount, to each match (Johansson and Moschitti, 2010; Yang and Cardie, 2012). All statistical comparisons are done using a two-sided paired t-test with a confidence level of $\alpha = .05$.

Baselines (CRF and SEMICRF). As baselines, we use the CRF-based method of Breck et al. (2007) and the SEMICRF-based method of Yang and Cardie (2012), which is the state-of-the-art in opinion expression extraction. Features that the baselines use are words, part-of-speech tags and membership in a manually constructed opinion lexicon (within a [-1, +1] context window). Since SEMICRF relaxes the Markovian assumption and operates at the segment-level instead of the token-level, it also has access to parse trees of sentences to generate candidate segments (Yang and Cardie, 2012).

Word Vectors (+VEC). We also include versions of the baselines that have access to pre-trained word vectors. In particular, CRF+VEC employs word vectors as continuous features per every token. Since SEMICRF has phrase-level rather than word-level features, we simply take the mean of every word vector for a phrase-level vector representation for SEMICRF+VEC as suggested in Mikolov et al. (2013).

In all of our experiments, we keep the word vectors fixed (i.e. do not finetune) to reduce the degree of freedom of our models. We use the publicly available 300-dimensional word vectors of Mikolov et al. (2013), trained on part of the Google News dataset ($\sim$100B words). Preliminary experiments with other word vector representations such as Collobert-Weston (2008) embeddings or HLBL (Mnih and Hinton, 2007) provided poorer results ($\sim -3\%$ difference in proportional and binary F1).

Regularizer. We do not employ any regularization for smaller networks ($\sim$24,000 parameters) because we have not observed strong overfitting (i.e. the difference between training and test performance is small). Larger networks are regularized with the recently proposed dropout technique (Hinton et al., 2012): we randomly set entries of hidden representations to 0 with a probability called the dropout rate, which is tuned over the development set. Dropout prevents learned

	Model		Precision		Recall		F1	
			Prop.	Bin.	Prop.	Bin.	Prop	Bin.
DSE	CRF		74.96*	82.28*	46.98	52.99	57.74	64.45
	semiCRF		61.67	69.41	**67.22***	73.08*	64.27	71.15*
	CRF	+vec	**74.97***	**82.43***	49.47	55.67	59.59	66.44
	semiCRF	+vec	66.00	71.98	60.96	68.13	63.30	69.91
	Deep RNN	3 100	65.56	69.12	66.73*	**74.69***	**66.01***	**71.72***
ESE	CRF		56.08	68.36	42.26	51.84	48.10	58.85
	semiCRF		45.64	69.06	58.05	64.15	50.95	66.37*
	CRF	+vec	**57.15***	69.84*	44.67	54.38	50.01	61.01
	semiCRF	+vec	53.76	**70.82***	52.72	61.59	53.10	65.73
	Deep RNN	3 100	52.04	60.50	**61.71***	**76.02***	**56.26***	**67.18***

Table 4: Comparison of Deep RNNs to state-of-the-art (semi)CRF baselines for DSE and ESE detection

features from co-adapting, and it has been reported to yield good results when training deep neural networks (Krizhevsky et al., 2012; Dahl et al., 2013).

Network Training. We use the standard multiclass cross-entropy as the objective function when training the neural networks. We use stochastic gradient descent with momentum with a fixed learning rate (.005) and a fixed momentum rate (.7). We update weights after minibatches of 80 sentences. We run 200 epochs for training. Weights are initialized from small random uniform noise. We experiment with networks of various sizes, however we have the same number of hidden units across multiple forward and backward hidden layers of a single RNN. We do not employ a pre-training step; deep architectures are trained with the supervised error signal, even though the output layer is connected to only the final hidden layer. With these configurations, every architecture successfully converges without any oscillatory behavior. Additionally, we employ early stopping for the neural networks: out of all iterations, the model with the best development set performance (Proportional F1) is selected as the final model to be evaluated.

4.1 Results and Discussion

Bidirectional vs. Unidirectional. Although our focus is on bidirectional RNNs, we first confirm that the SHALLOW bidirectional RNN outperforms a (shallow) unidirectional RNN for both DSE and ESE recognition. To make the comparison fair, each network has the same number of total parameters: we use 65 hidden units for the unidirectional, and 36 for the bidirectional network, respectively. Results are as expected: the bidirectional RNN obtains higher F1 scores than the unidirectional RNN — 63.83 vs. 60.35 (proportional overlap) and 69.62 vs. 68.31 (binary overlap) for DSEs; 54.22 vs. 51.51 (proportional) and 65.44 vs. 63.65 (binary) for ESEs. All differences are statistically significant at the 0.05 level. Thus, we will not include comparisons to the unidirectional RNNs in the remaining experiments.

Adding Depth. Next, we quantitatively investigate the effects of adding depth to RNNs. Tables 2 and 3 show the evaluation of RNNs of various depths and sizes. In both tables, the first group networks have approximately 24,000 parameters and the second group networks have approximately 200,000 parameters. Since all RNNs within a group have approximately the same number of parameters, they grow narrower as they get deeper. Within each group, bold shows the best result with an asterisk denoting statistically indistinguishable performance with respect to the best. As noted above, all statistical comparisons use a two-sided paired t-test with a confidence level of $\alpha = .05$.

In both DSE and ESE detection and for larger networks (bottom set of results), 3-layer RNNs provide the best results. For smaller networks (top set of results), 2, 3 and 4-layer RNNs show equally good performance for certain sizes and metrics and, in general, adding additional layers degrades performance. This could be related to how we train the architectures as well as to the decrease in width of the networks. In general, we observe a trend of increasing performance as we increase the number of layers, until a certain depth.

deepRNNs vs. (semi)CRF. Table 4 shows comparison of the best deep RNNs to the previous best results in the literature. In terms of F-measure, DEEP RNN performs best for both DSE and ESE detection, achieving a new state-of-the-art performance for the more strict proportional overlap measure, which is harder to improve upon than the binary evaluation metric. SEMICRF, with its very high recall, performs comparably to the DEEP RNN on the binary metric. Note that RNNs do not have access to any features other than word vectors.

In general, CRFs exhibit high precision but low recall (CRFs have the best precision on both DSE and ESE detection) while SEMICRFs exhibit a high recall, low precision performance. Compared to SEMICRF, the DEEP RNNs produce an even higher recall but sometimes lower precision for ESE detection. This suggests that the methods are complementary, and can

(1)

The situation obviously remains fluid from hour to hour but it [seems to be] [going in the right direction]
DEEPRNN The situation [obviously] remains fluid from hour to hour but it [seems to be going in the right] direction
SHALLOW The situation [obviously] remains fluid from hour to hour but it [seems to be going in] the right direction
SEMICRF The situation [obviously remains fluid from hour to hour but it seems to be going in the right direction]

(2)

have always said this is a multi-faceted campaign [but equally] we have also said any future military action [would have to be based on evidence] , ...
DEEPRNN have always said this is a multi-faceted campaign but [equally we] have also said any future military action [would have to be based on evidence] , ...
SHALLOW have always said this is a multi-faceted [campaign but equally we] have also said any future military action would have to be based on evidence , ...
SEMICRF have always said this is a multi-faceted campaign but equally we have also said any future military action would have to be based on evidence , ...

(3)

Ruud Lubbers , the United Nations Commissioner for Refugees , said Afghanistan was [not yet] secure for aid agencies to operate in and " [not enough] " food had been taken into the country .
DEEPRNN Ruud Lubbers , the United Nations Commissioner for Refugees , said Afghanistan was [not yet] secure for aid agencies to operate in and " [not enough] " food had been taken into the country .
SHALLOW Ruud Lubbers , the United Nations Commissioner for Refugees , said Afghanistan was [not yet] secure for aid agencies to operate in and " [not enough] " food had been taken into the country .
SEMICRF Ruud Lubbers , the United Nations Commissioner for Refugees , said Afghanistan was not yet secure for aid agencies to operate in and " not enough " food had been taken into the country .

Figure 2: Examples of output. In each set, the gold-standard annotations are shown in the first line.

potentially be even more powerful when combined in an ensemble method.

Word vectors. Word vectors help CRFs on both precision and recall on both tasks. However, SEMICRFs become more conservative with word vectors, producing higher precision and lower recall on both tasks. This sometimes hurts overall F-measure.

Among the (SEMI)CRF-based methods, SEMICRF obtains the highest F1 score for DSEs and for ESEs using the softer metric; SEMICRF+VEC performs best for ESEs according to the stricter proportional overlap measure.

Network size. Finally, we observe that even small networks (such as 4-layer deep RNN for DSE and 2-layer deep RNN for ESE) outperform conventional CRFs. This suggests that with the help of good word vectors, we can train compact but powerful sequential neural models.

When examining the output, we see some systematic differences between the previously top-performing SEMICRF and the RNN-based models. (See Figure 2.) First, SEMICRF often identifies excessively long subjective phrases as in Example 1. Here, none of the models exactly matches the gold standard, but the RNNs are much closer. And all three models appear to have identified an ESE that was mistakenly omitted by the human annotator — "obviously". At the same time, the SEMICRF sometimes entirely misses subjective expressions that the RNNs identify — this seems to occur when there are no clear indications of sentiment in the subjective expression. The latter can be seen in Examples 2 and 3, in which the SEMICRF does not identify "but equally", "would have to be based on evidence", "not yet", and "not enough".

We also observe evidence of the power of the DEEPRNN over the SHALLOWRNN in Examples 4 and 5. (See Figure 3.) In contrast to Figure 2, Figure 3 distinguishes subjective expressions that are (correctly) assigned an initial Begin label from those that consist only of Inside labels[2] — the latter are shown in ALL CAPS and indicate some degree of confusion in the model that produced them. In Example 4, SHALLOWRNN exhibits *some* evidence for each ESE — it labels one or more tokens as Inside an ESE ("any" and "time"). But it does not explicitly tag the beginning of the ESE. DEEPRNN does better, identifying the first ESE in its entirety ("in any case") and identifying more words as being Inside the second ESE ("it is high time). A similar situation occurs in Example 5.

5 Conclusion

In this paper we have explored an application of deep recurrent neural networks to the task of sentence-level opinion expression extraction. We empirically evaluated deep RNNs against conventional, shallow RNNs that have only a single hidden layer. We also compared our models with previous (semi)CRF-based approaches.

Experiments showed that deep RNNs outperformed shallow RNNs on both DSE and ESE extrac-

[2]Sequences of I's are decoded as the associated DSE or ESE even though they lack the initial B.

(4)

[In any case] , [it is high time] that a social debate be organized ...

DEEPRNN [In any case] , it is HIGH TIME that a social debate be organized ...

SHALLOW In ANY case , it is high TIME that a social debate be organized ...

(5)

Mr. Stoiber [has come a long way] from his refusal to [sacrifice himself] for the CDU in an election that [once looked impossible to win] , through his statement that he would [under no circumstances] run against the wishes...

DEEPRNN Mr. Stoiber [has come a long way from] his [refusal to sacrifice himself] for the CDU in an election that [once looked impossible to win] , through his statement that he would [under no circumstances run against] the wishes...

SHALLOW Mr. Stoiber has come A LONG WAY FROM his refusal to sacrifice himself for the CDU in an election that [once looked impossible] to win , through his statement that he would under NO CIRCUMSTANCES run against the wishes...

Figure 3: DEEPRNN Output vs. SHALLOWRNN Output. In each set of examples, the gold-standard annotations are shown in the first line. Tokens assigned a label of Inside with no preceding Begin tag are shown in ALL CAPS.

tion. Furthermore, deep RNNs outperformed previous (semi)CRF baselines, achieving new state-of-the-art results for fine-grained on opinion expression extraction.

We have trained our deep networks without any pre-training and with only the last hidden layer connected to the output layer. One potential future direction is to explore the effects of pre-training on the architecture. Pre-training might help to exploit the additional representational power available in deeper networks. Another direction is to investigate the impact of fine-tuning the word vectors during supervised training. Additionally, alternative notions of depth that are orthogonal to stacking, as in Pascanu et al. (2013) can be investigated for this task.

Acknowledgments

This work was supported in part by NSF grant IIS-1314778 and DARPA DEFT Grant FA8750-13-2-0015. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, DARPA or the U.S. Government.

References

Yoshua Bengio. 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

Eric Breck, Yejin Choi, and Claire Cardie. 2007. Identifying expressions of opinion in context. In *IJCAI*, pages 2683–2688.

Yejin Choi and Claire Cardie. 2010. Hierarchical sequential learning for extracting opinions and their attributes. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 269–274. Association for Computational Linguistics.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proceedings of HLT/EMNLP*, pages 355–362. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

George E Dahl, Tara N Sainath, and Geoffrey E Hinton. 2013. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE.

Salah El Hihi and Yoshua Bengio. 1995. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in Neural Information Processing Systems*, pages 493–499.

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

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume*, volume 15, pages 315–323.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 190–198.

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

Richard Johansson and Alessandro Moschitti. 2010. Syntactic and semantic structure for opinion expression detection. In *Proceedings of the Fourteenth Conference on Computational Natural Lan-*

guage Learning, pages 67–76. Association for Computational Linguistics.

Richard Johansson and Alessandro Moschitti. 2011. Extracting opinion expressions and their polarities: exploration of pipelines and joint models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 101–106. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, volume 1, page 4.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. Interspeech.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, JH Cernocky, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

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

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

M Arthur Munson, Claire Cardie, and Rich Caruana. 2005. Optimizing to arbitrary nlp metrics using ensemble selection. In *Proceedings of HLT/EMNLP*, pages 539–546. Association for Computational Linguistics.

Razvan Pascanu, Çağlar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Jürgen Schmidhuber. 1992. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT/EMNLP*, pages 347–354. Association for Computational Linguistics.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1335–1345. Association for Computational Linguistics.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of ACL*.

The Inside-Outside Recursive Neural Network model for Dependency Parsing

Phong Le and **Willem Zuidema**
Institute for Logic, Language, and Computation
University of Amsterdam, the Netherlands
{p.le,zuidema}@uva.nl

Abstract

We propose the first implementation of an infinite-order generative dependency model. The model is based on a new recursive neural network architecture, the Inside-Outside Recursive Neural Network. This architecture allows information to flow not only bottom-up, as in traditional recursive neural networks, but also top-down. This is achieved by computing content as well as context representations for any constituent, and letting these representations interact. Experimental results on the English section of the Universal Dependency Treebank show that the infinite-order model achieves a perplexity seven times lower than the traditional third-order model using counting, and tends to choose more accurate parses in k-best lists. In addition, reranking with this model achieves state-of-the-art unlabelled attachment scores and unlabelled exact match scores.

1 Introduction

Estimating probability distributions is the core issue in modern, data-driven natural language processing methods. Because of the traditional definition of discrete probability

$$Pr(A) \equiv \frac{\text{the number of times A occurs}}{\text{the size of event space}}$$

counting has become a standard method to tackle the problem. When data are sparse, smoothing techniques are needed to adjust counts for non-observed or rare events. However, successful use of those techniques has turned out to be an art. For instance, much skill and expertise is required to create reasonable reduction lists for back-off, and to avoid impractically large count tables, which store events and their counts.

An alternative to counting for estimating probability distributions is to use neural networks. Thanks to recent advances in deep learning, this approach has recently started to look very promising again, with state-of-the-art results in sentiment analysis (Socher et al., 2013), language modelling (Mikolov et al., 2010), and other tasks. The Mikolov et al. (2010) work, in particular, demonstrates the advantage of neural-network-based approaches over counting-based approaches in language modelling: it shows that recurrent neural networks are capable of capturing long histories efficiently and surpass standard n-gram techniques (e.g., Kneser-Ney smoothed 5-gram).

In this paper, keeping in mind the success of these models, we compare the two approaches. Complementing recent work that focused on such a comparison for the case of finding appropriate word vectors (Baroni et al., 2014), we focus here on models that involve more complex, hierarchical structures. Starting with existing generative models that use counting to estimate probability distributions over constituency and dependency parses (e.g., Eisner (1996b), Collins (2003)), we develop an alternative based on recursive neural networks. This is a non-trivial task because, to our knowledge, no existing neural network architecture can be used in this way. For instance, classic recurrent neural networks (Elman, 1990) unfold to left-branching trees, and are not able to process arbitrarily shaped parse trees that the counting approaches are applied to. Recursive neural networks (Socher et al., 2010) and extensions (Socher et al., 2012; Le et al., 2013), on the other hand, do work with trees of arbitrary shape, but process them in a bottom-up manner. The probabilities we need to estimate are, in contrast, defined by top-down generative models, or by models that require information flows in both directions (e.g., the probability of generating a node depends on the whole fragment rooted at its just-generated sis-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 729–739,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

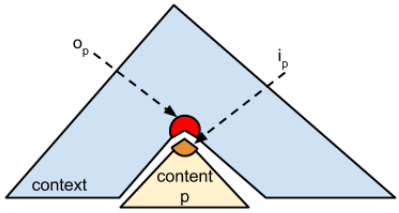


Figure 1: Inner ($\mathbf{i}_p$) and outer ($\mathbf{o}_p$) representations at the node that covers constituent p. They are vectorial representations of p's content and context, respectively.

ter).

To tackle this problem, we propose a new architecture: the Inside-Outside Recursive Neural Network (IORNN) in which information can flow not only bottom-up but also top-down, inward and outward. The crucial innovation in our architecture is that every node in a hierarchical structure is associated with two vectors: one vector, the *inner representation*, representing the content under that node, and another vector, the *outer representation*, representing its context (see Figure 1). Inner representations can be computed bottom-up; outer representations, in turn, can be computed top-down. This allows information to flow in any direction, depending on the application, and makes the IORNN a natural tool for estimating probabilities in tree-based generative models.

We demonstrate the use of the IORNN by applying it to an ∞-order generative dependency model which is impractical for counting due to the problem of data sparsity. Counting, instead, is used to estimate a third-order generative model as in Sangati et al. (2009) and Hayashi et al. (2011). Our experimental results show that our new model not only achieves a seven times lower perplexity than the third-order model, but also tends to choose more accurate candidates in k-best lists. In addition, reranking with this model achieves state-of-the-art scores on the task of supervised dependency parsing.

The outline of the paper is following. Firstly, we give an introduction to Eisner's generative model in Section 2. Then, we present the third-order model using counting in Section 3, and propose the IORNN in Section 4. Finally, in Section 5 we show our experimental results.

2 Eisner's Generative Model

Eisner (1996b) proposed a generative model for dependency parsing. The generation process is top-down: starting at the ROOT, it generates left dependents and then right dependents for the ROOT. After that, it generates left dependents and right dependents for each of ROOT's dependents. The process recursively continues until there is no further dependent to generate. The whole process is captured in the following formula

$$P(T(H)) = \prod_{l=1}^{L} P\left(H_l^L | \mathcal{C}_{H_l^L}\right) P\left(T(H_l^L)\right) \times \prod_{r=1}^{R} P\left(H_r^R | \mathcal{C}_{H_r^R}\right) P\left(T(H_r^R)\right) \quad (1)$$

where H is the current head, $T(N)$ is the fragment of the dependency parse rooted in N, and $\mathcal{C}_N$ is the context in which N is generated. H^L, H^R are respectively H's left dependents and right dependents, plus EOC (End-Of-Children), a special token to indicate that there are no more dependents to generate. Thus, $P(T(ROOT))$ is the probability of generating the entire dependency structure T. We refer to $\langle H_l^L, \mathcal{C}_{H_l^L}\rangle$, $\langle H_r^R, \mathcal{C}_{H_r^R}\rangle$ as "events", and $\langle \mathcal{C}_{H_l^L}\rangle$, $\langle \mathcal{C}_{H_r^R}\rangle$ as "conditioning contexts".

In order to avoid the problem of data sparsity, the conditioning context in which a dependent D is generated should capture only part of the fragment generated so far. Based on the amount of information that contexts hold, we can define the *order* of a generative model (see Hayashi et al. (2011, Table 3) for examples)

- first-order: $\mathcal{C}_D^1$ contains the head H,
- second-order: $\mathcal{C}_D^2$ contains H and the just-generated sibling S,
- third-order: $\mathcal{C}_D^3$ contains H, S, the sibling S' before S (tri-sibling); or H, S and the grand-head G (the head of H) (grandsibling) (the fragment enclosed in the blue doted contour in Figure 2),
- ∞-order: $\mathcal{C}_D^\infty$ contains all of D's ancestors, theirs siblings, and its generated siblings (the fragment enclosed in the red dashed contour in Figure 2).

In the original models (Eisner, 1996a), each dependent D is a 4-tuple $\langle dist, w, c, t\rangle$

- $dist(H, D)$ the distance between D and its head H, represented as one of the four ranges 1, 2, 3-6, 7-∞.

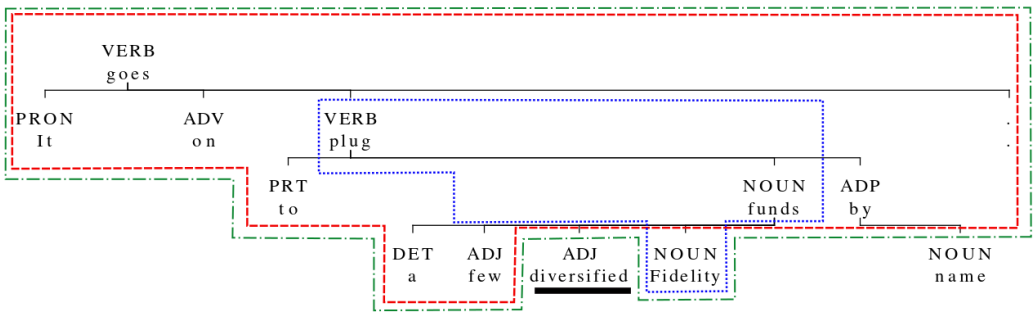


Figure 2: Example of different orders of context of "diversified". The blue dotted shape corresponds to the third-order outward context, while the red dashed shape corresponds to the ∞-order left-to-right context. The green dot-dashed shape corresponds to the context to compute the outer representation.

- $\underline{word}(D)$ the lowercase version of the word of D,
- $\underline{cap}(D)$ the capitalisation feature of the word of D (all letters are lowercase, all letters are uppercase, the first letter is uppercase, the first letter is lowercase),
- $\underline{tag}(D)$ the POS-tag of D,

Here, to make the dependency complete, $\underline{deprel}(D)$, the dependency relation of D (e.g., SBJ, DEP), is also taken into account.

3 Third-order Model with Counting

The third-order model we suggest is similar to the grandsibling model proposed by Sangati et al. (2009) and Hayashi et al. (2011). It defines the probability of generating a dependent $D = \langle dist, d, w, c, t \rangle$ as the product of the distance-based probability and the probabilities of generating each of its components (d, t, w, c, denoting dependency relation, POS-tag, word and capitalisation feature, respectively). Each of these probabilities is smoothed using back-off according to the given reduction lists (as explained below).

$$
\begin{aligned}
&P(D|\mathcal{C}_D) \\
&= P(dist(H,D), dwct(D)|H,S,G,dir) \\
&= P(d(D)|H,S,G,dir) \\
&\quad \text{reduction list: } \left\{\begin{array}{|l|}\hline tw(H),tw(S),tw(G),dir \\ \hline tw(H),tw(S),t(G),dir \\ \hline \left\{\begin{array}{l} tw(H),t(S),t(G),dir \\ t(H),tw(S),t(G),dir \end{array}\right. \\ \hline t(H),t(S),t(G),dir \\ \hline \end{array}\right. \\
&\times P(t(D)|d(D),H,S,G,dir) \\
&\quad \text{reduction list: } \left\{\begin{array}{|l|}\hline d(D),dtw(H),t(S),dir \\ \hline d(D),d(H),t(S),dir \\ \hline d(D),d(D),dir \\ \hline \end{array}\right. \\
&\times P(w(D)|dt(D),H,S,G,dir) \\
&\quad \text{reduction list: } \left\{\begin{array}{|l|}\hline dtw(H),t(S),dir \\ \hline dt(H),t(S),dir \\ \hline \end{array}\right. \\
&\times P(c(D)|dtw(D),H,S,G,dir) \\
&\quad \text{reduction list: } \left\{\begin{array}{|l|}\hline tw(D),d(H),dir \\ \hline tw(D),dir \\ \hline \end{array}\right. \\
&\times P(dist(H,D)|dtwc(D),H,S,G,dir) \qquad (2) \\
&\quad \text{reduction list: } \left\{\begin{array}{|l|}\hline dtw(D),dt(H),t(S),dir \\ \hline dt(D),dt(H),t(S),dir \\ \hline \end{array}\right.
\end{aligned}
$$

The reason for generating the dependency relation first is based on the similarity between relation/dependent and role/filler: we generate a role and then choose a filler for that role.

Back-off The back-off parameters are identical to Eisner (1996b). To estimate the probability $P(A|context)$ given a reduction list $L = (l_1, l_2, ..., l_n)$ of $context$, let

$$
p_i = \begin{cases} \frac{count(A,l_i)+0.005}{count(l_i)+0.5} & \text{if } i = n \\ \frac{count(A,l_i)+3p_{i+1}}{count(l_i)+3} & \text{otherwise} \end{cases}
$$

then $P(A|context) = p_1$.

4 The Inside-Outside Recursive Neural Network

In this section, we first describe the Recursive Neural Network architecture of Socher et al. (2010) and then propose an extension we call the Inside-Outside Recursive Neural Network (IORNN). The IORNN is a general architecture for trees, which works with tree-based generative models including those employed by Eisner (1996b) and Collins (2003). We then explain how to apply the IORNN to the ∞-order model. Note that for the present paper we are only concerned with the problem of computing the probability of

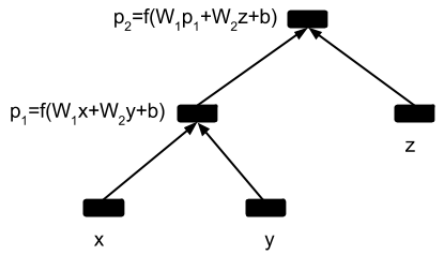


Figure 3: Recursive Neural Network (RNN).

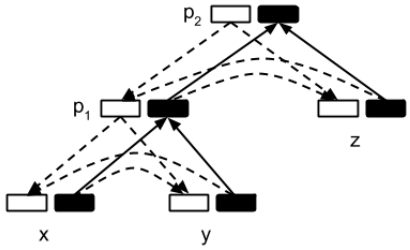


Figure 4: Inside-Outside Recursive Neural Network (IORNN). Black rectangles correspond to inner representations, white rectangles correspond to outer representations.

a tree; we assume an independently given parser is available to assign a syntactic structure, or multiple candidate structures, to an input string.

4.1 Recursive Neural Network

The architecture we propose can best be understood as an extension of the Recursive Neural Networks (RNNs) proposed by Socher et al. (2010), that we mentioned above. In order to see how an RNN works, consider the following example. Assume that there is a constituent with parse tree $(p_2\ (p_1\ x\ y)\ z)$ (Figure 3), and that $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$ are the (inner) representations of the three words x, y and z, respectively. We use a neural network which consists of a weight matrix $\mathbf{W}_1 \in \mathbb{R}^{n \times n}$ for left children and a weight matrix $\mathbf{W}_2 \in \mathbb{R}^{n \times n}$ for right children to compute the vector for a parent node in a bottom up manner. Thus, we compute p_1 as follows

$$\mathbf{p}_1 = f(\mathbf{W}_1\mathbf{x} + \mathbf{W}_2\mathbf{y} + \mathbf{b})$$

where $\mathbf{b}$ is a bias vector and f is an activation function (e.g., *tanh* or *logistic*). Having computed p_1, we can then move one level up in the hierarchy and compute p_2:

$$\mathbf{p}_2 = f(\mathbf{W}_1\mathbf{p}_1 + \mathbf{W}_2\mathbf{z} + \mathbf{b})$$

This process is continued until we reach the root node. The RNN thus computes a single vector for each node p in the tree, representing the *content* under that node. It has in common with logical semantics that representations for compounds (here *xyz*) are computed by recursively applying a composition function to meaning representations of the parts. It is difficult to characterise the expressivity of the resulting system in logical terms, but recent work suggests it is surprisingly powerful (e.g., Kanerva (2009)).

4.2 IORNN

We extend the RNN-architecture by adding a second vector to each node, representing the *context* of the node, shown as white rectangles in figure 4. The job of this second vector, the outer representation, is to summarize all information about the context of node p so that we can either predict its content (i.e., predict an inner representation), or pass on this information to the daughters of p (i.e., compute outer representations of these daughters). Outer representations thus allow information to flow top-down.

We explain the operation of the resulting *Inside-Outside Recursive Neural Network* in terms of the same example parse tree $(p_2\ (p_1\ x\ y)\ z)$ (see Figure 4). Each node u in the syntactic tree carries two vectors $\mathbf{o}_u$ and $\mathbf{i}_u$, the outer representation and inner representation of the constituent that is covered by the node.

Computing inner representations Inner representations are computed from the bottom up. We assume for every word w an inner representation $\mathbf{i}_w \in \mathbb{R}^n$. The inner representation of a non-terminal node, say p_1, is given by

$$\mathbf{i}_{p_1} = f(\mathbf{W}_1^i\mathbf{i}_x + \mathbf{W}_2^i\mathbf{i}_y + \mathbf{b}^i)$$

where $\mathbf{W}_1^i, \mathbf{W}_2^i$ are $n \times n$ real matrices, $\mathbf{b}^i$ is a bias vector, and f is an activation function, e.g. $tanh$. (This is the same as the computation of non-terminal vectors in the RNNs.) The inner representation of a parent node is thus a function of the inner representations of its children.

Computing outer representations Outer representations are computed from the top down. For a node which is not the root, say p_1, the outer repre-

sentation is given by

$$\mathbf{o}_{p_1} = g(\mathbf{W}_1^o \mathbf{o}_{p_2} + \mathbf{W}_2^o \mathbf{i}_z + \mathbf{b}^o)$$

where $\mathbf{W}_1^o, \mathbf{W}_2^o$ are $n \times n$ real matrices, $\mathbf{b}^o$ is a bias vector, and g is an activation function. The outer representation of a node is thus a function of the outer representation of its parent and the inner representation of its sisters.

If there is information about the external context of the utterance that is being processed, this information determines the outer representation of the root node $\mathbf{o}_{root}$. In our first experiments reported here, no such information was assumed to be available. In this case, a random value $\mathbf{o}_\emptyset$ is chosen at initialisation and assigned to the root nodes of all utterances; this value is then adjusted by the learning process discussed below.

Training Training the IORNN is to minimise an objective function $J(\theta)$ which depends on the purpose of usage where θ is the set of parameters. To do so, we compute the gradient $\partial J/\partial\theta$ and apply the gradient descent method. The gradient is effectively computed thanks to back-propagation through structure (Goller and Küchler, 1996). Following Socher et al. (2013), we use AdaGrad (Duchi et al., 2011) to update the parameters.

4.3 The ∞-order Model with IORNN

The RNN and IORNN are defined for context-free trees. To apply the IORNN architecture to dependency parses we need to adapt the definitions somewhat. In particular, in the generative dependency model, every step in the generative story involves the decision to generate a specific word while the span of the subtree that this word will dominate only becomes clear when all dependents are generated. We therefore introduce *partial outer representation* as a representation of the current context of a word in the generative process, and compute the final outer representation only when all its siblings have been generated.

Consider an example of head h and its dependents x, y (we ignore directions for simplicity) in Figure 5. Assume that we are in the state in the generative process where the generation of h is complete, i.e. we know its inner and outer representations $\mathbf{i}_h$ and $\mathbf{o}_h$. Now, when generating h's first dependent x (see Figure 5-a), we first compute x's *partial* outer representation (representing its context at this stage in the process), which is a function of the outer representation of the head (representing the head's context) and the inner representation of the head (representing the content of the head word):

$$\bar{\mathbf{o}}_1 = f(\mathbf{W}_{hi}\mathbf{i}_h + \mathbf{W}_{ho}\mathbf{o}_h + \mathbf{b}_o)$$

where $\mathbf{W}_{hi}, \mathbf{W}_{ho}$ are $n \times n$ real matrices, $\mathbf{b}_o$ is a bias vector, f is an activation function.

With the context of the first dependent determined, we can proceed and generate its content. For this purpose, we assume a separate weight matrix $\mathbf{W}$, trained (as explained below) to predict a specific word given a (partial) outer representation. To compute a proper probability for word x, we use the softmax function:

$$softmax(x, \bar{\mathbf{o}}_1) = \frac{e^{u(x,\bar{\mathbf{o}}_1)}}{\sum_{w \in V} e^{u(w,\bar{\mathbf{o}}_1)}}$$

where $\left[u(w_1, \bar{\mathbf{o}}_1), ..., u(w_{|V|}, \bar{\mathbf{o}}_1)\right]^T = \mathbf{W}\bar{\mathbf{o}}_1 + \mathbf{b}$ and V is the set of all possible dependents.

Note that since $\mathbf{o}_h$, the outer representation of h, represents the entire dependency structure generated up to that point, $\bar{\mathbf{o}}_1$ is a vectorial representation of the ∞-order context generating the first dependent (like the fragment enclosed in the red dashed contour in Figure 2). The softmax function thus estimates the probability $P(D = x|\mathcal{C}_D^\infty)$.

The next step, now that x is generated, is to compute the partial outer representation for the second dependent (see Figure 5-b)

$$\bar{\mathbf{o}}_2 = f(\mathbf{W}_{hi}\mathbf{i}_h + \mathbf{W}_{ho}\mathbf{o}_h + \mathbf{W}_{dr(x)}\mathbf{i}_x + \mathbf{b}_o)$$

where $\mathbf{W}_{dr(x)}$ is a $n \times n$ real matrix specific for the dependency relation of x with h.

Next y is generated (using the softmax function above), and the partial outer representation for the third dependent (see Figure 5-c) is computed:

$$\bar{\mathbf{o}}_3 = f(\mathbf{W}_{hi}\mathbf{i}_h + \mathbf{W}_{ho}\mathbf{o}_h + \frac{1}{2}\left(\mathbf{W}_{dr(x)}\mathbf{i}_x + \mathbf{W}_{dr(y)}\mathbf{i}_y\right) + \mathbf{b}_o)$$

Since the third dependent is the End-of-Children symbol (EOC), the process of generating dependents for h stops. We can then return to x and y to replace the partial outer representations with complete outer representations[1] (see

[1] According to the IORNN architecture, to compute the outer representation of a node, the inner representations of the whole fragments rooting at its sisters must be taken into account. Here, we replace the inner representation of a fragment by the inner representation of its root since the meaning of a phrase is often dominated by the meaning of its head.

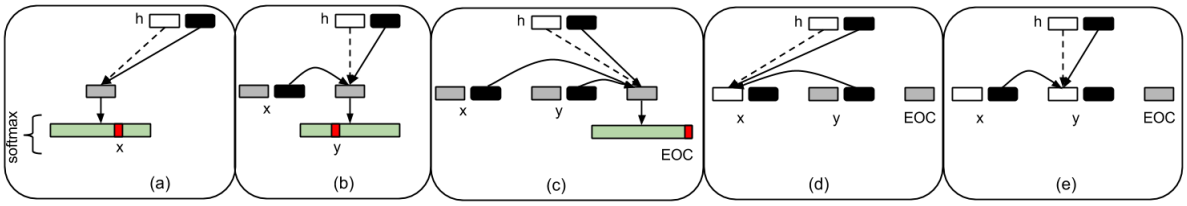


Figure 5: Example of applying IORNN to dependency parsing. Black, grey, white boxes are respectively inner, partial outer, and outer representations. For simplicity, only links related to the current computation are drawn (see text).

Figure 5-d,e):

$$\mathbf{o}_x = f(\mathbf{W}_{hi}\mathbf{i}_h + \mathbf{W}_{ho}\mathbf{o}_h + \mathbf{W}_{dr(y)}\mathbf{i}_y + \mathbf{b}_o)$$
$$\mathbf{o}_y = f(\mathbf{W}_{hi}\mathbf{i}_h + \mathbf{W}_{ho}\mathbf{o}_h + \mathbf{W}_{dr(x)}\mathbf{i}_x + \mathbf{b}_o)$$

In general, if u is the first dependent of h then

$$\bar{\mathbf{o}}_u = f(\mathbf{W}_{hi}\mathbf{i}_h + \mathbf{W}_{ho}\mathbf{o}_h + \mathbf{b}_o)$$

otherwise

$$\bar{\mathbf{o}}_u = f(\mathbf{W}_{hi}\mathbf{i}_h + \mathbf{W}_{ho}\mathbf{o}_h + \mathbf{b}_o + \frac{1}{|\bar{\mathcal{S}}(u)|}\sum_{v\in\bar{\mathcal{S}}(u)}\mathbf{W}_{dr(v)}\mathbf{i}_v)$$

where $\bar{\mathcal{S}}(u)$ is the set of u's sisters generated before it. And, if u is the only dependent of h (ignoring EOC) then

$$\mathbf{o}_u = f(\mathbf{W}_{hi}\mathbf{i}_h + \mathbf{W}_{ho}\mathbf{o}_h + \mathbf{b}_o)$$

otherwise

$$\mathbf{o}_u = f(\mathbf{W}_{hi}\mathbf{i}_h + \mathbf{W}_{ho}\mathbf{o}_h + \mathbf{b}_o + \frac{1}{|\mathcal{S}(u)|}\sum_{v\in\mathcal{S}(u)}\mathbf{W}_{dr(v)}\mathbf{i}_v)$$

where $\mathcal{S}(u)$ is the set of u's sisters.

We then continue this process to generate dependents for x and y until the process stops.

Inner Representations In the calculation of the probability of generating a word, described above, we assumed inner representations of all possible words to be given. These are, in fact, themselves a function of vector representations for the words (in our case, the word vectors are initially borrowed from Collobert et al. (2011)), the POS-tags and capitalisation features. That is, the inner representation at a node h is given by:

$$\mathbf{i}_h = f\left(\mathbf{W}_w\mathbf{w}_h + \mathbf{W}_p\mathbf{p}_h + \mathbf{W}_c\mathbf{c}_h\right)$$

where $\mathbf{W}_w \in R^{n\times d_w}$, $\mathbf{W}_p \in R^{n\times d_p}$, $\mathbf{W}_c \in R^{n\times d_c}$, $\mathbf{w}_h$ is the word vector of h, and $\mathbf{p}_h, \mathbf{c}_h$ are respectively binary vectors representing the POS-tag and capitalisation feature of h.

Training Training this IORNN is to minimise the following objective function which is the regularised cross-entropy

$$J(\theta) = -\frac{1}{m}\sum_{T\in\mathcal{D}}\sum_{w\in T}\log(P(w|\bar{\mathbf{o}}_w)) + \frac{1}{2}\left(\lambda_W\|\theta_W\|^2 + \lambda_L\|\theta_L\|^2\right)$$

where $\mathcal{D}$ is the set of training dependency parses, m is the number of dependents; θ_W, θ_L are the weight matrix set and the word embeddings ($\theta = (\theta_W, \theta_L)$); λ_W, λ_L are regularisation hyper-parameters.

Implementation We decompose a dependent D into four features: dependency relation, POS-tag, lowercase version of word, capitalisation feature of word. We then factorise $P(D|\mathcal{C}_D^\infty)$ similarly to Section 3, where each component is estimated by a softmax function.

5 Experiments

In our experiments, we convert the Penn Treebank to dependencies using the Universal dependency annotation (McDonald et al., 2013)[2]; this yields a dependency tree corpus we label PTB-U. In order to compare with other systems, we also experiment with an alternative conversion using the head rules of Yamada and Matsumoto (2003)[3]; this yields a dependency tree corpus we label PTB-YM. Sections 2-21 are used for training, section 22 for development, and section 23 for testing. For the PTB-U, the gold POS-tags are used. For the PTB-YM, the development and test sets are tagged by the Stanford POS-tagger[4] trained on the whole

[2] https://code.google.com/p/uni-dep-tb/

[3] http://stp.lingfil.uu.se/~nivre/research/Penn2Malt.html

[4] http://nlp.stanford.edu/software/tagger.shtml

	Perplexity
3rd-order model	1736.73
∞-order model	**236.58**

Table 1: Perplexities of the two models on PTB-U-22.

training data, whereas 10-way jackknifing is used to generate tags for the training set.

The vocabulary for both models, the third-order model and the ∞-order model, is taken as a list of words occurring more than two times in the training data. All other words are labelled 'UNKNOWN' and every digit is replaced by '0'. For the IORNN used by the ∞-order model, we set $n = 200$, and define f as the $tanh$ activation function. We initialise it with the 50-dim word embeddings from Collobert et al. (2011) and train it with the learning rate 0.1, $\lambda_W = 10^{-4}, \lambda_L = 10^{-10}$.

5.1 Perplexity

We firstly evaluate the two models on PTB-U-22 using the perplexity-per-word metric

$$ppl(P) = 2^{-\frac{1}{N}\sum_{T\in\mathcal{D}}\log_2 P(T)}$$

where $\mathcal{D}$ is a set of dependency parses, N is the total number of words. It is worth noting that, the better P estimates the true distribution P^* of $\mathcal{D}$, the lower its perplexity is. Because Eisner's model with the $dist(H, D)$ feature (Equation 2) is leaky (the model allocates some probability to events that can never legally arise), this feature is discarded (only in this experiment).

Table 1 shows results. The perplexity of the third-order model is more than seven times higher than the ∞-order model. This reflects the fact that data sparsity is more problematic for counting than for the IORNN.

To investigate why the perplexity of the third-order model is so high, we compute the percentages of events extracted from the development set appearing more than twice in the training set. Events are grouped according to the reduction lists in Equation 2 (see Table 2). We can see that reductions at level 0 (the finest) for dependency relations and words seriously suffer from data sparsity: more than half of the events occur less than three times, or not at all, in the training data. We thus conclude that counting-based models heavily rely on carefully designed reduction lists for back-off.

back-off level	d	t	w	c
0	47.4	61.6	43.7	87.7
1	69.8	98.4	77.8	97.3
2	76.0, 89.5	99.7		
3	97.9			
total	76.1	86.6	60.7	92.5

Table 2: Percentages of events extracted from PTB-U-22 appearing more than twice in the training set. Events are grouped according to the reduction lists in Equation 2. d, t, w, c stand for dependency relation, POS-tag, word, and capitalisation feature.

5.2 Reranking

In the second experiment, we evaluate the two models in the reranking framework proposed by Sangati et al. (2009) on PTB-U. We used the MSTParser (with the 2nd-order feature mode) (McDonald et al., 2005) to generate k-best lists. Two evaluation metrics are labelled attachment score (LAS) and unlabelled attachment score (UAS), *including* punctuation.

Rerankers Given $\mathcal{D}(S)$, a k-best list of parses of a sentence S, we define the *generative* reranker

$$T^* = \arg\max_{T\in\mathcal{D}(S)} P(T(ROOT))$$

which is identical to Sangati et al. (2009). Moreover, as in many mixture-model-based approaches, we define the *mixture* reranker as a combination of the generative model and the MST discriminative model (Hayashi et al., 2011)

$$T^* = \arg\max_{T\in\mathcal{D}(S)} \alpha \log P(T(ROOT)) + (1-\alpha)s(S,T)$$

where $s(S, T)$ is the score given by the MSTParser, and $\alpha \in [0, 1]$.

Results Figure 6 shows UASs of the generative reranker on the development set. The MSTParser achieves 92.32% and the Oracle achieve 96.23% when $k = 10$. With the third-order model, the generative reranker performs better than the MSTParser when $k < 6$ and the maximum improvement is 0.17%. Meanwhile, with the ∞-order model, the generative reranker always gains higher UASs than the MSTParser, and with $k = 6$, the difference reaches 0.7%. Figure 7 shows UASs of the mixture reranker on the same set. α is optimised by searching with the step-size 0.005. Unsurprisingly, we observe improvements over the

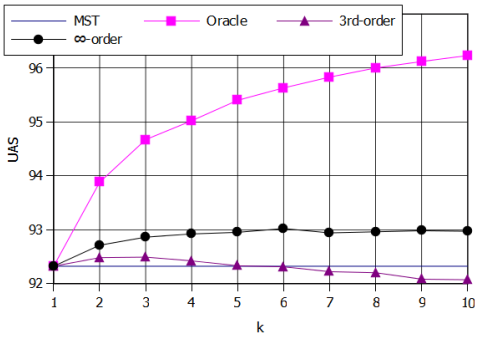


Figure 6: Performance of the generative reranker on PTB-U-22.

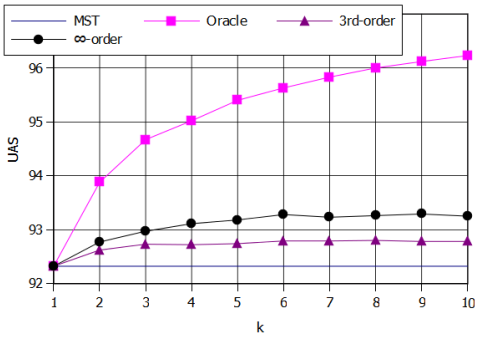


Figure 7: Performance of the mixture reranker on PTB-U-22. For each k, α was optimized with the step-size 0.005.

	LAS	UAS
MSTParser	89.97	91.99
Oracle ($k = 10$)	93.73	96.24
Generative reranker with		
3rd-order ($k = 3$)	90.27 (+0.30)	92.27 (+0.28)
∞-order ($k = 6$)	90.76 (+0.79)	92.83 (+0.84)
Mixture reranker with		
3rd-order ($k = 6$)	90.62 (+0.65)	92.62 (+0.63)
∞-order ($k = 9$)	**91.02 (+1.05)**	**93.08 (+1.09)**

Table 3: Comparison based on reranking on PTB-U-23. The numbers in the brackets are improvements over the MSTParser.

generative reranker as the mixture reranker can combine the advantages of the two models.

Table 3 shows scores of the two rerankers on the test set with the parameters tuned on the development set. Both the rerankers, either using third-order or ∞-order models, outperform the MSTParser. The fact that both gain higher improvements with the ∞-order model suggests that the IORNN surpasses counting.

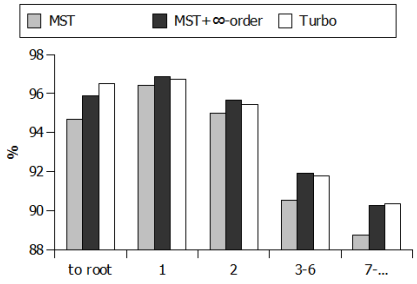


Figure 9: F1-scores of binned HEAD distance (PTB-U-23).

5.3 Comparison with other systems

We first compare the mixture reranker using the ∞-order model against the state-of-the-art dependency parser TurboParser (with the full mode) (Martins et al., 2013) on PTB-U-23. Table 4 shows LASs and UASs. When taking labels into account, the TurboParser outperforms the reranker. But without counting labels, the two systems perform comparably, and when ignoring punctuation the reranker even outperforms the TurboParser. This pattern is also observed when the exact match metrics are used (see Table 4). This is due to the fact that the TurboParser performs significantly better than the MSTParser, which generates k-best lists for the reranker, in labelling: the former achieves 96.03% label accuracy score whereas the latter achieves 94.92%.

One remarkable point is that reranking with the ∞-order model helps to improve the exact match scores 4% - 6.4% (see Table 4). Because the exact match scores correlate with the ability to handle global structures, we conclude that the IORNN is able to capture ∞-order contexts. Figure 8 shows distributions of correct-head accuracy over CPOS-tags and Figure 9 shows F1-scores of binned HEAD distance. Reranking with the ∞-order model is clearly helpful for all CPOS-tags and dependent-to-head distances, except a minor decrease on PRT.

We compare the reranker against other systems on PTB-YM-23 using the UAS metric ignoring punctuation (as the standard evaluation for English) (see Table 5). Our system performs slightly better than many state-of-the-art systems such as Martins et al. (2013) (a.k.a. TurboParser), Zhang and McDonald (2012), Koo and Collins (2010). It outperforms Hayashi et al. (2011) which is a reranker using a combination of third-order generative models with a variational model learnt

	LAS (w/o punc)	UAS (w/o punc)	LEM (w/o punc)	UEM (w/o punc)
MSTParser	89.97 (90.54)	91.99 (92.82)	32.37 (34.19)	42.80 (45.24)
w. ∞-order ($k = 9$)	91.02 (91.51)	**93.08** (**93.84**)	37.58 (39.16)	**49.17** (**51.53**)
TurboParser	**91.56** (**92.02**)	93.05 (93.70)	**40.65** (**41.72**)	48.05 (49.83)

Table 4: Comparison with the TurboParser on PTB-U-23. LEM and UEM are respectively the labelled exact match score and unlabelled exact match score metrics. The numbers in brackets are scores computed excluding punctuation.

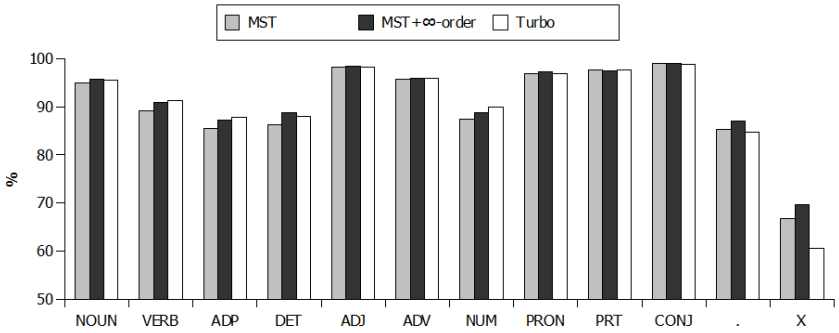


Figure 8: Distributions of correct-head accuracy over CPOS-tags (PTB-U-23).

System	UAS
Huang and Sagae (2010)	92.1
Koo and Collins (2010)	93.04
Zhang and McDonald (2012)	93.06
Martins et al. (2013)	93.07
Bohnet and Kuhn (2012)	93.39
Reranking	
Hayashi et al. (2011)	92.89
Hayashi et al. (2013)	93.12
MST+ ∞-order ($k = 12$)	**93.12**

Table 5: Comparison with other systems on PTB-YM-23 (excluding punctuation).

on the fly; performs equally with Hayashi et al. (2013) which is a discriminative reranker using the stacked technique; and slightly worse than Bohnet and Kuhn (2012), who develop a hybrid transition-based and graphical-based approach.

6 Related Work

Using neural networks to process trees was first proposed by Pollack (1990) in the Recursive Autoassociative Memory model which was used for unsupervised learning. Socher et al. (2010) later introduced the Recursive Neural Network architecture for supervised learning tasks such as syntactic parsing and sentiment analysis (Socher et al., 2013). Our IORNN is an extension of the RNN: the former can process trees not only bottom-up like the latter but also top-down.

Elman (1990) invented the simple recurrent neural network (SRNN) architecture which is capable of capturing very long histories. Mikolov et al. (2010) then applied it to language modelling and gained state-of-the-art results, outperforming the the standard n-gram techniques such as Kneser-Ney smoothed 5-gram. Our IORNN architecture for dependency parsing bears a resemblance to the SRNN in the sense that it can also capture long 'histories' in context representations (i.e., outer representations in our terminology). Moreover, the IORNN can be seen as a generalization of the SRNN since a left-branching tree is equivalent to a chain and vice versa.

The idea of letting parsing decisions depend on arbitrarily long derivation histories is also explored in Borensztajn and Zuidema (2011) and is related to parsing frameworks that allow arbitrarily large elementary trees (e.g., Scha (1990), O'Donnell et al. (2009), Sangati and Zuidema (2011), and van Cranenburgh and Bod (2013)).

Titov and Henderson (2007) were the first proposing to use deep networks for dependency parsing. They introduced a transition-based generative dependency model using incremental sigmoid belief networks and applied beam pruning for searching best trees. Differing from them, our work uses the IORNN architecture to rescore k-best candidates generated by an independent

graph-based parser, namely the MSTParser.

Reranking k-best lists was introduced by Collins and Koo (2005) and Charniak and Johnson (2005). Their rerankers are discriminative and for constituent parsing. Sangati et al. (2009) proposed to use a third-order generative model for reranking k-best lists of dependency parses. Hayashi et al. (2011) then followed this idea but combined generative models with a variational model learnt on the fly to rerank forests. In this paper, we also followed Sangati et al. (2009)'s idea but used an ∞-order generative model, which has never been used before.

7 Conclusion

In this paper, we proposed a new neural network architecture, the Inside-Outside Recursive Neural Network, that can process trees both bottom-up and top-down. The key idea is to extend the RNN such that every node in the tree has two vectors associated with it: an inner representation for its content, and an outer representation for its context. Inner and outer representations of any constituent can be computed simultaneously and interact with each other. This way, information can flow top-down, bottom-up, inward and outward. Thanks to this property, by applying the IORNN to dependency parses, we have shown that using an ∞-order generative model for dependency parsing, which has never been done before, is practical.

Our experimental results on the English section of the Universal Dependency Treebanks show that the ∞-order generative model approximates the true dependency distribution better than the traditional third-order model using counting, and tends to choose more accurate parses in k-best lists. In addition, reranking with this model even outperforms the state-of-the-art TurboParser on unlabelled score metrics.

Our source code is available at: `github.com/lephong/iornn-depparse`.

Acknowledgments

We thank Remko Scha and three anonymous reviewers for helpful comments.

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1.

Bernd Bohnet and Jonas Kuhn. 2012. The best of both worlds: a graph-based completion model for transition-based parsers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 77–87. Association for Computational Linguistics.

Gideon Borensztajn and Willem Zuidema. 2011. Episodic grammar: a computational model of the interaction between episodic and semantic memory in language processing. In *Proceedings of the 33d Annual Conference of the Cognitive Science Society (CogSci'11)*, pages 507–512. Lawrence Erlbaum Associates.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–66.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637.

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

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, pages 2121–2159.

Jason M. Eisner. 1996a. An empirical comparison of probability models for dependency grammar. Technical report, University of Pennsylvania Institute for Research in Cognitive Science.

Jason M Eisner. 1996b. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345. Association for Computational Linguistics.

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

Christoph Goller and Andreas Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *International Conference on Neural Networks*. IEEE.

Katsuhiko Hayashi, Taro Watanabe, Masayuki Asahara, and Yuji Matsumoto. 2011. Third-order variational reranking on packed-shared dependency

forests. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1479–1488. Association for Computational Linguistics.

Katsuhiko Hayashi, Shuhei Kondo, and Yuji Matsumoto. 2013. Efficient stacked dependency parsing by forest reranking. *Transactions of the Association for Computational Linguistics*, 1(1):139–150.

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

Pentti Kanerva. 2009. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2):139–159.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11. Association for Computational Linguistics.

Phong Le, Willem Zuidema, and Remko Scha. 2013. Learning from errors: Using vector-based compositional semantics for parse reranking. In *Proceedings Workshop on Continuous Vector Space Models and their Compositionality (at ACL 2013)*. Association for Computational Linguistics.

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98. Association for Computational Linguistics.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. *Proceedings of ACL, Sofia, Bulgaria*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Timothy J O'Donnell, Noah D Goodman, and Joshua B Tenenbaum. 2009. Fragment grammar: Exploring reuse in hierarchical generative processes. Technical report, Technical Report MIT-CSAIL-TR-2009-013, MIT.

Jordan B. Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46(1):77105.

Federico Sangati and Willem Zuidema. 2011. Accurate parsing with compact tree-substitution grammars: Double-DOP. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMLNP'11)*, pages 84–95. Association for Computational Linguistics.

Federico Sangati, Willem Zuidema, and Rens Bod. 2009. A generative re-ranking model for dependency parsing. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 238–241.

Remko Scha. 1990. Taaltheorie en taaltechnologie; competence en performance. In R. de Kort and G.L.J. Leerdam, editors, *Computertoepassingen in de Neerlandistiek*, pages 7–22. LVVN, Almere, the Netherlands. English translation at http://iaaa.nl/rs/LeerdamE.html.

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

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

Ivan Titov and James Henderson. 2007. A latent variable model for generative dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 144–155.

Andreas van Cranenburgh and Rens Bod. 2013. Discontinuous parsing with an efficient and accurate DOP model. In *Proceedings of the International Conference on Parsing Technologies (IWPT'13)*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of International Conference on Parsing Technologies (IWPT)*, pages 195–206.

Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 320–331. Association for Computational Linguistics.

A Fast and Accurate Dependency Parser using Neural Networks

Danqi Chen
Computer Science Department
Stanford University
danqi@cs.stanford.edu

Christopher D. Manning
Computer Science Department
Stanford University
manning@stanford.edu

Abstract

Almost all current dependency parsers classify based on millions of sparse indicator features. Not only do these features generalize poorly, but the cost of feature computation restricts parsing speed significantly. In this work, we propose a novel way of learning a neural network classifier for use in a greedy, transition-based dependency parser. Because this classifier learns and uses just a small number of dense features, it can work very fast, while achieving an about 2% improvement in unlabeled and labeled attachment scores on both English and Chinese datasets. Concretely, our parser is able to parse more than 1000 sentences per second at 92.2% unlabeled attachment score on the English Penn Treebank.

1 Introduction

In recent years, enormous parsing success has been achieved by the use of feature-based discriminative dependency parsers (Kübler et al., 2009). In particular, for practical applications, the speed of the subclass of transition-based dependency parsers has been very appealing.

However, these parsers are not perfect. First, from a statistical perspective, these parsers suffer from the use of millions of mainly poorly estimated feature weights. While in aggregate both lexicalized features and higher-order interaction term features are very important in improving the performance of these systems, nevertheless, there is insufficient data to correctly weight most such features. For this reason, techniques for introducing higher-support features such as word class features have also been very successful in improving parsing performance (Koo et al., 2008). Second, almost all existing parsers rely on a manually designed set of feature templates, which require a lot of expertise and are usually incomplete. Third, the use of many feature templates cause a less studied problem: in modern dependency parsers, most of the runtime is consumed not by the core parsing algorithm but in the feature extraction step (He et al., 2013). For instance, Bohnet (2010) reports that his baseline parser spends 99% of its time doing feature extraction, despite that being done in standard efficient ways.

In this work, we address all of these problems by using dense features in place of the sparse indicator features. This is inspired by the recent success of distributed word representations in many NLP tasks, e.g., POS tagging (Collobert et al., 2011), machine translation (Devlin et al., 2014), and constituency parsing (Socher et al., 2013). Low-dimensional, dense word embeddings can effectively alleviate sparsity by sharing statistical strength between similar words, and can provide us a good starting point to construct features of words and their interactions.

Nevertheless, there remain challenging problems of how to encode all the available information from the configuration and how to model higher-order features based on the dense representations. In this paper, we train a neural network classifier to make parsing decisions within a transition-based dependency parser. The neural network learns compact dense vector representations of words, part-of-speech (POS) tags, and dependency labels. This results in a fast, compact classifier, which uses only 200 learned dense features while yielding good gains in parsing accuracy and speed on two languages (English and Chinese) and two different dependency representations (CoNLL and Stanford dependencies). The main contributions of this work are: (i) showing the usefulness of dense representations that are learned within the parsing task, (ii) developing a neural network architecture that gives good accuracy and speed, and (iii) introducing a novel acti-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 740–750,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

vation function for the neural network that better captures higher-order interaction features.

2 Transition-based Dependency Parsing

Transition-based dependency parsing aims to predict a transition sequence from an initial configuration to some terminal configuration, which derives a target dependency parse tree, as shown in Figure 1. In this paper, we examine only greedy parsing, which uses a classifier to predict the correct transition based on features extracted from the configuration. This class of parsers is of great interest because of their efficiency, although they tend to perform slightly worse than the search-based parsers because of subsequent error propagation. However, our greedy parser can achieve comparable accuracy with a very good speed.[1]

As the basis of our parser, we employ the **arc-standard** system (Nivre, 2004), one of the most popular transition systems. In the arc-standard system, a *configuration* $c = (s, b, A)$ consists of a *stack* s, a *buffer* b, and a set of *dependency arcs* A. The initial configuration for a sentence $w_1, \ldots, w_n$ is $s = [\texttt{ROOT}], b = [w_1, \ldots, w_n], A = \emptyset$. A configuration c is terminal if the buffer is empty and the stack contains the single node `ROOT`, and the parse tree is given by A_c. Denoting s_i $(i = 1, 2, \ldots)$ as the i^{th} top element on the stack, and b_i $(i = 1, 2, \ldots)$ as the i^{th} element on the buffer, the arc-standard system defines three types of transitions:

- `LEFT-ARC`(l): adds an arc $s_1 \rightarrow s_2$ with label l and removes s_2 from the stack. Precondition: $|s| \geq 2$.
- `RIGHT-ARC`(l): adds an arc $s_2 \rightarrow s_1$ with label l and removes s_1 from the stack. Precondition: $|s| \geq 2$.
- `SHIFT`: moves b_1 from the buffer to the stack. Precondition: $|b| \geq 1$.

In the labeled version of parsing, there are in total $|\mathcal{T}| = 2N_l + 1$ transitions, where N_l is number of different arc labels. Figure 1 illustrates an example of one transition sequence from the initial configuration to a terminal one.

The essential goal of a greedy parser is to predict a correct transition from $\mathcal{T}$, based on one given configuration. Information that can be obtained from one configuration includes: (1) all the words and their corresponding POS tags (e.g., `has` / `VBZ`); (2) the head of a word and its label (e.g., `nsubj`, `dobj`) if applicable; (3) the position of a word on the stack/buffer or whether it has already been removed from the stack.

Single-word features (9)	
$s_1.w$; $s_1.t$; $s_1.wt$; $s_2.w$; $s_2.t$;	
$s_2.wt$; $b_1.w$; $b_1.t$; $b_1.wt$	
Word-pair features (8)	
$s_1.wt \circ s_2.wt$; $s_1.wt \circ s_2.w$; $s_1.wts_2.t$;	
$s_1.w \circ s_2.wt$; $s_1.t \circ s_2.wt$; $s_1.w \circ s_2.w$	
$s_1.t \circ s_2.t$; $s_1.t \circ b_1.t$	
Three-word feaures (8)	
$s_2.t \circ s_1.t \circ b_1.t$; $s_2.t \circ s_1.t \circ lc_1(s_1).t$;	
$s_2.t \circ s_1.t \circ rc_1(s_1).t$; $s_2.t \circ s_1.t \circ lc_1(s_2).t$;	
$s_2.t \circ s_1.t \circ rc_1(s_2).t$; $s_2.t \circ s_1.w \circ rc_1(s_2).t$;	
$s_2.t \circ s_1.w \circ lc_1(s_1).t$; $s_2.t \circ s_1.w \circ b_1.t$	

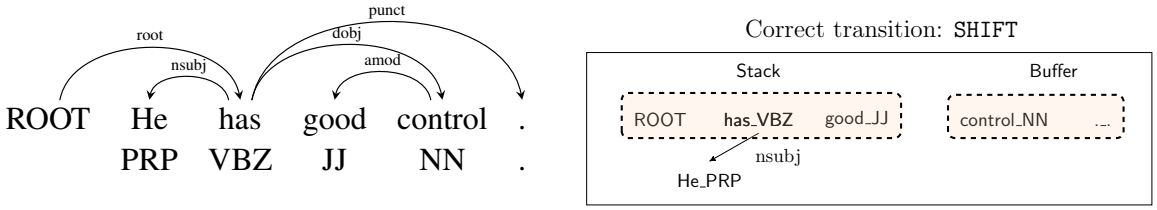
Table 1: The feature templates used for analysis. $lc_1(s_i)$ and $rc_1(s_i)$ denote the leftmost and rightmost children of s_i, w denotes word, t denotes POS tag.

Conventional approaches extract indicator features such as the conjunction of $1 \sim 3$ elements from the stack/buffer using their words, POS tags or arc labels. Table 1 lists a typical set of feature templates chosen from the ones of (Huang et al., 2009; Zhang and Nivre, 2011).[2] These features suffer from the following problems:

- **Sparsity**. The features, especially lexicalized features are highly sparse, and this is a common problem in many NLP tasks. The situation is severe in dependency parsing, because it depends critically on word-to-word interactions and thus the high-order features. To give a better understanding, we perform a feature analysis using the features in Table 1 on the English Penn Treebank (CoNLL representations). The results given in Table 2 demonstrate that: (1) lexicalized features are indispensable; (2) Not only are the word-pair features (especially s_1 and s_2) vital for predictions, the three-word conjunctions (e.g., $\{s_2, s_1, b_1\}$, $\{s_2, lc_1(s_1), s_1\}$) are also very important.

[1] Additionally, our parser can be naturally incorporated with beam search, but we leave this to future work.

[2] We exclude sophisticated features using labels, distance, valency and third-order features in this analysis, but we will include all of them in the final evaluation.



Transition	Stack	Buffer	A
	[ROOT]	[He has good control .]	$\emptyset$
SHIFT	[ROOT He]	[has good control .]	
SHIFT	[ROOT He has]	[good control .]	
LEFT-ARC(nsubj)	[ROOT has]	[good control .]	$A\cup$ nsubj(has,He)
SHIFT	[ROOT has good]	[control .]	
SHIFT	[ROOT has good control]	[.]	
LEFT-ARC(amod)	[ROOT has control]	[.]	$A\cup$amod(control,good)
RIGHT-ARC(dobj)	[ROOT has]	[.]	$A\cup$ dobj(has,control)
...	...	...	...
RIGHT-ARC(root)	[ROOT]	[]	$A\cup$ root(ROOT,has)

Figure 1: An example of transition-based dependency parsing. Above left: a desired dependency tree, above right: an intermediate configuration, bottom: a transition sequence of the arc-standard system.

Features	UAS
All features in Table 1	88.0
single-word & word-pair features	82.7
only single-word features	76.9
excluding all lexicalized features	81.5

Table 2: Performance of different feature sets. UAS: unlabeled attachment score.

- **Incompleteness**. Incompleteness is an unavoidable issue in all existing feature templates. Because even with expertise and manual handling involved, they still do not include the conjunction of every useful word combination. For example, the conjunction of s_1 and b_2 is omitted in almost all commonly used feature templates, however it could indicate that we cannot perform a RIGHT-ARC action if there is an arc from s_1 to b_2.
- **Expensive feature computation**. The feature generation of indicator features is generally expensive — we have to concatenate some words, POS tags, or arc labels for generating feature strings, and look them up in a huge table containing several millions of features. In our experiments, more than 95% of the time is consumed by feature computation during the parsing process.

So far, we have discussed preliminaries of transition-based dependency parsing and existing problems of sparse indicator features. In the following sections, we will elaborate our neural network model for learning dense features along with experimental evaluations that prove its efficiency.

3 Neural Network Based Parser

In this section, we first present our neural network model and its main components. Later, we give details of training and speedup of parsing process.

3.1 Model

Figure 2 describes our neural network architecture. First, as usual word embeddings, we represent each word as a d-dimensional vector $e_i^w \in \mathbb{R}^d$ and the full embedding matrix is $E^w \in \mathbb{R}^{d\times N_w}$ where N_w is the dictionary size. Meanwhile, we also map POS tags and arc labels to a d-dimensional vector space, where $e_i^t, e_j^l \in \mathbb{R}^d$ are the representations of i^{th} POS tag and j^{th} arc label. Correspondingly, the POS and label embedding matrices are $E^t \in \mathbb{R}^{d\times N_t}$ and $E^l \in \mathbb{R}^{d\times N_l}$ where N_t and N_l are the number of distinct POS tags and arc labels.

We choose a set of elements based on the stack / buffer positions for each type of information (word, POS or label), which might be useful for our predictions. We denote the sets as S^w, S^t, S^l respectively. For example, given the configuration in Figure 2 and $S^t =$

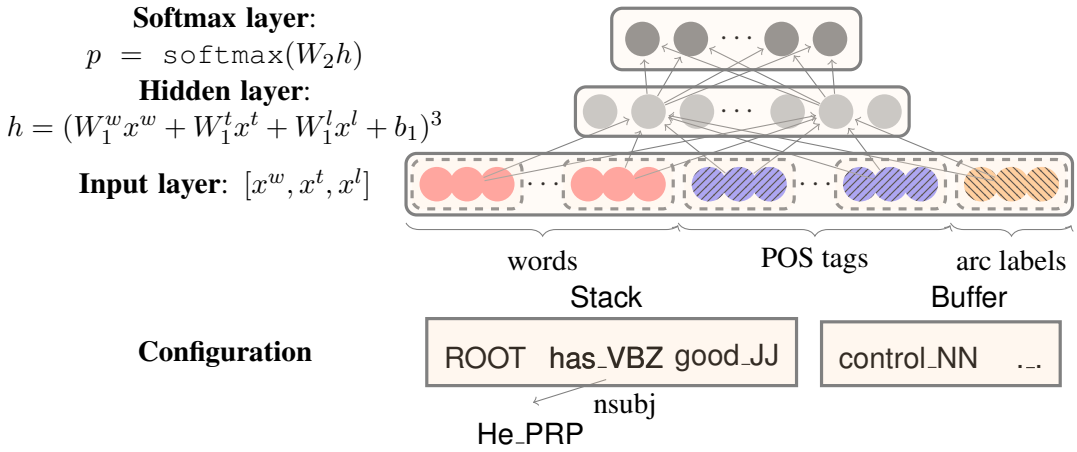


Figure 2: Our neural network architecture.

$\{lc_1(s_2).t, s_2.t, rc_1(s_2).t, s_1.t\}$, we will extract PRP, VBZ, NULL, JJ in order. Here we use a special token NULL to represent a non-existent element.

We build a standard neural network with one hidden layer, where the corresponding embeddings of our chosen elements from S^w, S^t, S^l will be added to the input layer. Denoting n_w, n_t, n_l as the number of chosen elements of each type, we add $x^w = [e^w_{w_1}; e^w_{w_2}; \dots e^w_{w_{n_w}}]$ to the input layer, where $S^w = \{w_1, \dots, w_{n_w}\}$. Similarly, we add the POS tag features x^t and arc label features x^l to the input layer.

We map the input layer to a hidden layer with d_h nodes through a **cube activation function**:

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

where $W_1^w \in \mathbb{R}^{d_h \times (d \cdot n_w)}$, $W_1^t \in \mathbb{R}^{d_h \times (d \cdot n_t)}$, $W_1^l \in \mathbb{R}^{d_h \times (d \cdot n_l)}$, and $b_1 \in \mathbb{R}^{d_h}$ is the bias.

A softmax layer is finally added on the top of the hidden layer for modeling multi-class probabilities $p = \texttt{softmax}(W_2 h)$, where $W_2 \in \mathbb{R}^{|\mathcal{T}| \times d_h}$.

POS and label embeddings

To our best knowledge, this is the first attempt to introduce POS tag and arc label embeddings instead of discrete representations.

Although the POS tags $\mathcal{P}$ = {NN, NNP, NNS, DT, JJ, ...} (for English) and arc labels $\mathcal{L}$ = {amod, tmod, nsubj, csubj, dobj, ...} (for Stanford Dependencies on English) are relatively small discrete sets, they still exhibit many semantical similarities like words. For example, NN (singular noun) should be closer to NNS (plural noun) than DT (determiner), and amod (adjective modifier) should be closer to num (numeric modifier) than nsubj (nominal subject). We expect these semantic meanings to be effectively captured by the dense representations.

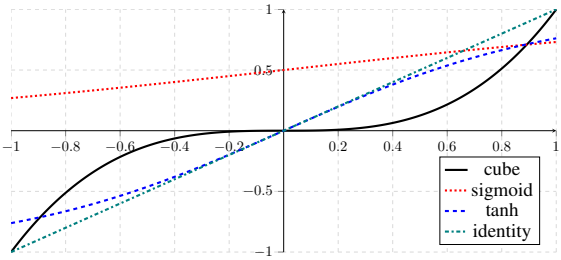


Figure 3: Different activation functions used in neural networks.

Cube activation function

As stated above, we introduce a novel activation function: cube $g(x) = x^3$ in our model instead of the commonly used tanh or sigmoid functions (Figure 3).

Intuitively, every hidden unit is computed by a (non-linear) mapping on a weighted sum of input units plus a bias. Using $g(x) = x^3$ can model the product terms of $x_i x_j x_k$ for any three different elements at the input layer directly:

$$g(w_1 x_1 + \dots + w_m x_m + b) = \sum_{i,j,k} (w_i w_j w_k) x_i x_j x_k + \sum_{i,j} b(w_i w_j) x_i x_j \dots$$

In our case, x_i, x_j, x_k could come from different dimensions of three embeddings. We believe that this better captures the interaction of three ele-

ments, which is a very desired property of dependency parsing.

Experimental results also verify the success of the cube activation function empirically (see more comparisons in Section 4). However, the expressive power of this activation function is still open to investigate theoretically.

The choice of S^w, S^t, S^l

Following (Zhang and Nivre, 2011), we pick a rich set of elements for our final parser. In detail, S^w contains $n_w = 18$ elements: (1) The top 3 words on the stack and buffer: $s_1, s_2, s_3, b_1, b_2, b_3$; (2) The first and second leftmost / rightmost children of the top two words on the stack: $lc_1(s_i), rc_1(s_i), lc_2(s_i), rc_2(s_i),\ i = 1, 2$. (3) The leftmost of leftmost / rightmost of rightmost children of the top two words on the stack: $lc_1(lc_1(s_i)), rc_1(rc_1(s_i)), i = 1, 2$.

We use the corresponding POS tags for S^t ($n_t = 18$), and the corresponding arc labels of words excluding those 6 words on the stack/buffer for S_l ($n_l = 12$). A good advantage of our parser is that we can add a rich set of elements cheaply, instead of hand-crafting many more indicator features.

3.2 Training

We first generate training examples $\{(c_i, t_i)\}_{i=1}^m$ from the training sentences and their gold parse trees using a "shortest stack" oracle which always prefers $\texttt{LEFT-ARC}_l$ over $\texttt{SHIFT}$, where c_i is a configuration, $t_i \in \mathcal{T}$ is the oracle transition.

The final training objective is to minimize the cross-entropy loss, plus a l_2-regularization term:

$$L(\theta) = -\sum_i \log p_{t_i} + \frac{\lambda}{2}\|\theta\|^2$$

where θ is the set of all parameters $\{W_1^w, W_1^t, W_1^l, b_1, W_2, E^w, E^t, E^l\}$. A slight variation is that we compute the softmax probabilities only among the feasible transitions in practice.

For initialization of parameters, we use pre-trained word embeddings to initialize E^w and use random initialization within $(-0.01, 0.01)$ for E^t and E^l. Concretely, we use the pre-trained word embeddings from (Collobert et al., 2011) for English (#dictionary = 130,000, coverage = 72.7%), and our trained 50-dimensional `word2vec` embeddings (Mikolov et al., 2013) on Wikipedia and Gigaword corpus for Chinese (#dictionary = 285,791, coverage = 79.0%). We will also compare with random initialization of E^w in Section 4. The training error derivatives will be back-propagated to these embeddings during the training process.

We use mini-batched AdaGrad (Duchi et al., 2011) for optimization and also apply a dropout (Hinton et al., 2012) with 0.5 rate. The parameters which achieve the best unlabeled attachment score on the development set will be chosen for final evaluation.

3.3 Parsing

We perform greedy decoding in parsing. At each step, we extract all the corresponding word, POS and label embeddings from the current configuration c, compute the hidden layer $h(c) \in \mathbb{R}^{d_h}$, and pick the transition with the highest score: $t = \arg\max_{t \text{ is feasible}} W_2(t, \cdot)h(c)$, and then execute $c \rightarrow t(c)$.

Comparing with indicator features, our parser does not need to compute conjunction features and look them up in a huge feature table, and thus greatly reduces feature generation time. Instead, it involves many matrix addition and multiplication operations. To further speed up the parsing time, we apply a **pre-computation** trick, similar to (Devlin et al., 2014). For each position chosen from S^w, we pre-compute matrix multiplications for most top frequent $10,000$ words. Thus, computing the hidden layer only requires looking up the table for these frequent words, and adding the d_h-dimensional vector. Similarly, we also pre-compute matrix computations for all positions and all POS tags and arc labels. We only use this optimization in the neural network parser, but it is only feasible for a parser like the neural network parser which uses a small number of features. In practice, this pre-computation step increases the speed of our parser $8 \sim 10$ times.

4 Experiments

4.1 Datasets

We conduct our experiments on the English Penn Treebank (PTB) and the Chinese Penn Treebank (CTB) datasets.

For English, we follow the standard splits of `PTB3`, using sections 2-21 for training, section 22 as development set and 23 as test set. We adopt two different dependency representations: CoNLL Syntactic Dependencies (CD) (Johansson

Dataset	#Train	#Dev	#Test	#words (N_w)	#POS (N_t)	#labels (N_l)	projective (%)
PTB: CD	39,832	1,700	2,416	44,352	45	17	99.4
PTB: SD	39,832	1,700	2,416	44,389	45	45	99.9
CTB	16,091	803	1,910	34,577	35	12	100.0

Table 3: Data Statistics. "Projective" is the percentage of projective trees on the training set.

and Nugues, 2007) using the LTH Constituent-to-Dependency Conversion Tool[3] and Stanford Basic Dependencies (SD) (de Marneffe et al., 2006) using the Stanford parser v3.3.0.[4] The POS tags are assigned using Stanford POS tagger (Toutanova et al., 2003) with ten-way jackknifing of the training data (accuracy $\approx 97.3\%$).

For Chinese, we adopt the same split of `CTB5` as described in (Zhang and Clark, 2008). Dependencies are converted using the Penn2Malt tool[5] with the head-finding rules of (Zhang and Clark, 2008). And following (Zhang and Clark, 2008; Zhang and Nivre, 2011), we use gold segmentation and POS tags for the input.

Table 3 gives statistics of the three datasets.[6] In particular, over 99% of the trees are projective in all datasets.

4.2 Results

The following hyper-parameters are used in all experiments: embedding size $d = 50$, hidden layer size $h = 200$, regularization parameter $\lambda = 10^{-8}$, initial learning rate of Adagrad $\alpha = 0.01$.

To situate the performance of our parser, we first make a comparison with our own implementation of greedy arc-eager and arc-standard parsers. These parsers are trained with structured averaged perceptron using the "early-update" strategy. The feature templates of (Zhang and Nivre, 2011) are used for the arc-eager system, and they are also adapted to the arc-standard system.[7]

Furthermore, we also compare our parser with two popular, off-the-shelf parsers: MaltParser — a greedy transition-based dependency parser (Nivre et al., 2006),[8] and MSTParser — a first-order graph-based parser (McDonald and Pereira, 2006).[9] In this comparison, for MaltParser, we select `stackproj` (arc-standard) and `nivreeager` (arc-eager) as parsing algorithms, and liblinear (Fan et al., 2008) for optimization.[10] For MSTParser, we use default options.

On all datasets, we report unlabeled attachment scores (UAS) and labeled attachment scores (LAS) and punctuation is excluded in all evaluation metrics.[11] Our parser and the baseline arc-standard and arc-eager parsers are all implemented in Java. The parsing speeds are measured on an Intel Core i7 2.7GHz CPU with 16GB RAM and the runtime does not include pre-computation or parameter loading time.

Table 4, Table 5 and Table 6 show the comparison of accuracy and parsing speed on PTB (CoNLL dependencies), PTB (Stanford dependencies) and CTB respectively.

Parser	Dev		Test		Speed
	UAS	LAS	UAS	LAS	(sent/s)
standard	89.9	88.7	89.7	88.3	51
eager	90.3	89.2	89.9	88.6	63
Malt:sp	90.0	88.8	89.9	88.5	560
Malt:eager	90.1	88.9	90.1	88.7	535
MSTParser	92.1	90.8	**92.0**	90.5	12
Our parser	**92.2**	**91.0**	**92.0**	**90.7**	**1013**

Table 4: Accuracy and parsing speed on PTB + CoNLL dependencies.

Clearly, our parser is superior in terms of both accuracy and speed. Comparing with the baselines of arc-eager and arc-standard parsers, our parser achieves around 2% improvement in UAS and LAS on all datasets, while running about 20 times faster.

It is worth noting that the efficiency of our

[3] http://nlp.cs.lth.se/software/treebank_converter/

[4] http://nlp.stanford.edu/software/lex-parser.shtml

[5] http://stp.lingfil.uu.se/ nivre/research/Penn2Malt.html

[6] Pennconverter and Stanford dependencies generate slightly different tokenization, e.g., Pennconverter splits the token `WCRS\/Boston_NNP` into three tokens `WCRS_NNP / _CC Boston_NNP`.

[7] Since arc-standard is bottom-up, we remove all features using the head of stack elements, and also add the right child features of the first stack element.

[8] http://www.maltparser.org/

[9] http://www.seas.upenn.edu/ strctlrn/MSTParser/MSTParser.html

[10] We do not compare with libsvm optimization, which is known to be sightly more accurate, but orders of magnitude slower (Kong and Smith, 2014).

[11] A token is a punctuation if its gold POS tag is {" " : , .} for English and `PU` for Chinese.

Parser	Dev		Test		Speed
	UAS	LAS	UAS	LAS	(sent/s)
standard	90.2	87.8	89.4	87.3	26
eager	89.8	87.4	89.6	87.4	34
Malt:sp	89.8	87.2	89.3	86.9	469
Malt:eager	89.6	86.9	89.4	86.8	448
MSTParser	91.4	88.1	90.7	87.6	10
Our parser	**92.0**	**89.7**	**91.8**	**89.6**	**654**

Table 5: Accuracy and parsing speed on PTB + Stanford dependencies.

Parser	Dev		Test		Speed
	UAS	LAS	UAS	LAS	(sent/s)
standard	82.4	80.9	82.7	81.2	72
eager	81.1	79.7	80.3	78.7	80
Malt:sp	82.4	80.5	82.4	80.6	420
Malt:eager	81.2	79.3	80.2	78.4	393
MSTParser	**84.0**	82.1	83.0	81.2	6
Our parser	**84.0**	**82.4**	**83.9**	**82.4**	**936**

Table 6: Accuracy and parsing speed on CTB.

parser even surpasses MaltParser using liblinear, which is known to be highly optimized, while our parser achieves much better accuracy.

Also, despite the fact that the graph-based MSTParser achieves a similar result to ours on PTB (CoNLL dependencies), our parser is nearly 100 times faster. In particular, our transition-based parser has a great advantage in LAS, especially for the fine-grained label set of Stanford dependencies.

4.3 Effects of Parser Components

Herein, we examine components that account for the performance of our parser.

Cube activation function

We compare our `cube` activation function (x^3) with two widely used non-linear functions: `tanh` ($\frac{e^x-e^{-x}}{e^x+e^{-x}}$), `sigmoid` ($\frac{1}{1+e^{-x}}$), and also the `identity` function (x), as shown in Figure 4 (left).

In short, `cube` outperforms all other activation functions significantly and `identity` works the worst. Concretely, `cube` can achieve $0.8\% \sim 1.2\%$ improvement in UAS over `tanh` and other functions, thus verifying the effectiveness of the cube activation function empirically.

Initialization of pre-trained word embeddings

We further analyze the influence of using pre-trained word embeddings for initialization. Figure 4 (middle) shows that using pre-trained word embeddings can obtain around 0.7% improvement on PTB and 1.7% improvement on CTB, compared with using random initialization within $(-0.01, 0.01)$. On the one hand, the pre-trained word embeddings of Chinese appear more useful than those of English; on the other hand, our model is still able to achieve comparable accuracy without the help of pre-trained word embeddings.

POS tag and arc label embeddings

As shown in Figure 4 (right), POS embeddings yield around 1.7% improvement on PTB and nearly 10% improvement on CTB and the label embeddings yield a much smaller 0.3% and 1.4% improvement respectively.

However, we can obtain little gain from label embeddings when the POS embeddings are present. This may be because the POS tags of two tokens already capture most of the label information between them.

4.4 Model Analysis

Last but not least, we will examine the parameters we have learned, and hope to investigate what these dense features capture. We use the weights learned from the English Penn Treebank using Stanford dependencies for analysis.

What do E^t, E^l capture?

We first introduced E^t and E^l as the dense representations of all POS tags and arc labels, and we wonder whether these embeddings could carry some semantic information.

Figure 5 presents t-SNE visualizations (van der Maaten and Hinton, 2008) of these embeddings. It clearly shows that these embeddings effectively exhibit the similarities between POS tags or arc labels. For instance, the three adjective POS tags `JJ`, `JJR`, `JJS` have very close embeddings, and also the three labels representing clausal complements `acomp`, `ccomp`, `xcomp` are grouped together.

Since these embeddings can effectively encode the semantic regularities, we believe that they can be also used as alternative features of POS tags (or arc labels) in other NLP tasks, and help boost the performance.

What do W_1^w, W_1^t, W_1^l capture?

Knowing that E^t and E^l (as well as the word embeddings E^w) can capture semantic information very well, next we hope to investigate what each feature in the hidden layer has really learned.

Since we currently only have $h = 200$ learned dense features, we wonder if it is sufficient to learn the word conjunctions as sparse indicator features, or even more. We examine the weights $W_1^w(k, \cdot) \in \mathbb{R}^{d \cdot n_w}$, $W_1^t(k, \cdot) \in \mathbb{R}^{d \cdot n_t}$, $W_1^l(k, \cdot) \in \mathbb{R}^{d \cdot n_l}$ for each hidden unit k, and reshape them to $d \times n_t$, $d \times n_w$, $d \times n_l$ matrices, such that the weights of each column corresponds to the embeddings of one specific element (e.g., $s_1.t$).

We pick the weights with absolute value > 0.2, and visualize them for each feature. Figure 6 gives the visualization of three sampled features, and it exhibits many interesting phenomena:

- Different features have varied distributions of the weights. However, most of the discriminative weights come from W_1^t (the middle zone in Figure 6), and this further justifies the importance of POS tags in dependency parsing.
- We carefully examine many of the $h = 200$ features, and find that they actually encode very different views of information. For the three sampled features in Figure 6, the largest weights are dominated by:
  - Feature 1: $s_1.t, s_2.t, lc(s_1).t.$
  - Feautre 2: $rc(s_1).t, s_1.t, b_1.t.$
  - Feature 3: $s_1.t, s_1.w, lc(s_1).t, lc(s_1).l.$

  These features all seem very plausible, as observed in the experiments on indicator feature systems. Thus our model is able to automatically identify the most useful information for predictions, instead of hand-crafting them as indicator features.
- More importantly, we can extract features regarding the conjunctions of more than 3 elements easily, and also those not presented in the indicator feature systems. For example, the 3rd feature above captures the conjunction of words and POS tags of s_1, the tag of its leftmost child, and also the label between them, while this information is not encoded in the original feature templates of (Zhang and Nivre, 2011).

5 Related Work

There have been several lines of earlier work in using neural networks for parsing which have points of overlap but also major differences from our work here. One big difference is that much early work uses localist one-hot word representations rather than the distributed representations of modern work. (Mayberry III and Miikkulainen, 1999) explored a shift reduce constituency parser with one-hot word representations and did subsequent parsing work in (Mayberry III and Miikkulainen, 2005).

(Henderson, 2004) was the first to attempt to use neural networks in a broad-coverage Penn Treebank parser, using a simple synchrony network to predict parse decisions in a constituency parser. More recently, (Titov and Henderson, 2007) applied Incremental Sigmoid Belief Networks to constituency parsing and then (Garg and Henderson, 2011) extended this work to transition-based dependency parsers using a Temporal Restricted Boltzman Machine. These are very different neural network architectures, and are much less scalable and in practice a restricted vocabulary was used to make the architecture practical.

There have been a number of recent uses of deep learning for constituency parsing (Collobert, 2011; Socher et al., 2013). (Socher et al., 2014) has also built models over dependency representations but this work has not attempted to learn neural networks for dependency parsing.

Most recently, (Stenetorp, 2013) attempted to build recursive neural networks for transition-based dependency parsing, however the empirical performance of his model is still unsatisfactory.

6 Conclusion

We have presented a novel dependency parser using neural networks. Experimental evaluations show that our parser outperforms other greedy parsers using sparse indicator features in both accuracy and speed. This is achieved by representing all words, POS tags and arc labels as dense vectors, and modeling their interactions through a novel cube activation function. Our model only relies on dense features, and is able to automatically learn the most useful feature conjunctions for making predictions.

An interesting line of future work is to combine our neural network based classifier with search-based models to further improve accuracy. Also,

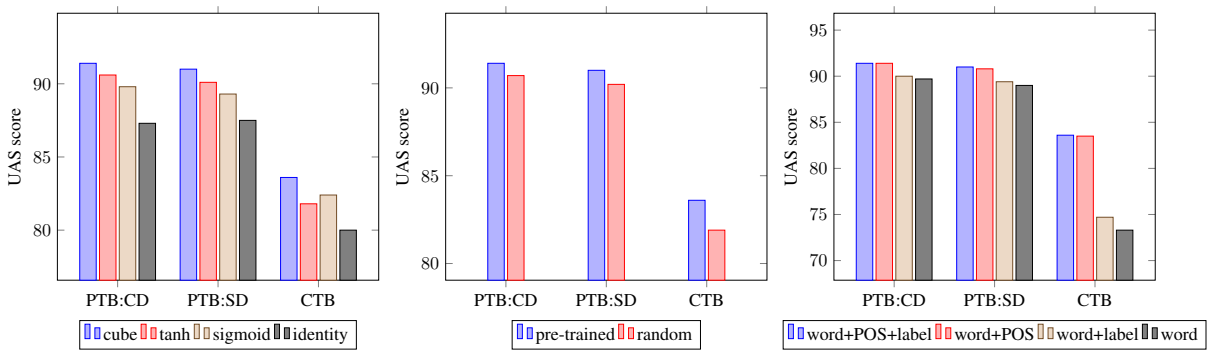


Figure 4: Effects of different parser components. Left: comparison of different activation functions. Middle: comparison of pre-trained word vectors and random initialization. Right: effects of POS and label embeddings.

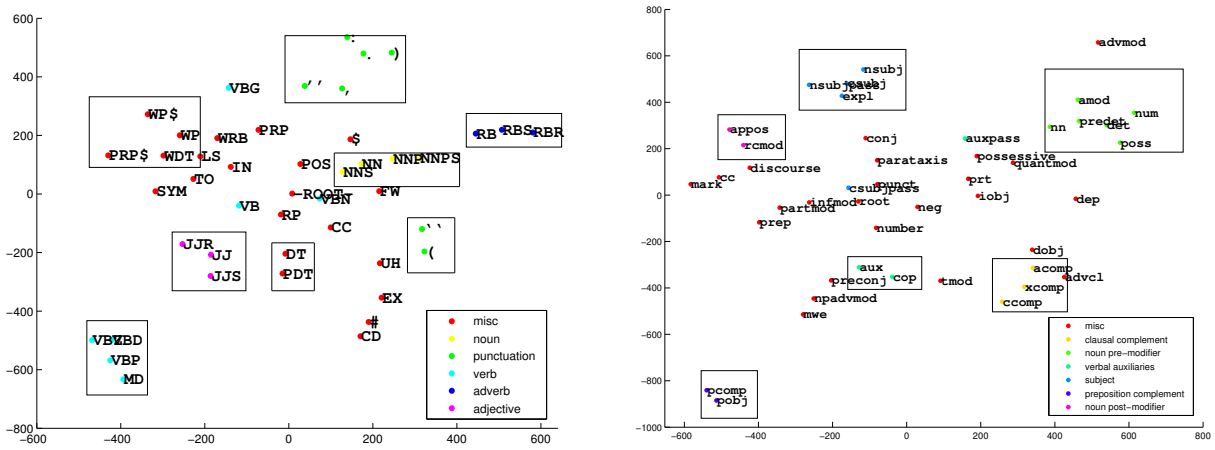


Figure 5: t-SNE visualization of POS and label embeddings.

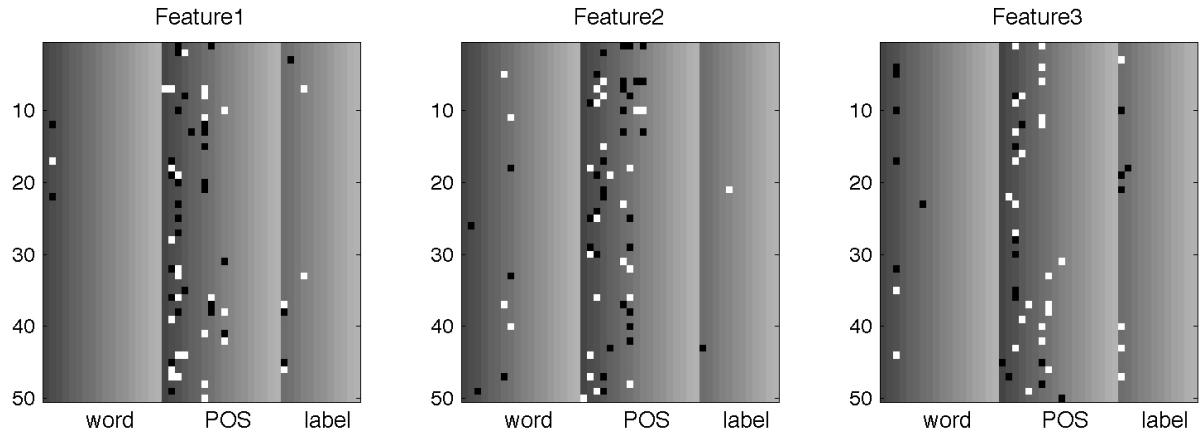


Figure 6: Three sampled features. In each feature, each row denotes a dimension of embeddings and each column denotes a chosen element, e.g., $s_1.t$ or $lc(s_1).w$, and the parameters are divided into 3 zones, corresponding to $W_1^w(k,:)$ (left), $W_1^t(k,:)$ (middle) and $W_1^l(k,:)$ (right). White and black dots denote the most positive weights and most negative weights respectively.

there is still room for improvement in our architecture, such as better capturing word conjunctions, or adding richer features (e.g., distance, valency).

Acknowledgments

Stanford University gratefully acknowledges the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040 and the Defense Threat Reduction Agency (DTRA) under Air Force Research Laboratory (AFRL) contract no. FA8650-10-C-7020. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Coling*.

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

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *AISTATS*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *ACL*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*.

Nikhil Garg and James Henderson. 2011. Temporal restricted boltzmann machines for dependency parsing. In *ACL-HLT*.

He He, Hal Daumé III, and Jason Eisner. 2013. Dynamic feature selection for dependency parsing. In *EMNLP*.

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

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

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *EMNLP*.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for english. In *Proceedings of NODALIDA*, Tartu, Estonia.

Lingpeng Kong and Noah A. Smith. 2014. An empirical comparison of parsing methods for Stanford dependencies. *CoRR*, abs/1404.4314.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *ACL*.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool.

Marshall R. Mayberry III and Risto Miikkulainen. 1999. Sardsrn: A neural network shift-reduce parser. In *IJCAI*.

Marshall R. Mayberry III and Risto Miikkulainen. 2005. Broad-coverage parsing with neural networks. *Neural Processing Letters*.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

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

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *LREC*.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *ACL*.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *TACL*.

Pontus Stenetorp. 2013. Transition-based dependency parsing using recursive neural networks. In *NIPS Workshop on Deep Learning*.

Ivan Titov and James Henderson. 2007. Fast and robust multilingual dependency parsing with a generative latent variable model. In *EMNLP-CoNLL*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *The Journal of Machine Learning Research*.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing using beam-search. In *EMNLP*.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *ACL*.

Why are You Taking this Stance? Identifying and Classifying Reasons in Ideological Debates

Kazi Saidul Hasan and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{saidul,vince}@hlt.utdallas.edu

Abstract

Recent years have seen a surge of interest in stance classification in online debates. Oftentimes, however, it is important to determine not only the stance expressed by an author in her debate posts, but also the reasons behind her supporting or opposing the issue under debate. We therefore examine the new task of reason classification in this paper. Given the close interplay between stance classification and reason classification, we design computational models for examining how automatically computed stance information can be profitably exploited for reason classification. Experiments on our reason-annotated corpus of ideological debate posts from four domains demonstrate that sophisticated models of stances and reasons can indeed yield more accurate reason and stance classification results than their simpler counterparts.

1 Introduction

In recent years, researchers have begun exploring new opinion mining tasks. One such task is *debate stance classification* (SC): given a post written for a *two-sided* topic discussed in an online debate forum, determine which of the two sides (i.e., *for* or *against*) its author is taking (Agrawal et al., 2003; Thomas et al., 2006; Bansal et al., 2008; Somasundaran and Wiebe, 2009; Burfoot et al., 2011; Hasan and Ng, 2013b). For example, the author of the post shown in Figure 1 is pro-abortion.

Oftentimes, however, it is important to determine not only the author's stance expressed in her debate posts, but also the reasons why she supports or opposes the issue under debate. Intuitively, given a debate topic such as *"Should abortion be banned?"* or *"Do you support Obamacare?"*, it should not be difficult for us to come up with a set of reasons people typically use to back up their stances. Given a set of reasons associated with each stance in an online debate, the goal of *post-level reason classification* is to identify those reason(s) an author uses to back up her stance in her debate post. A more challenging version of this task is *sentence-level reason classification*, where the goal is to identify not only the reason(s) an author uses in her post, but also the sentence(s) in the post that the author uses to describe each of her reasons. For example, the author of the post shown in Figure 1 mentions two reasons why she supports abortion, namely *it's a woman's right to abort* and *unwanted babies are threat to their parents' future*, which are mentioned in the first and third sentences in the post respectively.

> [I feel that abortion should remain legal, or rather, parents should have the power to make the decision themselves and not face any legal hindrance of any form.][1] Let us take a look from the social perspective. [If parents cannot afford to provide for the child, or if the family is facing financial constraints, it is understandable that abortion can remain as one of the options.][2]
>
> **Reason 1:** Woman's right to abort
> **Reason 2:** Unwanted babies are threat to their parents' future

Figure 1: A sample post on abortion annotated with reasons.

Our goal in this paper is to examine post- and sentence-level reason classification (RC) in *ideological debates*. Many online debaters use emotional languages, which may involve sarcasm and insults, to express their points, thereby making RC and SC in ideological debates potentially more challenging than that in other debate settings such as congressional debates and company-internal discussions (Walker et al., 2012).

Besides examining the new task of RC in ideological debates, we believe that our work makes three contributions. First, we propose to address post-level RC by means of sentence-level RC by

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 751–762,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

(1) determining the reason(s) associated with each of its sentences (if any), and then (2) taking the union of the set of reasons associated with all of its sentences to be the set of reasons associated with the post. We hypothesize that this sentence-based approach, which exploits a training set in which each sentence in a post is labeled with its reason, would achieve better performance than a *multi-label* text classification approach to post-level RC, which learns to determine the subset of reasons a post contains directly from a training set in which each post is labeled with the corresponding set of reasons. In other words, we hypothesize that we could achieve better results for post-level RC by learning from *sentence-level* than from *post-level* reason annotations, as sentence-level reason annotations can enable a learning algorithm to accurately attribute an annotated reason to a particular portion of a post.

Second, we propose *stance-supported* RC systems, hypothesizing that *automatically computed* stance information can be profitably exploited for RC. Since we are exploiting automatically computed (and thus potentially noisy) stance information, we hypothesize that the effectiveness of such information would depend in part on the way it is exploited in RC systems. As a result, we introduce a set of stance-supported models for RC, starting with simple pipeline models and then moving on to joint models with increasing sophistication. Note that exploiting stance information by no means guarantees that RC performance will improve, as an incorrect determination of stance could lead to an incorrect identification of reasons. Hence, one of our goals is to examine how to model stances and reasons so that RC can benefit from stance information.

Finally, since progress on RC is hindered in part by the lack of an annotated corpus, we make our reason-annotated dataset publicly available.[1] To our knowledge, this will be the first publicly available corpus for sentence- and post-level RC.

2 Corpus and Annotation

We collected debate posts from four popular *domains*, Abortion (ABO), Gay Rights (GAY), Obama (OBA), and Marijuana (MAR), from an online debate forum[2]. All debates are two-sided, so each post receives one of two *stance labels*, *for* or *against*, depending on whether the author of the post *supports* or *opposes* abortion, gay rights, Obama, or the legalization of marijuana respectively. A post's stance label is given by its author.

Note that each post belongs to a *thread*, which is a tree with one or more nodes such that (1) each node corresponds to a debate post, and (2) a post y_i is the parent of another post y_j if y_j is a reply to y_i. Given a thread, we generate *post sequences*, each of which is a path from the root of the thread to one of its leaves. Hence, a post sequence is an ordered set of posts such that each post is a reply to its immediately preceding post in the sequence. Table 2a shows the statistics of the four stance-labeled datasets.

While the debate posts contain the stance labels given by their authors, they are not annotated with reasons. As part of our study of RC, we annotate each post with the reasons it gives for its stance. Our annotation procedure is composed of three steps. First, two human annotators independently examined each post and identified the reasons authors present to support their stances (i.e., *for* and *against*) in each domain. Second, they discussed and agreed on the reasons identified for each domain. Third, they independently annotated the text of each post with reason labels from the post's domain. To do this, they labeled each sentence of a post with the set of reasons the author expressed in that sentence. Any sentence that does not belong to any reason class was assigned the NONE class.

After the annotators completed the aforementioned steps, they were asked to collapse all the reason classes that occur in less than 2% of the sentences annotated with non-NONE classes into the OTHER class. In other words, all the sentences that were originally annotated with one of these infrequent reason classes will now be labeled as OTHER. Our decision to merge infrequent classes is motivated by two observations. First, from a practical point of view, infrequent reasons do not carry much weight. Second, from a modeling perspective, it is often not worth increasing model complexity by handling infrequent classes. The resulting set of reason classes for each domain is shown in Table 1.

A closer examination of the resulting annotations reveals that approximately 3% of the sentences received multiple reason labels. Again, to avoid the complexity of modeling multi-labeled

[1] http://www.hlt.utdallas.edu/~saidul/stance/

[2] http://www.createdebate.com/

Domain	Stance	Reason classes
ABO	*for*	[F1] Abortion is a woman's right (26%); [F2] Rape victims need it to be legal (7%); [F3] A fetus is not human (38%); [F4] Mother's life in danger (5%); [F5] Unwanted babies are ill-treated by parents (8%); [F6] Birth control fails at times (3%); [F7] Abortion is not murder (3%); [F8] Mother is not healthy/financially solvent (4%); [F9] Others (6%)
	against	[A1] Put baby up for adoption (9%); [A2] Abortion kills a life (29%); [A3] An unborn baby is a human and has the right to live (40%); [A4] Be willing to have the baby if you have sex (14%); [A5] Abortion is harmful for women (5%); [A6] Others (3%)
GAY	*for*	[F1] Gay marriage is like any other marriage (14%); [F2] Gay people should have the same rights as straight people (36%); [F3] Gay parents can adopt and ensure a happy life for a baby (10%); [F4] People are born gay (18%); [F5] Religion should not be used against gay rights (11%); [F6] Others (11%)
	against	[A1] Religion does not permit gay marriages (18%); [A2] Gay marriages are not normal/against nature (39%); [A3] Gay parents can not raise kids properly (11%); [A4] Gay people have problems and create social issues (16%); [A5] Others (16%)
OBA	*for*	[F1] Fixed the economy (21%); [F2] Ending the wars (7%); [F3] Better than the republican candidates (25%); [F4] Makes good decisions/policies (8%); [F5] Has qualities of a good leader (14%); [F6] Ensured better healthcare (8%); [F7] Executed effective foreign policies (6%); [F8] Created more jobs (4%); [F9] Others (7%)
	against	[A1] Destroyed our economy (26%); [A2] Wars are still on (11%); [A3] Unemployment rate is high (5%); [A4] Healthcare bill is a failure(9%); [A5] Poor decision-maker (7%); [A6] We have better republicans than Obama (5%); [A7] Not eligible as a leader (20%); [A8] Ineffective foreign policies (4%); [A9] Others (13%)
MAR	*for*	[F1] Not addictive (23%); [F2] Used as a medicine (11%); [F3] Legalized marijuana can be controlled and regulated by the government (33%); [F4] Prohibition violates human rights (15%); [F5] Does not cause any damage to our bodies (6%); [F6] Others (12%)
	against	[A1] Damages our bodies (23%); [A2] Responsible for brain damage (22%); [A3] If legalized, people will use marijuana and other drugs more (12%); [A4] Causes crime (9%); [A5] Highly addictive (17%); [A6] Others (17%)

Table 1: Reason classes and their percentages in the corresponding stance for each domain.

sentences given their rarity, we asked each annotator to pick the reason that was highlighted the most in each multi-labeled sentence.

Inter-annotator agreement scores at the sentence level and the post level, expressed in terms of Cohen's Kappa (Carletta, 1996), are shown in Table 2b. Given that the majority of sentences were labeled as NONE, we avoid inflating agreement by *not* considering the sentences labeled with NONE by both annotators when computing Kappa. As we can see, we achieved substantial post-level agreement and high sentence-level agreement.

The major source of inter-annotator disagreement for all four datasets stems from the fact that in many cases, the annotators, while agreeing on the reason class, differ on how long the text span for a reason should be. This hurts sentence-level agreement but not post-level agreement, since the latter only concerns *whether* a reason was mentioned in a post, and explains why the sentence-level agreement scores are lower than the corresponding post-level scores. Minor sources of disagreement arise from the facts that (1) the annotators selected different reason labels for some of the multi-labeled sentences, and (2) they tend to disagree in some cases where authors use sarcasm to present a reason. Each case of disagreement is resolved through discussion among the annotators.

	ABO	GAY	OBA	MAR
Stance-labeled posts	1741	1376	985	626
for posts (%)	54.9	63.4	53.9	69.5
Average post sequence length	4.1	4.0	2.6	2.5

(a) Statistics of stance-labeled posts

	ABO	GAY	OBA	MAR
Reason-labeled posts	463	561	447	432
% of sentences w/ reason tags	20.4	29.8	34.4	43.7
Kappa (sentence)	0.66	0.63	0.61	0.67
Kappa (post)	0.82	0.80	0.78	0.83

(b) Statistics of reason-labeled posts

Table 2: Stance and reason annotation statistics.

3 Baseline RC System

Our baseline system uses a maximum entropy (MaxEnt) classifier to determine whether a reason is expressed in a post and/or its sentence(s). We create one training instance for each sentence in each post in the training set, using the reason label as its class label. We represent each instance using five types of features, as described below.

N-gram features. We encode each unigram and bigram collected from the training sentences as a

binary feature indicating the n-gram's presence or absence in a given sentence.

Dependency-based features. To capture the inter-word relationships that n-grams may not, we employ the dependency-based features previously used for stance classification in Anand et al. (2011). These features have three variants. In the first variant, the pair of arguments involved in each dependency relation extracted by a dependency parser is used as a feature. The second variant is the same as the first except that the head (i.e., the first argument in a relation) is replaced by its part-of-speech tag. The features in the third variant, the topic-opinion features, are created by replacing each sentiment-bearing word in features of the first two types with its corresponding polarity label (i.e., + or −).

Frame-semantic features. While dependency-based features capture the syntactic dependencies, frame-semantic features encode the semantic representation of the concepts in a sentence. Following our previous work on stance classification (Hasan and Ng, 2013c), we employ three types of features computed based on the frame-semantic parse of each sentence in a post obtained from SEMAFOR (Das et al., 2010). *Frame-word interaction* features encode whether two words appear in different elements of the same frame. Hence, each frame-word interaction feature consists of (1) the name of the frame f from which it is created, and (2) an unordered word pair in which the words are taken from two frame elements of f. A *frame-pair* feature is represented as a word pair corresponding to the names of two frames and encodes whether the target word of the first frame appears within an element of the second frame. Finally, *frame n-gram* features are a variant of word n-grams. For each word n-gram in the sentence, a frame n-gram feature is created by replacing one or more words in the word n-gram with the name of the frame or the frame element in which the word appears. A detailed description of these features can be found in Hasan and Ng (2013c).

Quotation features. We employ two quotation features. *IsQuote* is a binary feature that indicates whether a sentence is a quote or not (i.e., whether it appeared in its parent post in the post sequence). Note that if an instance is a quote from a previous post, it is unlikely that it represents a reason the author is presenting to support her argument. Instead, the author may have quoted this before stating her counter-argument. *FollowsQuote* is a binary feature that indicates whether a sentence follows a sentence for which the *IsQuote* feature value is *true*. Intuitively, a sentence following a quote is likely to present a counter-argument.

Positional feature. We split each post into four parts (such that each part contains roughly the same number of sentences) and create one positional feature that encodes which part of the post contains a given sentence. This feature is motivated by our observations on the training data that (1) reasons are more likely to appear in the second half of a post and (2) on average more than one-third of the reasons appear in the last quarter of a post.

After training, we can apply the resulting RC system to classify the test instances, which are generated in the same way as the training instances. Once the sentences of a test post are classified, we simply assume its post-level reason labels to be the set of reason labels assigned by the classifier to its sentences.

4 Stance-Supported RC Systems

In this section, we propose a set of systems for RC. Unlike the baseline RC system, these RC systems are *stance-supported*, enabling us to explore how different ways of modeling automatically computed stances and reasons can improve RC classification. Below we present our systems in increasing order of modeling sophistication.

4.1 Pipeline Systems

We examine two pipeline systems, P1 and P2. Given a set of test posts, both systems first determine the stance of each post and then apply a stance-specific *reason classifier* to each of them.

More specifically, both P1 and P2 employ two stance-specific reason classifiers: one is trained on all the posts labeled as *for* and the other is trained on all the posts labeled as *against*. Each stance-specific reason classifier is trained using MaxEnt on the same feature set as that of the Baseline RC system. It computes for a particular stance s the probability $P(r|s,t)$, where r is a reason label and t is a sentence in a test post p.

P1 and P2 differ only with respect to the SC model used to stance-label each post. In P1, the stance s of a post p is determined by applying to p a stance classifier that computes $P(s|p)$. To train the classifier, we employ MaxEnt. Each train-

ing instance corresponds to a training post and is represented by all but the quotation and positional features used to train the Baseline RC system, since these two feature types are sentence-based rather than post-based. After training, the resulting classifier can be used to stance-label a post independently of the other posts.

In P2, on the other hand, we recast SC as a sequence labeling task. In other words, we train a SC model that assumes as input a post sequence and outputs a stance sequence, with one stance label for each post in the input post sequence. This choice is motivated by an observation we made previously (Hasan and Ng, 2013a): since each post in a sequence is a reply to the preceding post, we could exploit their dependencies by determining their stance labels together.[3]

As our sequence learner, we employ a maximum entropy Markov model (MEMM) (McCallum et al., 2000). Given an input post sequence $P_S = (p_1, p_2, \ldots, p_n)$, the MEMM finds the most probable stance sequence $S = (s_1, s_2, \ldots, s_n)$ by computing $P(S|P_S)$, where:

$$P(S|P_S) = \prod_{k=1}^{n} P(s_k|s_{k-1}, p_k) \qquad (1)$$

This probability can be computed efficiently via dynamic programming (DP), using a modified version of the Viterbi algorithm (Viterbi, 1967).

There is a caveat, however. Recall that the post sequences are generated from a thread. Since a test post may appear in more than one sequence, different occurrences of it may be assigned different stance labels by the MEMM. To determine the final stance label for the post, we average the probabilities assigned to the *for* stance over all its occurrences; if the average is ≥ 0.5, then its final label is *for*; otherwise, its label is *against*.

4.2 System based on Joint Inference

One weakness of the pipeline systems is that errors may propagate from the SC system to the RC system. If the stance of a post is incorrectly labeled, its reasons will also be incorrectly labeled.

To avoid this problem, we employ joint inference. Specifically, we first train a SC system and a RC system independently of each other. We employ the Baseline as our RC system, since this is the only RC system that is not stance-specific. For the SC system, we employ P2.

Since the SC system and the RC system are trained independently of each other, their outputs may not be consistent. For instance, an inconsistency arises if a post is labeled as *for* but one or more of its reasons are associated with the opposing stance. In fact, an inconsistency can arise in the output of the RC system alone: reasons associated with both stances may be assigned by the RC systems to different sentences of a given post.

To enforce consistency, we apply integer linear programming (ILP) (Roth and Yih, 2004). We formulate one ILP program for each debate post. Each ILP program contains two post-stance variables (x_{for} and $x_{against}$) and $|T| * |L_R|$ reason variables (i.e., one indicator variable $z_{t,r}$ for each reason class r and each sentence t), where $|T|$ is the number of sentences in the post and $|L_R|$ is the number of reason labels. Our objective is to maximize the linear combination of these variables and their corresponding probabilities assigned by their respective classifiers (see (2) below) subject to two types of constraints, the *integrity* constraints and the *post-reason* constraints. The integrity constraints ensure that each post is assigned exactly one stance and each sentence in a post is assigned exactly one reason class (see the two equality constraints in (3)). The post-reason constraints ensure consistency between the predictions made by the SC and the RC systems. Specifically, (1) if there is at least one reason supporting the *for* stance, the post must be assigned a *for* label; and (2) a *for* post must have at least one *for* reason. These constraints are defined for the *against* label as well (see the constraints in (4)).

Maximize:

$$\sum_{s \in L_S} a_s x_s + \frac{1}{|T|} \sum_{t=1}^{|T|} \sum_{r \in L_R} b_{t,r} z_{t,r} \qquad (2)$$

subject to:

$$\sum_{s \in L_S} x_s = 1, \ \forall\, t \sum_{r \in L_R} z_{t,r} = 1,$$
$$x_s \in \{0,\ 1\},\ z_{t,r} \in \{0,\ 1\} \qquad (3)$$

$$\forall\, t \ \ x_s \geq z_{t,r},\ \sum_{t=1}^{|T|} z_{t,r} \geq x_s \qquad (4)$$

[3] While we could similarly recast the problem of assigning reasons to the sentences in a post as a sequence learning task, we did not pursue this idea further because preliminary experiments indicated that sequence learning for RC was ineffective: there is little, if any, dependency between the reason labels in consecutive sentences.

Note that (1) a_s and $b_{t,r}$ are two sets of probabilities assigned by the SC and RC systems respectively; (2) L_S and L_R denote the set of stance labels and reason labels respectively; and (3) the fraction $\frac{1}{|T|}$ ensures that both classifiers are contributing equally to the objective function.

4.3 Systems based on Joint Learning

Another way to avoid the error propagation problem in pipeline systems is to perform joint learning. In joint learning, the two tasks, SC and RC, are learned jointly. Below we propose three joint models in increasing level of sophistication.

J1 is a joint model that, given a test post p, finds the stance label s and the reason label for each of the sentences that together maximize the probability $P(R_p, s|p)$, where $R_p = (r_1, r_2, \ldots, r_n)$ is the sequence of reason labels with r_i $(1 \leq i \leq n)$ being the reason label assigned to t_i, the i-th sentence in p. Using Chain Rule,

$$\begin{aligned} P(R_p, s|p) &= P(s|p)P(R_p|s, p) \\ &= P(s|p)\prod_{i=1}^{n} P(r_i|s, t_i) \end{aligned} \tag{5}$$

Hence, $P(R_p, s|p)$ can be computed by using the stance-specific RC classifier and the SC classifier employed in P1.

The second joint model, J2, is the same as J1, except that we recast SC as a sequence labeling task. As before, we employ MEMM to learn how to predict stance sequences. Given a post sequence $P_S = (p_1, p_2, \ldots, p_n)$, J2 finds the stance sequence $S = (s_1, s_2, \ldots, s_n)$ and reasons $R = (R_1, R_2, \ldots, R_n)$ that jointly maximize $P(R, S|P_S)$. Note that R_i is the sequence of reason labels assigned to the sentences in post i.

The R and S that jointly maximize $P(R, S|P_S)$ can be found efficiently via DP, using a modified version of the Viterbi algorithm. Unlike in P2, in J2 the decoding process is slightly more complicated because we have to take into account R_i. Below we show the recursive definitions used to compute the entries in the DP table, where $v_k(h)$ is the (k,h)-th entry of the table; $P(h|p)$ is provided by the MaxEnt stance classifier used in P1; $P(h|j, p)$ is provided by the MEMM stance classifier used in P2; $P(r_i^{max}|h, t_i)$ is provided by the stance-specific reason classifier used in the pipeline systems; and r_i^{max} is the reason label for sentence t_i that has the highest probability according to the reason classifier.

Base case:

$$v_1(h) = P(h|p)\prod_{i=1}^{n} P(r_i^{max}|h, t_i) \tag{6}$$

Recursive definition:

$$v_k(h) = \max_j v_{k-1}(j)P(h|j, p)\prod_{i=1}^{n} P(r_i^{max}|h, t_i) \tag{7}$$

To motivate our third joint model, J3, we make the following observation. Recall that a post in a post sequence is a reply to its preceding post. An inspection of the training data reveals that in many cases, a reply is a rebuttal to the preceding post, where an author attempts to argue why the points or reasons raised in the preceding post are wrong and then provides her reasons for the opposing stance. Motivated by this observation, we hypothesize that the reasons mentioned in the preceding post could be useful for predicting the reasons in the current post. However, none of the models we have presented so far makes use of the reasons predicted for the preceding post.

This motivates the design of J3, which we build on top of J2. Specifically, to incorporate the reason labels predicted for the preceding post in a post sequence, we augment the feature set of the stance-specific reason classifiers with a set of *reason features*, with one binary feature for each reason. The value of a reason feature is 1 if and only if the corresponding reason is predicted to be present in the preceding post. Hence, in J3, we can apply the same DP equations we used in J2 except that the set of features used by the reason classifier is augmented with the reason features.

5 Evaluation

While our primary goal is to evaluate the RC systems introduced in the previous section, we are also interested in whether SC performance can improve when SC is jointly modeled with RC. More specifically, our evaluation is driven by the following question: will RC performance and SC performance improve as we employ more sophisticated methods for modeling reasons and stances? Before showing the results, we describe the metrics for evaluating RC and SC systems.

System	ABO			GAY			OBA			MAR		
	Stance	Reason		Stance	Reason		Stance	Reason		Stance	Reason	
		Sentence	Post		Sentence	Post		Sentence	Post		Sentence	Post
Baseline	–	32.7	45.0	–	23.3	40.5	–	19.5	31.5	–	28.7	44.2
P1	62.8	34.5	46.3	63.4	24.5	43.2	61.0	20.3	33.5	67.2	30.5	47.3
P2	65.1	36.1	47.7	64.2	26.6	45.5	63.8	21.1	34.4	68.5	32.9	48.8
ILP	65.2	36.5	48.4	64.6	28.0	46.7	63.6	22.8	35.0	68.8	33.1	48.9
J1	62.5	36.0	47.6	64.0	26.7	45.6	61.2	23.1	35.7	67.8	33.3	49.2
J2	65.9	37.9	50.6	65.3	29.6	48.5	63.5	24.5	37.1	68.7	34.5	50.5
J3	**66.3**	**39.5**	**52.3**	**65.7**	**31.4**	**49.8**	**64.0**	**25.1**	**38.0**	**69.0**	**35.1**	**51.1**

Table 3: SC accuracies and RC F-scores for our five-fold cross-validation experiments.

5.1 Experimental Setup

We express SC results in terms of *accuracy* (i.e., the percentage of test posts labeled with the correct stance) and RC results in terms of *F-score* micro-averaged over all reason classes except the NONE class. For each RC system, we report its sentence-level RC score and post-level RC score, which are computed over sentences and posts respectively. As mentioned at the end of Section 3, the set of post-level reason labels of a given post is automatically obtained by taking the union of the set of reason labels assigned to each of its sentences. Hence, a reason classifier will be rewarded as long as it can predict, for any sentence in a test post, a reason label that the annotators assigned to some sentence in the same post.

We obtain these scores via five-fold cross-validation experiments. During fold partition, all posts that are in the same post sequence are assigned to the same fold. All reason and stance classifiers are domain-specific, meaning that each of them is trained on sentences/posts from exactly one domain and is applied to classify sentences/posts from the same domain. We use the Stanford maximum entropy classifier[4] for classification and solve ILP programs using *lpsolve*[5].

5.2 Results and Discussion

Results are shown in Table 3. Each row corresponds to one of our seven RC systems, showing its SC accuracy as well as its sentence- and post-level RC F-scores for each domain.

Let us begin by discussing the RC results. **First**, P1 and P2 significantly beat the Baseline on all four domains by an average of 1.4 and 3.1 points at the sentence level and by an average of 2.3 and 3.8 points at the post level respectively.[6] These results show that stance information can indeed be profitably used for RC even if it is incorporated into RC systems in a simple manner. **Second**, improving SC through sequence learning can improve RC: the systems in which SC is recast as sequence labeling (P2 and J2) perform significantly better than the corresponding systems that do not (P1 and J1). **Third**, ILP significantly beats P2 on two domains (ABO and GAY) and achieves the same level of performance as P2 on the remaining domains. These results suggest that joint inference is no worse (and sometimes even better) than pipeline learning as far as exploiting stance information for RC is concerned. **Fourth**, the systems trained via joint learning (J1 and J2) beat their corresponding pipeline counterparts (P1 and P2) on all four domains, significantly so by an average of 2.3 and 2.5 points at the sentence level and by an average of 2.0 and 2.6 points at the post level respectively, suggesting that joint learning is indeed a better way to incorporate stance information than pipeline learning. **Finally**, J3, the joint system that exploits reasons predicted for the previous post, significantly beats J2, the system on which it is built, by 1.6 and 1.8 points at the sentence level and by 1.7 and 1.3 points at the post level for ABO and GAY respectively. It also yields small, statistically insignificant, improvements (0.6 points at the sentence level and 0.6–0.9 points at the post level) for the remaining two domains. These results suggest that the reasons predicted for the previous post indeed provide useful information for predicting the current post's reasons.

Overall, these results are consistent with our hy-

[4] http://nlp.stanford.edu/software/classifier.shtml

[5] http://sourceforge.net/projects/lpsolve/

[6] All significance tests are paired t-tests ($p < 0.05$).

pothesis that the usefulness of stance information depends in part on the way it is exploited, and that RC performance increases as we employ more sophisticated methods for modeling reasons and stances. Our best system, J3, significantly beats the Baseline by an average of 6.7 and 7.5 points at the sentence and post levels respectively.

As mentioned earlier, a secondary goal of this work is to determine whether joint modeling can improve SC as well. For that reason, we compare the performances of the best pipeline model (P2) and the best joint model (J3) on each domain. We find that in terms of SC accuracy, J3 is significantly better than P2 on ABO and GAY, and yields slightly, though insignificantly, better performance on the remaining two domains. In other words, our results suggest that joint modeling of SC and RC has a positive impact on SC performance on all domains, and the impact can sometimes be large enough to yield significantly better results.

5.3 Further Comparison

We hypothesized in the introduction that the sentence-based approach to post-level RC would yield better performance than the multi-label text classification approach. In Section 5.2, we presented results of the sentence-based approach to RC. So, to test this hypothesis, we next evaluate the multi-label text classification approach to RC.

Recall that the multi-label text classification approach assumes the following setup. Given a set of training posts where each post is multi-labeled with the set of reasons it contains, the goal is to train a system to determine the set of reasons a test post contains. Hence, unlike in the sentence-based approach, in this approach no sentence-level reason annotations are exploited during training.

We implement this approach by recasting multi-label text classification as n binary text classification tasks, where n is the number of reason classes for a domain. In the binary classification task for predicting reason i, we train a binary classifier c_i using the one-versus-all training scheme. Specifically, to train c_i, we create one training instance for each post p in the training set, labeling it as positive if and only if p contains reason i. Note that if i is a minority reason, the class distribution of the resulting training set will be highly skewed towards the negatives, which will in turn cause the resulting MaxEnt classifier to be biased towards predicting a test instance as negative.

To address this problem, we adjust the classification thresholds associated with the binary classifiers. Recall that a test instance is classified as positive by a binary classifier if and only if its probability of belonging to the positive class is above the classification threshold used. Hence, adjusting the threshold amounts to adjusting the number of test instances classified as positive, thus addressing the bias problem mentioned above. Specifically, we adjust the thresholds of the classifiers as follows. We train the binary classifiers to optimize the overall F-score by jointly tuning their classification thresholds on 25% of the training data reserved for development purposes. Since computing the exact solution to this optimization problem is computationally expensive, we employ a local search algorithm that changes the value of one threshold at a time to optimize F-score while keeping the remaining thresholds fixed. During testing, classifier c_i will classify a test instance as positive if its probability of belonging to the positive class is above the corresponding threshold.

We apply this multi-label text classification approach to obtain post-level RC scores for the Baseline, P1 and P2. Note that since P1 and P2 are pipeline systems, the binary classifiers they use to predict a test post's reasons depend on the post's predicted stance. Specifically, if a test post is predicted to have a positive (negative) stance, then only the reason classifiers associated with the positive (negative) stance will be used to predict the reasons it contains. On the other hand, this approach cannot be used in combination with ILP or the joint models to produce post-level RC scores: they all require a reason classifier trained on reason-annotated *sentences*, which are not available in the multi-label text classification approach.

Post-level RC results of the Baseline and the two pipeline systems, P1 and P2, obtained via this multi-label text classification approach are shown in Table 4. These scores are significantly lower than the corresponding scores in Table 3 by 3.2, 2.9, and 3.1 points for the Baseline, P1, and P2 respectively, when averaged over the four domains. They confirm our hypothesis that the sentence-based approach to post-level RC is indeed better than its multi-class text classification counterpart.

5.4 Error Analysis

To get a better understanding of our best-performing RC system (J3), we examine its major

System	ABO	GAY	OBA	MAR
Baseline	39.8	37.9	30.1	40.8
P1	41.5	41.0	31.7	44.7
P2	43.3	42.6	32.0	46.3

Table 4: Post-level RC F-scores obtained via the multi-class text classification approach.

sources of error in this subsection.

For the four domains, 75–83% of the errors can be attributed to the system's inability to decide whether a sentence describes a reason or not. Specifically, in 51–54% of the erroneous cases, a reason sentence is misclassified as NONE. On the other hand, 23–30% of the cases are concerned with assigning a reason label to a NONE sentence. The remaining 17–25% of the errors concern mislabeling a reason sentence with the wrong reason.

A closer examination of the errors reveals that they resulted primarily from (1) the lack of access to background knowledge, (2) the failure to process complex discourse structures, and (3) the failure to process sarcastic statements and rhetorical questions. We present two examples for each of these three major sources of error from the ABO and OBA domains in Table 5. In each example, we show their predicted (P) and gold (G) labels.

Lack of access to background knowledge. Consider the first example for ABO in Table 5. Our system misclassifies this sentence in part because it lacks the background knowledge that "genetic code" is one of the characteristics of life and a fetus having it means a fetus has life (A3). Similarly, the system cannot determine the reason for the first OBA example without the knowledge that "deficit spending" is a term related to the economy and that increasing it is bad (A1). We believe some of these relations can be extracted from lexical knowledge bases such as YAGO2 (Suchanek et al., 2007), Freebase (Bollacker et al., 2008), and BabelNet (Navigli and Ponzetto, 2012).

Failure to process complex discourse structures. Our system misclassifies the second example for ABO in part because the first part of the sentence (i.e., *Sure, the fetus has the potential to one day be a person*) expresses a meaning that is completely inverted by the second part. Such complex discourse structures often lead to classification errors even for sentences whose interpretation requires no background knowledge. We believe that this problem can be addressed in part by a better understanding of the structure of a discourse, particularly the relation between two discourse segments, using a discourse parser.

Failure to process sarcastic statements and rhetorical questions. Owing to the nature of our dataset (i.e., debate posts), many errors arise from sentences containing sarcasm and/or rhetorical questions. This is especially a problem in long post sequences, where authors frequently restate their opponents' positions, sometimes ironically. A first step towards handling these errors would therefore be to identify sentences containing sarcasm and/or rhetorical questions.

6 Related Work

In this section, we discuss related work in the areas of document-level RC, argument recognition, textual entailment in online debates, argumentation mining, and sentiment analysis.

Document-level reason classification. Persing and Ng (2009) apply a multi-label text classification approach to document-level RC of aviation safety incident reports. Given a set of pre-defined reasons, their RC system seeks to identify the reasons that can explain why the incident described in a given report occurred. Their work is different from ours in at least two respects. First, while our posts occur in post sequences (which can be profitably exploited in RC, for example, as in J3), their incident reports were written independently of each other. Second, they do not perform sentence-level RC, as the lack of sentence-level reason annotations in their dataset prevented them from training a sentence-level reason classifier.

Argument recognition. Boltužić and Šnajder (2014) propose a multi-class classification task called *argument recognition* in online discussions. Given a post and a reason for a particular domain, the task is to predict the extent to which the author of the post supports or opposes the reason as measured on a five-point ordinal scale ranging from "explicitly supports" to "explicitly opposes". Hence, unlike RC, argument recognition focuses on the *magnitude* rather than the *existence* of a post-reason relationship. In addition, Boltužić and Šnajder focus on post-level (rather than sentence-level) classification and employ perfect (rather than predicted) stance information.

Textual entailment in online debates. Given the title of a debate and a post written in response to it, this task seeks to detect arguments in

Domain	Background knowledge			Complex discourse structure			Sarcasm/rhetorical questions		
	Example	P	G	Example	P	G	Example	P	G
ABO	Science does agree that the fetus has an individual genetic code and fits into the biological definition of life.	NONE	A3	Sure, the fetus has the potential to one day be a person, but right now it is not.	NONE	F3	So are there enough homes for 50,000,000 babies?	F9	F5
OBA	Democrats have increased deficit spending by 2 trillion dollars over 2 years.	NONE	A1	Bush raised the debt by two billion for the wars, Obama has outspent that in a week.	A2	A1	I agree, Bush put us in debt for the next 100 years, so we can blame Obama forever.	NONE	F3

Table 5: Examples of the major sources of error. P and G stand for predicted tag and gold tag respectively.

the post that entail or contradict the title (Cabrio and Villata, 2012). Hence, this task is concerned with *identifying* text segments that correspond to rationales without a predefined set of rationales, whereas RC is concerned with both *identifying* text segments and *classifying* them based on a given set of reasons.

Argumentation mining. The goal of this task is to extract the argumentative structure of a document. Researchers have proposed approaches to mine the structure of scientific papers (Teufel and Moens, 2000; Teufel, 2001), product reviews (Villalba and Saint-Dizier, 2012; Wyner et al., 2012), newspaper articles (Feng and Hirst, 2011), and legal documents (Brüninghaus and Ashley, 2005; Wyner et al., 2010; Palau and Moens, 2011; Ashley and Walker, 2013). A major difference between this task and RC is that the argument types refer to generic structural cues, textual patterns etc., whereas our reason classes refer to the specific reasons an author may mention to support her stance in a domain. For instance, in the case of a scientific article, the argument types correspond to general background, description of the paper's or some other papers' approach, objective, contrastive and/or comparative comments, etc. (Teufel and Moens, 2000). The argument types for legal documents refer to legal factors which are either pro-plaintiff or pro-defendant (Brüninghaus and Ashley, 2005). For instance, for trade secret law cases, factors such as *Waiver-of-Confidentiality* and *Disclosure-in-Public-Forum* refer to certain facts strengthening the claim of one of the sides participating in a case.

Sentiment analysis. RC resembles certain tasks in sentiment analysis. One such task is pro and con reason classification in reviews (Kim and Hovy, 2006), where sentences containing opinions as well as reasons justifying the opinions are to be extracted and classified as PRO, CON, or NONE. Hence, this task focuses on categorizing sentences into coarse-grained, high-level groups (e.g., PRO vs. CON, POSITIVE vs. NEGATIVE), but does not attempt to subcategorize the PRO and CON classes into fine-grained reason classes, unlike RC. Somewhat similar to the PRO and CON sentence classification task is the task of determining the *relevance* of a sentence in a review for polarity classification. Zaidan et al. (2007) coined the term *rationale* to refer to any subjective textual content that contains evidence supporting the author's opinion or stance. These rationales, however, may not always contain reasons. For instance, a sentence that mentions that the author likes a product is a rationale, but it does not contain any reason for her liking it. Methods have been proposed for automatically identifying rationales (e.g., Yessenalina et al. (2010), Trivedi and Eisenstein (2013)) and distinguishing subjective from objective materials in a review (e.g., Pang and Lee (2004), Wiebe and Riloff (2005), McDonald et al. (2007), Zhao et al. (2008)). Note that in all these attempts, the end goal is not to classify sentences, but to employ the results of sentence classification to improve a higher-level task, such as sentiment classification.

7 Conclusion

We examined the new task of reason classification. We exploited stance information for reason classification, proposing systems of varying complexity for modeling stances and reasons. Experiments on our reason-annotated corpus of ideological debate posts from four domains demonstrate that sophisticated models of stances and reasons can indeed yield more accurate reason and stance classification results than their simpler counterparts. Nevertheless, reason classification remains a challenging task: the best post-level F-scores are in the low 50s. By making our corpus publicly available, we hope to stimulate further research on this task.

Acknowledgments

We thank the three anonymous reviewers for their detailed and insightful comments on an earlier draft of this paper. This work was supported in part by NSF Grants IIS-1147644 and IIS-1219142. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of NSF.

References

Rakesh Agrawal, Sridhar Rajagopalan, Ramakrishnan Srikant, and Yirong Xu. 2003. Mining newsgroups using networks arising from social behavior. In *Proceedings of the 12th International Conference on World Wide Web*, pages 529–535.

Pranav Anand, Marilyn Walker, Rob Abbott, Jean E. Fox Tree, Robeson Bowmani, and Michael Minor. 2011. Cats rule and dogs drool!: Classifying stance in online debate. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pages 1–9.

Kevin D. Ashley and Vern R. Walker. 2013. From information retrieval (IR) to argument retrieval (AR) for legal cases: Report on a baseline study. In *Proceedings of the 26th International Conference on Legal Knowledge and Information System*, pages 29–38.

Mohit Bansal, Claire Cardie, and Lillian Lee. 2008. The power of negative thinking: Exploiting label disagreement in the min-cut classification framework. In *COLING 2008: Companion volume: Posters*, pages 15–18.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Filip Boltužić and Jan Šnajder. 2014. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58.

Stefanie Brüninghaus and Kevin D. Ashley. 2005. Reasoning with textual cases. In *Proceedings of the 6th International Conference on Case-Based Reasoning*, pages 137–151.

Clinton Burfoot, Steven Bird, and Timothy Baldwin. 2011. Collective classification of congressional floor-debate transcripts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1506–1515.

Elena Cabrio and Serena Villata. 2012. Natural language arguments: A combined approach. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pages 205–210.

Jean Carletta. 1996. Assessing agreement on classification tasks: The Kappa statistic. *Computational Linguistics*, 22(2):249–254.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. SEMAFOR 1.0: A probabilistic frame-semantic parser. Technical report, Carnegie Mellon University Technical Report CMU-LTI-10-001.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 987–996.

Kazi Saidul Hasan and Vincent Ng. 2013a. Extra-linguistic constraints on stance recognition in ideological debates. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 816–821.

Kazi Saidul Hasan and Vincent Ng. 2013b. Frame semantics for stance classification. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 124–132.

Kazi Saidul Hasan and Vincent Ng. 2013c. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356.

Soo-Min Kim and Eduard Hovy. 2006. Automatic identification of pro and con reasons in online reviews. In *Proceedings of the COLING/ACL Main Conference Poster Sessions*, pages 483–490.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the 17th International Conference on Machine Learning*, pages 591–598.

Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 432–439.

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

Raquel Mochales Palau and Marie-Francine Moens. 2011. Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 271–278.

Isaac Persing and Vincent Ng. 2009. Semi-supervised cause identification from aviation safety reports. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 843–851.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning*, pages 1–8.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 226–234.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A core of semantic knowledge. In *Proceedings of the 16th International World Wide Web Conference*, pages 697–706.

Simone Teufel and Marc Moens. 2000. What's yours and what's mine: Determining intellectual attribution in scientific text. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 9–17.

Simone Teufel. 2001. Task-based evaluation of summary quality: Describing relationships between scientific papers. In *Proceedings of the NAACL Workshop on Automatic Summarization*, pages 12–21.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 327–335.

Rakshit Trivedi and Jacob Eisenstein. 2013. Discourse connectors for latent subjectivity in sentiment analysis. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 808–813.

Maria Paz Garcia Villalba and Patrick Saint-Dizier. 2012. Some facets of argument mining for opinion analysis. In *Proceedings of the Fourth International Conference on Computational Models of Argument*, pages 23–34.

Andrew J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Marilyn Walker, Pranav Anand, Rob Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 592–596.

Janyce Wiebe and Ellen Riloff. 2005. Creating subjective and objective sentence classifiers from unannotated texts. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 486–497.

Adam Wyner, Raquel Mochales-Palau, Marie-Francine Moens, and David Milward. 2010. Approaches to text mining arguments from legal cases. In Enrico Francesconi, Simonetta Montemagni, Wim Peters, and Daniela Tiscornia, editors, *Semantic Processing of Legal Texts: Where the Language of Law Meets the Law of Language*, pages 60–79. Springer-Verlag.

Adam Wyner, Jodi Schneider, Katie Atkinson, and Trevor J. M. Bench-Capon. 2012. Semi-automated argumentative analysis of online product reviews. In *Proceedings of the Fourth International Conference on Computational Models of Argument*, pages 43–50.

Ainur Yessenalina, Yejin Choi, and Claire Cardie. 2010. Automatically generating annotator rationales to improve sentiment classification. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 336–341.

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 260–267.

Jun Zhao, Kang Liu, and Gen Wang. 2008. Adding redundant features for crfs-based sentence sentiment classification. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 117–126.

Chinese Zero Pronoun Resolution: An Unsupervised Probabilistic Model Rivaling Supervised Resolvers

Chen Chen and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{yzcchen,vince}@hlt.utdallas.edu

Abstract

State-of-the-art Chinese zero pronoun resolution systems are supervised, thus relying on training data containing manually resolved zero pronouns. To eliminate the reliance on annotated data, we present a generative model for unsupervised Chinese zero pronoun resolution. At the core of our model is a novel hypothesis: a probabilistic pronoun resolver trained on overt pronouns in an unsupervised manner can be used to resolve zero pronouns. Experiments demonstrate that our unsupervised model rivals its state-of-the-art supervised counterparts in performance when resolving the Chinese zero pronouns in the OntoNotes corpus.

1 Introduction

A zero pronoun (ZP) is a gap in a sentence that is found when a phonetically null form is used to refer to a real-world entity. An anaphoric zero pronoun (AZP) is a ZP that corefers with one or more preceding noun phrases (NPs) in the associated text. Below is an example taken from the Chinese TreeBank (CTB), where the ZP (denoted as *pro*) refers to 俄罗斯 (Russia).

[俄罗斯] 作为米洛舍夫维奇一贯的支持者，*pro* 曾经提出调停这场政治危机。

([Russia] is a consistent supporter of Milošević, *pro* has proposed to mediate the political crisis.)

As we can see, ZPs lack grammatical attributes that are useful for overt pronoun resolution such as NUMBER and GENDER. This makes ZP resolution more challenging than overt pronoun resolution.

Automatic ZP resolution is typically composed of two steps. The first step, AZP identification, involves extracting ZPs that are anaphoric. The second step, AZP resolution, aims to identify an antecedent of an AZP. State-of-the-art ZP resolvers have tackled both of these steps in a supervised manner, training a classifier for AZP identification and another one for AZP resolution (e.g., Zhao and Ng (2007), Chen and Ng (2013)).

In this paper, we focus on the second task, AZP resolution, designing a model that assumes as input the AZPs in a document and resolves each of them. Note that the task of AZP resolution alone is by no means easy: even when gold-standard AZPs are given, state-of-the-art supervised resolvers can only achieve an F-score of 47.7% for *resolving* Chinese AZPs (Chen and Ng, 2013). For the sake of completeness, we will evaluate our AZP resolution model using both gold-standard AZPs as well as AZPs automatically identified by a rule-based approach that we propose in this paper.

Our contribution lies in the proposal of the first *unsupervised* probabilistic model for AZP resolution that rivals its supervised counterparts in performance when evaluated on the Chinese portion of the OntoNotes 5.0 corpus. Its main advantage is that it does not require training data with manually resolved AZPs. This, together with the fact that its underlying generative process is not language-dependent, enables it to be applied to languages where such annotated data is not readily available. At its core is a novel hypothesis: we can apply a probabilistic pronoun resolution model trained on *overt* pronouns in an *unsupervised* manner to resolve *zero* pronouns. Motivated by Cherry and Bergsma's (2005) and Charniak and Elsner's (2009) work on unsupervised English pronoun resolution, we train our unsupervised resolver on Chinese overt pronouns using the Expectation-Maximization (EM) algorithm (Dempster et al., 1977).

2 Related Work

Chinese ZP resolution. Early approaches to Chinese ZP resolution are *rule-based*. Converse (2006) applied Hobbs' algorithm (Hobbs,

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 763–774,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

1978) to resolve the ZPs in the CTB documents. Yeh and Chen (2007) hand-engineered a set of rules for ZP resolution based on Centering Theory (Grosz et al., 1995).

In contrast, virtually all recent approaches to this task are based on *supervised* learning. Zhao and Ng (2007) are the first to employ a supervised learning approach to Chinese ZP resolution. They trained an AZP resolver by employing syntactic and positional features in combination with a decision tree learner. Unlike Zhao and Ng, Kong and Zhou (2010) employed context-sensitive convolution tree kernels (Zhou et al., 2008) in their resolver to model syntactic information. More recently, we extended Zhao and Ng's feature set with novel features that encode the context surrounding a ZP and its candidate antecedents, and exploited the coreference links between ZPs as bridges to find textually distant antecedents for ZPs (Chen and Ng, 2013).

ZP resolution for other languages. There have been rule-based and supervised machine learning approaches for resolving ZPs in other languages. For example, to resolve ZPs in Spanish texts, Ferrández and Peral (2000) proposed a set of handcrafted rules that encode preferences for candidate antecedents. In addition, supervised approaches have been extensively employed to resolve ZPs in Korean (e.g., Han (2006)), Japanese (e.g., Seki et al. (2002), Isozaki and Hirao (2003), Iida et al. (2006; 2007), Imamura et al. (2009), Iida and Poesio (2011), Sasano and Kurohashi (2011)), and Italian (e.g., Iida and Poesio (2011)).

3 Chinese Overt Pronouns

Since our approach relies heavily on Chinese *overt* pronouns, in this section we introduce them by describing their four grammatical attributes, namely NUMBER, GENDER, PERSON and ANIMACY. NUMBER has two values, *singular* and *plural*. GENDER has three values, *neuter*, *masculine* and *feminine*. PERSON has three values, *first*, *second* and *third*. Finally, ANIMACY has two values, *animate* and *inanimate*.

We exploit ten personal pronouns that have well-defined grammatical attribute values, namely 你 (singular you), 我 (I), 他 (he), 她 (she), 它 (it), 你们 (plural you), 我们 (we), 他们 (masculine they), 她们 (feminine they), and 它们 (impersonal they). As can be seen in Table 1, each of them can be uniquely identified using these four attributes.

Pronouns	NUMBER	GENDER	PERSON	ANIMACY
我 (I)	singular	neuter	first	animate
你 (you)	singular	neuter	second	animate
他 (he)	singular	masculine	third	animate
她 (she)	singular	feminine	third	animate
它 (it)	singular	neuter	third	inanimate
你们 (you)	plural	neuter	second	animate
我们 (we)	plural	neuter	first	animate
他们 (they)	plural	masculine	third	animate
她们 (they)	plural	feminine	third	animate
它们 (they)	plural	neuter	third	inanimate

Table 1: Attribute values of Chinese overt pronouns.

4 The Generative Model

4.1 Notation

Let p be an overt pronoun in PR, the set of the 10 overt pronouns described in Section 3. C, the set of candidate antecedents of p, contains all and only those maximal or modifier NPs that precede p in the associated text and are at most two sentences away from it.[1] k is the context surrounding p as well as every candidate antecedent c in C; k_c is the context surrounding p and candidate antecedent c; and l is a binary variable indicating whether c is the correct antecedent of p. The set $A = \{Num, Gen, Per, Ani\}$ has four elements, which correspond to NUMBER, GENDER, PERSON and ANIMACY respectively. a is an attribute in A. Finally, p_a and c_a are the attribute values of p and c with respect to a respectively.

4.2 Training

Our model estimates $P(p, k, c, l)$, the probability of seeing (1) the overt pronoun p; (2) the context k surrounding p and its candidate antecedents; (3) a candidate antecedent c of p; and (4) whether c is the correct antecedent of p. Since we estimate this probability from a raw, unannotated corpus, we are effectively treating p, k, and c as observed data and l as hidden data.

Owing to the presence of hidden data, we estimate the model parameters using the EM algorithm. Specifically, we use EM to iteratively estimate the model parameters from data in which each overt pronoun is labeled with the probability it corefers with each of its candidate antecedents and apply the resulting model to re-label each overt pronoun with the probability it corefers with each of its candidate antecedents. Below we describe

[1] Only 8% of the overt pronouns in our corpus, the Chinese portion of the OntoNotes 5.0 corpus, do not have any antecedent in the preceding two sentences.

the details of the E-step and the M-step.

4.2.1 E-Step

The goal of the E-step is to compute $P(l{=}1|p,k,c)$, the probability that a candidate antecedent c is the correct antecedent of p given context k. Assuming that exactly one of the p's candidate antecedents is its correct antecedent, we can rewrite $P(l{=}1|p,k,c)$ as follows:

$$P(l{=}1|p,k,c) = \frac{P(p,k,c,l{=}1)}{\sum_{c' \in C} P(p,k,c',l{=}1)} \quad (1)$$

Applying Chain Rule, we can rewrite $P(p,k,c,l{=}1)$ as follows:

$$\begin{aligned} P(p,k,c,l{=}1) = P(p|k,c,l{=}1) * P(l{=}1|k,c) \\ * P(c|k) * P(k) \end{aligned} \quad (2)$$

Next, given $l = 1$ (i.e., c is the antecedent of p), we assume that we can generate p from c without looking at the context.[2] Then we represent p using its grammatical attributes A. We further assume that p's value with respect to attribute $a \in A$ is independent of the value of each of its remaining attributes given the antecedent's value with respect to a. So we can rewrite $P(p|k,c,l{=}1)$ as follows:

$$\begin{aligned} & P(p|k,c,l{=}1) \approx P(p|c,l{=}1) \\ \approx\ & P(p_{Num}, p_{Gen}, p_{Per}, p_{Ani}|c,l{=}1) \\ \approx\ & \prod_{a \in A} P(p_a|c_a,l{=}1) \end{aligned} \quad (3)$$

Moreover, we assume that (1) given p and c's context, the probability of c being the antecedent of p is not affected by the context of the other candidate antecedents; and (2) k_c is sufficient for determining whether c is the antecedent of p. So,

$$P(l{=}1|k,c) \approx P(l{=}1|k_c,c) \approx P(l{=}1|k_c) \quad (4)$$

Furthermore, we assume that given context k, each candidate antecedent of p is generated with equal probability. In other words,

$$P(c|k) \approx P(c'|k) \ \ \forall\, c, c' \in C \quad (5)$$

Given Equations (2), (3), (4) and (5), we can rewrite $P(l{=}1|p,k,c)$ as:

$$\begin{aligned} P(l{=}1|p,k,c) &= \frac{P(p,k,c,l{=}1)}{\sum_{c' \in C} P(p,k,c',l{=}1)} \\ &\approx \frac{\prod_{a \in A} P(p_a|c_a,l{=}1) * P(l{=}1|k_c)}{\sum_{c' \in C} \prod_{a \in A} P(p_a|c'_a,l{=}1) * P(l{=}1|k_{c'})} \end{aligned} \quad (6)$$

[2] This assumption is reasonable because it is fairly easy to determine which pronoun can be used to refer to a given NP.

As we can see from Equation (6), our model has two groups of parameters, namely $P(p_a|c_a,l{=}1)$ and $P(l{=}1|k_c)$. Since we have four grammatical attributes, $P(p_a|c_a,l{=}1)$ contains four sets of parameters, with one set per attribute. Using Equation (6) and the current parameter estimates, we can compute $P(l{=}1|p,k,c)$.

Two points deserve mention before we describe the M-step. First, we estimate $P(l{=}1|p,k,c)$ from all and only those overt pronouns $p \in PR$ that are surface or deep subjects in their corresponding sentences. This condition is motivated by our observation that 99.56% of the ZPs in our evaluation corpus (i.e., OntoNotes 5.0) are surface or deep subjects. In other words, we impose this condition so that we can focus our efforts on learning a model for resolving overt pronouns that are subjects. This is by no means a limitation of our model: if we were given a corpus in which many ZPs occur as grammatical objects, we could similarly train another model on overt objects. Second, since in the E-step we attempt to probabilistically label every overt pronoun p that satisfies the condition above, our model is effectively making the simplifying assumption that every overt pronoun is anaphoric. This is clearly an overly simplistic assumption. One way to relax this assumption, which we leave as future work, is to first identify those pronouns that are anaphoric and then use EM to estimate the joint probability only from the anaphoric pronouns.

4.2.2 M-Step

Given $P(l{=}1|p,k,c)$, the goal of the M-step is to (re)estimate the model parameters, $P(p_a|c_a,l{=}1)$ and $P(l{=}1|k_c)$, using maximum likelihood estimation. Specifically, $P(p_a|c_a,l{=}1)$ is estimated as follows:

$$P(p_a|c_a,l{=}1) = \frac{Count(p_a,c_a,l{=}1) + \theta}{Count(c_a,l{=}1) + \theta * |a|} \quad (7)$$

where $Count(c_a,l{=}1)$ is the expected number of times c has attribute value c_a when it is the antecedent of p; $|a|$ is the number of possible values of attribute a; θ is the Laplace smoothing parameter, which we set to 1; and $Count(p_a,c_a,l{=}1)$ is the expected number of times p has attribute value p_a when its antecedent c has attribute value c_a. Given attribute values p'_a and c'_a, we compute

$Count(p'_a, c'_a, l{=}1)$ as follows:

$$Count(p'_a, c'_a, l{=}1) = \sum_{p,c:p_a=p'_a,c_a=c'_a} P(l{=}1|p,k,c) \quad (8)$$

Similarly, $P(l{=}1|k_c)$ is estimated as follows:

$$P(l{=}1|k_c) = \frac{Count(k_c, l{=}1) + \theta}{Count(k_c) + \theta * 2} \quad (9)$$

where $Count(k_c)$ is the number of times k_c appears in the training data, and $Count(k_c, l{=}1)$ is the expected number of times k_c is the context surrounding a pronoun and its antecedent c. Given context k'_c, we compute $Count(k'_c, l{=}1)$ as follows:

$$Count(k'_c, l{=}1) = \sum_{k:k_c=k'_c} P(l{=}1|p,k,c) \quad (10)$$

To start the induction process, we initialize all parameters with uniform values. Specifically, $P(p_a|c_a, l{=}1)$ is set to $\frac{1}{|a|}$, and $P(l{=}1|k_c)$ is set to 0.5. Then we iteratively run the E-step and the M-step until convergence.

There are two important questions we have not addressed. First, how can we compute the four attribute values of a candidate antecedent (i.e., c_a for each attribute a), which we need to estimate $P(p_a|c_a, l{=}1)$? Second, what features should we use to represent context k_c, which we need to estimate $P(l{=}1|k_c)$? We defer the discussion of these questions to Sections 5 and 6.

4.3 Inference

After training, we can apply the resulting model to resolve AZPs. Given an AZP z, we determine its antecedent as follows:

$$(\hat{c}, \hat{p}) = \underset{c \in C,\ p \in PR}{\arg\max}\ P(l{=}1|p,k,c) \quad (11)$$

where PR is our set of 10 Chinese overt pronouns and C is the set of candidate antecedents of z. In other words, we apply Formula (11) to each AZP z, searching for the candidate antecedent c and overt pronoun p that maximize $P(l{=}1|p,k,c)$ when p is used to fill the ZP gap left behind by z. The c that results in the maximum probability value over all overt pronouns in PR is chosen as the antecedent of z. In essence, since the model is trained on overt pronouns but is applied to ZPs, we have to exhaustively fill the ZP's gap under consideration with each of the 10 overt pronouns in PR during inference.

Although we can now apply our generative model to resolve AZPs, the resolution procedure can be improved further. The improvement is motivated by a problem we observed previously (Chen and Ng, 2013): an AZP and its closest antecedent can sometimes be far away from each other, thus making it difficult to correctly resolve the AZP. To address this problem, we employ the following resolution procedure in our experiments. Given a test document, we process its AZPs in a left-to-right manner. As soon as we resolve an AZP to a preceding NP c, we fill the corresponding AZP's gap with c. Hence, when we process an AZP z, all of its preceding AZPs in the associated text have been resolved, with their gaps filled by the NPs they are resolved to. To resolve z, we create test instances between z and its candidate antecedents in the same way as described before. The only difference is that the set of candidate antecedents of z may now include those NPs that are used to fill the gaps of the AZPs resolved so far. In other words, this incremental resolution procedure may increase the number of candidate antecedents of each AZP z. Some of these additional candidate antecedents are closer to z than the original candidate antecedents, thereby facilitating the resolution of z. If the model resolves z to the additional candidate antecedent that fills the gap left behind by, say, AZP z', we postprocess the output by resolving z to the NP that z' is resolved to.[3]

5 Attributes of Candidate Antecedents

In this section, we describe how we determine the four grammatical attribute values (NUMBER, GENDER, PERSON and ANIMACY) of a candidate antecedent c, as they are used to represent c when estimating $P(p_a|c_a, l{=}1)$ for each attribute a.

5.1 ANIMACY

We determine the ANIMACY of a candidate antecedent c heuristically. Specifically, we first check the NP type of c. If c is a pronoun, we look up its ANIMACY in Table 1. If c is a named entity, there are two cases to consider: if c is a *person*[4], we label it as *animate*; otherwise, we label it as *inanimate*. If c is a common noun, we look up the ANIMACY of its head noun in an automatically

[3] This postprocessing step is needed because the additional candidate antecedents are only gap fillers.

[4] A detailed description of our named entity recognizer can be found in Chen and Ng (2014).

constructed word list WL. If the head noun is not in WL, we set its ANIMACY to *unknown*.

Our method for constructing WL is motivated by an observation of measure words in Chinese: some of them only modify *inanimate* nouns while others only modify *animate* nouns. For example, the nouns modified by the measure word 张 are always *inanimate*, as in 一张纸 (one piece of paper). On the other hand, the nouns modified by the measure word 位 are always *animate*, as in 一位工人 (one worker).

Given this observation, we first define two lists, M_{ani} and M_{inani}. M_{ani} is a list of measure words that can only modify *animate* nouns. M_{inani} is a list of measure words that can only modify *inanimate* nouns.[5] There exists a special measure word in Chinese, 个, which can be used to modify most of the common nouns regardless of their ANIMACY. As a result, we remove 个 from both lists. After constructing M_{ani} and M_{inani}, we (1) parse the Chinese Gigaword corpus (Parker et al., 2009), which contains 4,370,600 documents, using an efficient dependency parser, ctbparser[6] (Qian et al., 2010), and then (2) collect all pairs of words (m, n), where m is a measure word, n is a common noun, and there is a NMOD dependency relation between m and n. Finally, we determine the ANIMACY of a given common noun n as follows. First, we retrieve all of the pairs containing n. Then, we sum over all occurrences of m in M_{ani} (call the sum C_{ani}), as well as all occurrences of m in M_{inani} (call the sum C_{inani}). If $C_{ani} > C_{inani}$, we label this common noun as *animate*; otherwise, we label it as *inanimate*.

Table 2 shows the learned values of $P(p_{Ani}|c_{Ani}, l{=}1)$. These results are consistent with our intuition: an animate (inanimate) pronoun is more likely to be generated from an animate (inanimate) antecedent than from an inanimate (animate) antecedent. Note that animate pronouns are more likely to be generated than inanimate pronouns regardless of the antecedent's ANIMACY. This can be attributed to the fact that 94.6% of the pronouns in our corpus are animate.

5.2 GENDER

We determine the GENDER of a candidate antecedent c as follows. If c is a pronoun, we look up its GENDER in Table 1. Otherwise, we determine its GENDER based on its ANIMACY. Specifically, if c is *inanimate*, we set its GENDER to *neuter*. Otherwise, we determine its gender by looking up a gender word list constructed by Bergsma and Lin's (2006) approach. If the word is not in the list, we set its GENDER to *masculine* by default.

Antecedent \ Pronoun	animate	inanimate
animate	0.999	0.001
inanimate	0.858	0.142
unknown	0.945	0.055

Table 2: Learned values of $P(p_{Ani}|c_{Ani}, l{=}1)$.

Next, we describe how the aforementioned gender word list is constructed. Following Bergsma and Lin (2006), we define a *dependency path* as the sequence of non-terminal nodes and dependency labels between two potentially coreferent entities in a dependency parse tree. From the parsed Chinese Gigaword corpus, we first collect every dependency path that connects two pronouns. For each path P collected, we compute $CL(P)$, the coreference likelihood of P, as follows:

$$CL(P) = \frac{N_I(P)}{N_I(P) + N_D(P)} \qquad (12)$$

where $N_I(P)$ is the number of times P connects two identical pronouns, and $N_D(P)$ is the number of times it connects two different pronouns. Assuming that two identical pronouns in a sentence are coreferent (Bergsma and Lin, 2006), we can see that the larger a path's CL value is, the more likely it is that the two NPs it connects are coreferent. To ensure that we have dependency paths that are strongly indicative of coreference relations, we consider a dependency path P a *coreferent path* if and only if $CL(P) > 0.8$.

Given these coreferent paths, we can compute the GENDER of a noun n as follows. First, we compute (1) $N_M(n)$, the number of coreferent paths connecting n with a *masculine* pronoun; and (2) $N_F(n)$, the number of coreferent paths connecting n with a *feminine* pronoun. Then, if $N_F(n) > N_M(n)$, we set n's gender to *feminine*; otherwise, we set it to *masculine*.

Table 3 shows the learned values of $P(p_{Gen}|c_{Gen}, l{=}1)$. These results are consistent with our intuition: a pronoun is a lot more likely to be generated from an antecedent with the same GENDER than one with a different GENDER.

[5] We create these two lists with the help of this page: http://chinesenotes.com/ref_measure_words.htm

[6] http://code.google.com/p/ctbparser/

Antecedent \ Pronoun	neuter	feminine	masculine
neuter	0.864	0.018	0.117
feminine	0.065	0.930	0.005
masculine	0.130	0.041	0.828

Table 3: Learned values of $P(p_{Gen}|c_{Gen}, l{=}1)$.

Antecedent \ Pronoun	singular	plural
singular	0.861	0.139
plural	0.26	0.74

Table 4: Learned values of $P(p_{Num}|c_{Num}, l{=}1)$.

5.3 NUMBER

When computing the NUMBER of a candidate antecedent in English, Charniak and Elsner (2009) rely on part-of-speech information. For example, NN and NNP denote singular nouns, whereas NNS and NNPS denote plural nouns. However, Chinese part-of-speech tags do not provide such information. Hence, we need a different method for finding the NUMBER of a candidate antecedent c in Chinese. If c is a pronoun, we look up its NUMBER in Table 1. If c is a named entity, its NUMBER is *singular*. If c is a common noun, we infer its NUMBER from its string: if the string ends with 们 or is modified by a quantity word (e.g., 一些, 许多), c is *plural*; otherwise, c is *singular*.

Table 4 shows the learned values of $P(p_{Num}|c_{Num}, l{=}1)$. These results are consistent with our intuition: a pronoun is more likely to be generated from an antecedent with the same NUMBER than one with a different NUMBER.

5.4 PERSON

Finally, we compute the PERSON of a candidate antecedent c. Similar to Charniak and Elsner (2009), we set 我 (I) and 我们 (we) to *first* person, 你 (singular you) and 你们 (plural you) to *second* person, and everything else to *third* person. We estimate two sets of probabilities $P(p_{Per}|c_{Per}, l{=}1)$, one where p and c are from the same speaker, and the other where they are from different speakers.[7] This is based on our observation that $P(p_{Per}|c_{Per}, l{=}1)$ could be very different in these two cases.

[7]We employ a simple heuristic to identify the speaker of NPs occurring in direct speech: we assume that the speaker is the subject of the speech's reporting verb. So for example, we identify *Jack* as the speaker of *This book* in the sentence *"This book is good," Jack said.*

Antecedent \ Pronoun	first	second	third
first	0.856	0.119	0.025
second	0.219	0.766	0.016
third	0.289	0.077	0.634

Table 5: Learned values of $P(p_{Per}|c_{Per}, l{=}1)$ (same speaker).

Antecedent \ Pronoun	first	second	third
first	0.417	0.525	0.057
second	0.75	0.23	0.02
third	0.437	0.229	0.334

Table 6: Learned values of $P(p_{Per}|c_{Per}, l{=}1)$ (different speakers).

Tables 5 and 6 show the learned values of these two sets of probabilities. These results are consistent with our intuition. In the same-speaker case, a pronoun is a lot more likely to be generated from an antecedent with the same speaker than one with a different speaker. In the different-speaker case, a first (second) person pronoun is most likely to be generated from a second (first) person pronoun.

6 Context Features

To fully specify our model, we need to describe how to represent k_c, which is needed to compute $P(l{=}1|k_c)$. Recall that k_c encodes the context surrounding candidate antecedent c and the associated pronoun p. As described below, we represent k_c using eight features, some of which are motivated by previous work on supervised AZP resolution (e.g., Zhao and Ng (2007), Chen and Ng (2013)). Note that (1) all but feature 1 are computed based on syntactic parse trees, and (2) features 2, 3, 6, and 8 are ternary-valued features.

1. the sentence distance between c and p;
2. whether the node spanning c has an ancestor NP node; if so, whether this NP node is a descendant of c's lowest ancestor IP node;
3. whether the node spanning c has an ancestor VP node; if so, whether this VP node is a descendant of c's lowest ancestor IP node;
4. whether vp has an ancestor NP node, where vp is the VP node spanning the VP that follows p;
5. whether vp has an ancestor VP node;

	Training	Test
Documents	1,391	172
Sentences	36,487	6,083
Words	756,063	110,034
Overt Subject Pronouns	13,418	–
AZPs	–	1,713

Table 7: Statistics on the training and test sets.

6. whether p is the first word of a sentence; if not, whether p is the first word of an IP clause;
7. whether c is a subject whose governing verb is lexically identical to the verb governing p;
8. whether c is the closest candidate antecedent with subject grammatical role and is semantically compatible with p's governing verb; if not, whether c is the first semantically compatible candidate antecedent[8].

Our approach to determine semantic compatibility (in feature 8) resembles Kehler et al.'s (2004) and Yang et al.'s (2005) methods for computing selectional preferences. Specifically, for each verb and each noun that serves as a subject in Chinese Gigaword, we compute their mutual information (MI). Now, given a pronoun p and a candidate antecedent c in the training/test corpus, we retrieve the MI value of c and p's governing verb. We then consider them semantically compatible if and only if their MI value is greater than zero.

7 Evaluation

7.1 Experimental Setup

Datasets. We employ the Chinese portion of the OntoNotes 5.0 corpus that was used in the official CoNLL-2012 shared task (Pradhan et al., 2012). In the CoNLL-2012 data, the training set and development set contain ZP coreference annotations, but the test set does not. Therefore, we train our models on the training set and perform evaluation on the development set. Statistics on the datasets are shown in Table 7. The documents in these datasets come from six sources, namely Broadcast News (BN), Newswire (NW), Broadcast Conversation (BC), Telephone Conversation (TC), Web Blog (WB) and Magazine (MZ).

[8] We sort the candidate antecedents of p as follows. We first consider the subject candidate antecedents in the same sentence as p from right to left, then the other candidate antecedents in the same sentence from right to left. Next, we consider the candidate antecedents in the previous sentence, also preferring candidates that are subjects, but in left-to-right order. Finally, we consider the candidate antecedents two sentences back, following the subject-first, left-to-right order.

Evaluation measures. We express the results of ZP resolution in terms of recall (R), precision (P) and F-score (F).

Evaluation settings. Following Chen and Ng (2013), we evaluate our model in three settings. In Setting 1, we assume the availability of gold syntactic parse trees and gold AZPs. In Setting 2, we employ gold syntactic parse trees and system (i.e., automatically identified) AZPs. Finally, in Setting 3, we employ system syntactic parse trees and system AZPs. The gold and system syntactic parse trees, as well as the gold AZPs, are obtained from the CoNLL-2012 shared task dataset, while the system AZPs are identified by the rule-based approach described in the Appendix.[9] Since our AZP identification approach does not rely on any labeled data, we are effectively evaluating an end-to-end unsupervised AZP resolver in Setting 3.

7.2 Results

Baseline systems. We employ seven resolvers as baseline systems. To gauge the difficulty of the task, we employ four simple rule-based resolvers, which resolve an AZP z to (1) the candidate antecedent closest to z (Baseline 1); (2) the subject NP closest to z (Baseline 2); (3) the closest candidate antecedent that is semantically compatible with z (Baseline 3); and (4) the first candidate antecedent that is semantically compatible with z, where the candidate antecedents are visited according to the order described in Footnote 8 (Baseline 4). These four baselines allow us to study the role of (1) recency, (2) salience, (3) recency combined with semantic compatibility, and (4) salience combined with semantic compatibility in AZP resolution respectively. The remaining three baselines are state-of-the-art supervised AZP resolvers, which include our own resolver (Chen and Ng, 2013) as well as our re-implementations of Zhao and Ng's (2007) resolver and Kong and Zhou's (2010) resolver.

The test set results of these seven baseline resolvers when evaluated under the three aforementioned evaluation settings are shown in Table 8. The system AZPs employed by the rule-based resolvers are obtained using our rule-based

[9] One may wonder why we do not train a supervised system for identifying AZPs and instead experiment with a rule-based AZP identification system. The reason is that employing labeled data defeats the whole purpose of having an unsupervised AZP resolution model: if annotated data is available for training an AZP identification system, the same data can be used to train an AZP resolution system.

Baseline	Setting 1: Gold Parses, Gold AZPs			Setting 2: Gold Parses, System AZPs			Setting 3: System Parses, System AZPs		
	R	P	F	R	P	F	R	P	F
Selecting closest candidate antecedent	25.0	25.2	25.1	18.3	10.8	13.6	10.3	6.7	8.1
Selecting closest subject	42.0	43.6	42.8	31.8	19.2	23.9	18.0	11.9	14.4
Selecting closest semantically compatible candidate antecedent	28.5	28.8	28.7	20.5	12.2	15.3	11.7	7.6	9.2
Selecting first semantically compatible candidate antecedent	45.2	45.7	45.5	33.6	20.0	25.1	18.9	12.3	14.9
Zhao and Ng (2007)	41.5	41.5	41.5	22.4	24.4	23.3	12.7	14.2	13.4
Kong and Zhou (2010)	44.9	44.9	44.9	33.0	19.3	24.4	18.7	11.9	14.5
Chen and Ng (2013)	47.7	47.7	47.7	25.3	27.6	26.4	14.9	16.7	15.7

Table 8: AZP resolution results of the baseline systems on the test set.

Source	Setting 1: Gold Parses, Gold AZPs						Setting 2: Gold Parses, System AZPs						Setting 3: System Parses, System AZPs					
	Best Baseline			Our Model			Best Baseline			Our Model			Best Baseline			Our Model		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Overall	47.7	47.7	47.7	47.5	47.9	47.7	25.3	27.6	26.4	35.4	21.0	26.4	14.9	16.7	15.7	19.9	12.9	15.7
NW	38.1	38.1	38.1	41.7	41.7	41.7	15.5	21.7	18.1	29.8	24.8	27.0	6.0	12.2	8.0	11.9	13.0	12.4
MZ	34.6	34.6	34.6	34.0	34.2	34.1	18.5	19.6	19.0	24.1	14.5	18.1	6.2	9.4	7.5	6.2	5.2	5.7
WB	46.1	46.1	46.1	47.9	47.9	47.9	21.8	22.0	21.8	37.3	18.7	24.9	8.5	11.4	9.7	19.0	11.3	14.2
BN	47.2	47.2	47.2	52.8	52.8	52.8	21.8	33.2	26.3	31.5	28.1	29.7	14.6	26.3	18.8	18.2	19.5	18.8
BC	52.7	52.7	52.7	49.8	50.3	50.0	23.3	30.7	26.5	38.0	21.0	27.0	12.7	16.2	14.3	20.6	12.4	15.5
TC	51.2	51.2	51.2	45.2	46.7	46.0	43.1	28.2	34.1	42.4	20.3	27.4	33.2	17.1	22.5	32.2	13.3	18.8

Table 9: AZP resolution results of the best baseline and our unsupervised model on the test set.

AZP identification system. On the other hand, since our supervised resolvers are meant to be re-implementations of existing resolvers, we follow previous work and let them employ a supervised AZP identification system. In particular, we employ the one described in Chen and Ng (2013).

Several observations can be made about these results. First, among the rule-based resolvers, Baseline 4 achieves the best performance, outperforming Baselines 1, 2, and 3 by 12.9%, 1.5%, and 10.8% in F-score respectively when averaged over the three evaluation settings. From their relative performance, which remains the same in the three settings, we can conclude that as far as AZP resolution is concerned, (1) salience plays a greater role than recency; and (2) semantic compatibility is useful. Second, among the supervised baselines, our supervised resolver (Chen and Ng, 2013) achieves the best performance, outperforming Zhao and Ng's resolver and Kong and Zhou's resolver by 3.9% and 2.0% in F-score respectively when averaged over the three evaluation settings. Finally, comparing the rule-based resolvers and the learning-based resolvers, we can see that the best rule-based baseline (Baseline 4) performs even better than Zhao and Ng's resolver and Kong and Zhou's resolver.

In the rest of this subsection, we will compare our unsupervised model against the best baseline, Chen and Ng's (2013) supervised resolver.

Our model. Results of the best baseline and our model on the entire test set and each of the six sources are shown in Table 9. As we can see, our model achieves the same overall F-score as the best baseline under all three settings, despite the fact that it is unsupervised. In fact, our model even outperforms the best baseline on NW, WB and BN in Setting 1, NW, WB, BN and BC in Setting 2, and NW, WB and BC in Setting 3.

It is worth mentioning that while the two resolvers achieved the same overall performance, their outputs differ a lot from each other. Specifically, the two models only agree on the antecedents of 55% of the AZPs in Setting 1.[10]

7.3 Ablation Experiments

Impact of $P(p_a|c_a, l{=}1)$ and $P(l{=}1|k_c)$. Recall that our model is composed of five probability terms, $P(p_a|c_a, l{=}1)$ for each of the four grammatical attributes and $P(l{=}1|k_c)$, the context probability. To investigate the contribution of context and each attribute to overall performance, we conduct ablation experiments. Specifically, in each ablation experiment, we remove exactly one probability term from the model and retrain it.

[10]Note that it is difficult to directly compare the outputs produced under Settings 2 and 3: the AZPs identified by the best baseline are quite different from those identified by our rule-based system, as can be inferred from the AZP identification results in Table 12.

System	Setting 1			Setting 3		
	R	P	F	R	P	F
Full model	47.5	47.9	47.7	19.9	12.9	15.7
− NUMBER	47.5	47.9	47.7	19.7	12.8	15.5
− GENDER	44.5	45.0	44.7	19.2	12.5	15.1
− PERSON	45.2	45.6	45.4	19.1	12.4	15.1
− ANIMACY	45.1	45.5	45.3	19.1	12.4	15.1
− Context Features	32.9	33.1	33.0	15.2	9.8	11.9

Table 10: Probability term ablation results.

Ablation results under Settings 1 and 3 are shown in Table 10. As we can see, under Setting 1, after NUMBER is ablated, performance does not drop. We attribute this to the fact that almost all candidate antecedents are *singular*. On the other hand, when we ablate any of the remaining three attributes, performance drops significantly by 2.3−3.0% in overall F-score.[11] Similar trends can be observed with respect to Setting 3: after NUMBER is ablated, performance only decreases by 0.2%, while ablating any of the other three attributes results in a drop of 0.6%.

Results after ablating context are shown in the last row of Table 10. As we can see, the F-score drops significantly by 14.7% and 3.8% under Settings 1 and 3 respectively. These results illustrate the importance of context features in our model.

Context feature ablation. Recall that we employed eight context features to encode the relationship between a pronoun and a candidate antecedent. To determine the relative contribution of these eight features to overall performance, we conduct ablation experiments under Settings 1 and 3. In these ablation experiments, all four grammatical attributes are retained in the model.

Ablation results are shown in rows 2−9 of Table 11. To facilitate comparison, the F-score of the model in which all eight context features are used is shown in row 1. As we can see, feature 8 (the rule-based feature) is the most useful feature: its removal causes the F-scores of our resolver to drop significantly by 6.4% under Setting 1 and 1.5% under Setting 3.

7.4 Error Analysis

To gain additional insights into our full model, we examine its major sources of error below. To focus on errors attributable to AZP resolution, we analyze our full model under Setting 1.

Specifically, we randomly select 100 AZPs that our model incorrectly resolves under Setting 1.

[11] All significance tests are paired t-tests, with $p < 0.05$.

System	Setting 1			Setting 3		
	R	P	F	R	P	F
Full model	47.5	47.9	47.7	19.9	12.9	15.7
− Feature 1	46.1	46.5	46.3	19.4	12.6	15.3
− Feature 2	46.5	46.9	46.7	19.4	12.6	15.3
− Feature 3	45.3	45.7	45.5	19.1	12.4	15.1
− Feature 4	47.4	47.8	47.6	20.1	13.0	15.8
− Feature 5	47.4	47.8	47.6	19.7	12.8	15.5
− Feature 6	47.1	47.5	47.3	19.6	12.7	15.4
− Feature 7	47.1	47.5	47.3	20.1	13.0	15.8
− Feature 8	41.2	41.6	41.4	18.0	11.8	14.2

Table 11: Context feature ablation results.

We found that 17 errors are attributable to discourse disfluency, lack of background knowledge and subject detection, while the remaining 83 errors can be divided into three types:

Failure to recognize the topics of a document. Our model incorrectly resolves 32 AZPs that are coreferent with NPs corresponding to the topics of the associated documents. Consider the following example:

[八里乡] 位于台北盆地西北端。行政区隶属于台北县，*pro* 为台北县廿九个乡镇市之一。
([Bali Town] is located in the Northwest of Taipei Basin. Its administrative area is affiliated with Taipei County, *pro* is one of Taipei County's 29 towns and cities.)[12]

The model incorrectly resolves the AZP *pro* to 行政区 (Its administrative area). The reason is that the correct antecedent, 八里乡 (Bali Town), is far from *pro*: there are five candidate antecedents between *pro* and 八里乡 (Bali Town). Note, however, that it is easy for a human to resolve *pro* to 八里乡 (Bali Town) because the whole passage is discussing 八里乡 (Bali Town). Hence, to correctly handle such cases, one may construct a topic model over the passage and assign each candidate antecedent a prior probability so that the resulting system favors the selection of candidates representing the topics as antecedents.

Errors in computing semantic compatibility. This type of error contributes to 28 of the incorrectly resolved AZPs. When computing semantic compatibility in our model, we only consider the mutual information between a candidate antecedent and the pronoun's governing verb, but in some cases, additional context needs to be taken into account. Consider the following example:

[12] The pronoun *Its* in the phrase *Its administrative area* is inserted into the English translation for the sake of grammaticality and correct understanding of the sentence. The corresponding Chinese phrase does not contain any pronoun.

[一支海军陆战队] 杀死了约 [24 名手无寸铁的伊拉克人]，*pro* 包括妇女和六名儿童。
([Marines] killed about [24 unarmed Iraqis], *pro* include women and six children.)

There are two candidate antecedents in this example, 一支海军陆战队 (Marines) and 24 名手无寸铁的伊拉克人 (24 unarmed Iraqis), which we denote as c_1 and c_2 respectively. The correct antecedent of *pro* is c_2 , while our model wrongly resolves *pro* to c_1. Note that both c_1 and c_2 are compatible with the AZP's governing verb 包括 (include). However, if the object of the governing verb, i.e., 妇女和六名儿童 (women and six children), were also considered, the model could determine that c_1 is not compatible with the object while c_2 is, and then correctly resolve *pro* to c_2.

Failure to recognize and exploit semantically similar sentences. This type of error contributes to 23 wrongly resolved AZPs. Recall that an AZP is omitted for brevity, so the sentence it appears in often expresses similar meaning to an earlier sentence. However, our model fails to handle such cases. Consider the following example:

[指挥部和突进的部队] 之间也会失去联络。..... *pro* 就联络不上了。
([The command and the onrush of troops] lost connection with each other. ... *pro* cannot connect with each other.)

The above example shows two sentences that are separated by some other sentences. The AZP under consideration is in the last sentence, while the first sentence contains the correct antecedent 指挥部和突进的部队 (the command and the onrush troops), denoted as c_1. Our model fails to resolve *pro* to c_1, because there are many competing candidate antecedents between c_1 and AZP. However, if our model were aware of the similarity between the constructions appearing after c_1 and *pro*, i.e., 之间也会失去联络 (lost connection with each other) and 就联络不上了 (cannot connect with each other), then it might be able to correctly resolve the AZP.

8 Conclusion

We proposed an unsupervised model for Chinese zero pronoun resolution, investigating the novel hypothesis that an unsupervised probabilistic resolver trained on overt pronouns can be applied to resolve ZPs. To our knowledge, this is the first unsupervised probabilistic model for this task. Experiments on the OntoNotes 5.0 corpus showed that our unsupervised model rivaled its state-of-the-art supervised counterparts in performance.

Acknowledgments

We thank the three anonymous reviewers for their detailed comments. This work was supported in part by NSF Grants IIS-1147644 and IIS-1219142.

References

Shane Bergsma and Dekang Lin. 2006. Bootstrapping path-based pronoun resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 33--40.

Eugene Charniak and Micha Elsner. 2009. EM works for pronoun anaphora resolution. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 148--156.

Chen Chen and Vincent Ng. 2013. Chinese zero pronoun resolution: Some recent advances. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1360--1365.

Chen Chen and Vincent Ng. 2014. SinoCoreferencer: An end-to-end Chinese event coreference resolver. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, pages 4532--4538.

Colin Cherry and Shane Bergsma. 2005. An expectation maximization approach to pronoun resolution. In *Proceedings of the Ninth Conference on Natural Language Learning*, pages 88--95.

Susan Converse. 2006. *Pronominal Anaphora Resolution in Chinese*. Ph.D. thesis, University of Pennsylvania.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1--38.

Antonio Ferrández and Jesús Peral. 2000. A computational approach to zero-pronouns in Spanish. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 166--172.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203--226.

Na-Rae Han. 2006. *Korean Zero Pronouns: Analysis and Resolution*. Ph.D. thesis, University of Pennsylvania.

Jerry Hobbs. 1978. Resolving pronoun references. *Lingua*, 44:311--338.

Ryu Iida and Massimo Poesio. 2011. A cross-lingual ILP solution to zero anaphora resolution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 804--813.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploiting syntactic patterns as clues in zero-anaphora resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 625--632.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007. Zero-anaphora resolution by learning rich syntactic pattern features. *ACM Transactions on Asian Language Information Processing*, 6(4).

Kenji Imamura, Kuniko Saito, and Tomoko Izumi. 2009. Discriminative approach to predicate-argument structure analysis with zero-anaphora resolution. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 85--88.

Hideki Isozaki and Tsutomu Hirao. 2003. Japanese zero pronoun resolution based on ranking rules and machine learning. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 184--191.

Andrew Kehler, Douglas Appelt, Lara Taylor, and Aleksandr Simma. 2004. The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proceedings of 2004 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 289--296.

Fang Kong and GuoDong Zhou. 2010. A tree kernel-based unified framework for Chinese zero anaphora resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 882--891.

Robert Parker, David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. 2009. Chinese Gigaword fourth edition. Linguistic Data Consortium. Philadelphia, PA.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning: Shared Task*, pages 1--40.

Xian Qian, Qi Zhang, Xuangjing Huang, and Lide Wu. 2010. 2d trie for fast parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 904--912.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to Japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 758--766.

Kazuhiro Seki, Atsushi Fujii, and Tetsuya Ishikawa. 2002. A probabilistic method for analyzing Japanese anaphora integrating zero pronoun detection and resolution. In *Proceedings of the 19th International Conference on Computational Linguistics*.

Xiaofeng Yang, Jian Su, and Chew Lim Tan. 2005. Improving pronoun resolution using statistics-based semantic compatibility information. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 165--172.

Ching-Long Yeh and Yi-Chun Chen. 2007. Zero anaphora resolution in Chinese with shallow parsing. *Journal of Chinese Language and Computing*, 17(1):41--56.

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of Chinese zero pronouns: A machine learning approach. In *Proceedings of the 2007 Joint Conference on Empirical Methods on Natural Language Processing and Computational Natural Language Learning*, pages 541--550.

GuoDong Zhou, Fang Kong, and Qiaoming Zhu. 2008. Context-sensitive convolution tree kernel for pronoun resolution. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing*, pages 25--31.

Appendix: Automatic AZP Identification

Our automatic AZP identification system employs an ordered set of rules. The first rule is a positive rule that aims to extract as many *candidate* AZPs as possible. It is followed by seven negative rules that aim to improve precision by filtering out erroneous candidate AZPs. Below we first describe the rules and then evaluate this rule-based system.

Rule 1. Add candidate AZP z if it occurs before the leftmost word spanned by a VP node vp.

Rule 2. Remove z if its associated vp is in a coordinate structure or modified by an adverbial node.

Rule 3. Remove z if the parent of its associated vp node is not an IP node.

Rule 4. Remove z if its associated vp has a NP or QP node as an ancestor.

Rule 5. Remove z if one of the left sibling nodes of vp is NP, QP, IP or ICP.

Rule 6. Remove z if (1) z does not begin a sentence, (2) the highest node whose spanning word sequence ends with the left non-comma neighbor word of z is either NP, QP or IP, and (3) the parent of this node is VP.

Systems	Gold Parses			System Parses		
	R	P	F	R	P	F
Rule-based	72.4	42.3	53.4	42.3	26.8	32.8
Supervised	50.6	55.1	52.8	30.8	34.4	32.5

Table 12: AZP identification results on the test set.

Rule 7. Remove z if vp's lowest IP ancestor has (1) a VP node as its parent and (2) a VV node as its left sibling.

Rule 8. Remove z if it begins a document.

To gauge the performance of our rule-based AZP identification system, we compare it with our supervised AZP identification system (Chen and Ng, 2013). Results of the two systems on our test set are shown in Table 12. As we can see, the F-scores achieved by the rule-based system is comparable to those of the supervised system.

Unsupervised Sentence Enhancement for Automatic Summarization

Jackie Chi Kit Cheung
University of Toronto
10 King's College Rd., Room 3302
Toronto, ON, Canada M5S 3G4
jcheung@cs.toronto.edu

Gerald Penn
University of Toronto
10 King's College Rd., Room 3302
Toronto, ON, Canada M5S 3G4
gpenn@cs.toronto.edu

Abstract

We present *sentence enhancement* as a novel technique for text-to-text generation in abstractive summarization. Compared to extraction or previous approaches to sentence fusion, sentence enhancement increases the range of possible summary sentences by allowing the combination of dependency subtrees from any sentence from the source text. Our experiments indicate that our approach yields summary sentences that are competitive with a sentence fusion baseline in terms of content quality, but better in terms of grammaticality, and that the benefit of sentence enhancement relies crucially on an event coreference resolution algorithm using distributional semantics. We also consider how text-to-text generation approaches to summarization can be extended beyond the source text by examining how human summary writers incorporate source-text-external elements into their summary sentences.

1 Introduction

Sentence fusion is the technique of merging several input sentences into one output sentence while retaining the important content (Barzilay and McKeown, 2005; Filippova and Strube, 2008; Thadani and McKeown, 2013). For example, the input sentences in Figure 1 may be fused into one output sentence.

As a text-to-text generation technique, sentence fusion is attractive because it provides an avenue for moving beyond sentence extraction in automatic summarization, while not requiring deep semantic analysis beyond, say, a dependency parser and lexical semantic resources.

Input: *Bil Mar Foods Co., a meat processor owned by Sara Lee, announced a recall of certain* ***lots of hot dogs and packaged meat****.*
Input: ***The outbreak led to the recall on Tuesday of*** *15 million pounds of hot dogs and cold cuts* ***produced at the Bil Mar Foods plant.***
Output: *The outbreak led to the recall on Tuesday of lots of hot dogs and packaged meats produced at the Bil Mar Foods plant.*

Figure 1: An example of fusing two input sentences into an output sentence. The sections of the input sentences that are retained in the output are shown in bold.

The overall trajectory pursued in the field can be characterized as a move away from local contexts relying heavily on the original source text towards more global contexts involving reformulation of the text. Whereas sentence extraction and sentence compression (Knight and Marcu, 2000, for example) involve taking one sentence and perhaps removing parts of it, traditional sentence fusion involves reformulating a small number of relatively similar sentences in order to take the union or intersection of the information present therein.

In this paper, we move further along this path in the following ways. First, we present **sentence enhancement** as a novel technique which extends sentence fusion by combining the subtrees of many sentences into the output sentence, rather than just a few. Doing so allows relevant information from sentences that are not similar to the original input sentences to be added during fusion. As

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 775–786,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Source text: *This fact has been underscored in the last few months by two unexpected outbreaks* ***of food-borne illness****.*

Output: *The outbreak* ***of food-borne illness*** *led to the recall on Tuesday of lots of hot dogs and meats produced at the Bil Mar Foods plant.*

Figure 2: An example of sentence enhancement, in which parts of dissimilar sentences are incorporated into the output sentence.

shown in Figure 2, the phrase *of food-borne illness* can be added to the previous output sentence, despite originating in a source text sentence that is quite different overall.

Elsner and Santhanam (2011) proposed a supervised method to fuse disparate sentences, which takes as input a small number of sentences with compatible information that have been manually identified by editors of articles. By contrast, our algorithm is unsupervised, and tackles the problem of identifying compatible event mergers in the entire source text using an event coreference module. Our method outperforms a previous syntax-based sentence fusion baseline on measures of summary content quality and grammaticality.

Second, we analyze how text-to-text generation systems may make use of text that is not in the source text itself, but in articles on a related topic in the same domain. By examining the parts of human-written summaries that are not found in the source text, we find that using in-domain text allows summary writers to more precisely express some target semantic content, but that more sophisticated computational semantic techniques will be required to enable automatic systems to likewise do so.

A more general argument of this paper is that the apparent dichotomy between text-to-text generation and semantics-to-text generation can be resolved by viewing them simply as having different starting points towards the same end goal of precise and wide-coverage NLG. The statistical generation techniques developed by the text-to-text generation community have been successful in many domains. Yet the results of our experiments and studies demonstrate the following: as text-to-text generation techniques move beyond using local contexts towards more dramatic reformulations of the kind that human writers perform, more semantic analysis will be needed in order to ensure that the reformulations preserve the inferences that can be drawn from the input text.

2 Related Work

A relatively large body of work exists in sentence compression (Knight and Marcu, 2000; McDonald, 2006; Galley and McKeown, 2007; Cohn and Lapata, 2008; Clarke and Lapata, 2008, inter alia), and sentence fusion (Barzilay and McKeown, 2005; Marsi and Krahmer, 2005; Filippova and Strube, 2008; Filippova, 2010; Thadani and McKeown, 2013). Unlike this work, our sentence enhancement algorithm considers the entire source text and is not limited to the initial input sentences. Few previous papers focus on combining the content of diverse sentences into one output sentence. Wan et al. (2008) propose sentence augmentation by identifying "seed" words in a single original sentence, then adding information from auxiliary sentences based on word co-occurrence counts. Elsner and Santhanam (2011) investigate the idea of fusing disparate sentences with a supervised algorithm, as discussed above.

Previous studies on cut-and-paste summarization (Jing and McKeown, 2000; Saggion and Lapalme, 2002) investigate the operations that human summarizers perform on the source text in order to produce the summary text. Our previous work argued that current extractive systems rely too heavily on notions of information centrality (Cheung and Penn, 2013). This paper extends this work by identifying specific linguistic factors correlated with the use of source-text-external elements.

3 A Sentence Enhancement Algorithm

The basic steps in our sentence expansion algorithm are as follows: (1) clustering to identify initial input sentences, (2) sentence graph creation, (3) sentence graph expansion, (4) tree generation, and (5) linearization.

At a high level, our method for sentence enhancement is inspired by the syntactic sentence fusion approach of Filippova and Strube (2008) (henceforth, F&S) originally developed for German, in that it operates over the dependency parses of a small number of input sentences to produce an output sentence which fuses parts of the in-

put sentences. We adopt the same assumption as F&S that these initial **core sentences** have a high degree of similarity with each other, and should form the core of a new sentence to be generated (Step 1). While fusion from highly disparate input sentences is possible, Elsner and Santhanam (2011) showed how difficult it is to do so correctly, even where such cases are manually identified. We thus aim for a more targeted type of fusion initially. Next, the dependency trees of the core sentences are fused into an intermediate **sentence graph** (Step 2), a directed acyclic graph from which the final sentence will be generated (Steps 4 and 5). We will compare against our implementation of F&S, adapted to English.

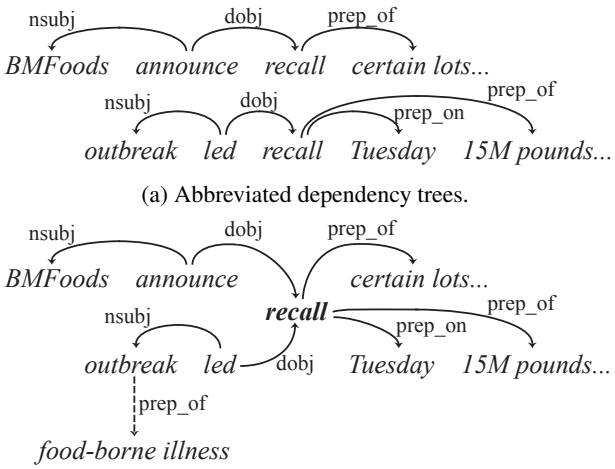
However, unlike F&S or other previous approaches to sentence fusion, the sentence enhancement algorithm may also avail itself of the dependency parses of all of the other sentences in the source text, which expands the range of possible sentences that may be produced. This is accomplished by expanding the sentence graph with parts of these sentences (Step 3). One important issue here is that the expansion must be modulated by an event coreference component to ensure that the merging of information from different points in the source text is valid and does not result in incorrect or nonsensical inferences.

3.1 Core sentence identification

To generate the core sentence clusters, we first identify clusters of similar sentences, then rank the clusters according to their salience. The top cluster in the source text is then selected to be the input to the sentence fusion algorithms.

Sentence alignment is performed by complete-link agglomerative clustering, which requires a measure of similarity between sentences and a stopping criterion. We define the similarity between two sentences to be the standard cosine similarity between the lemmata of the sentences, weighted by IDF and excluding stopwords, and clustering is run until a similarity threshold of 0.5 is reached. Since complete-link clustering prefers small coherent clusters and we select the top-scoring cluster in each document collection, the method is somewhat robust to different choices of the stopping threshold.

The clusters are scored according to the signature term method of Lin and Hovy (2000), which assigns an importance score to each term according to how much more often it appears in the source text compared to some irrelevant background text using a log likelihood ratio. Specifically, the score of a cluster is equal to the sum of the importance scores of the set of lemmata in the cluster.



(a) Abbreviated dependency trees.

(b) Sentence graph after merging the nodes with lemma *recall* (in bold), and expanding the node *outbreak* (dashed outgoing edge).

Figure 3: An example of the input dependency trees for sentence graph creation and expansion, using the input sentences of Figure 1.

3.2 Sentence graph creation

After core sentence identification, the next step is to align the nodes of the dependency trees of the core input sentences in order to create the initial sentence graph. The input to this step is the collapsed dependency tree representations of the core sentences produced by the Stanford parser[1]. In this representation, preposition nodes are collapsed into the label of the dependency edge between the functor of the prepositional phrase and the prepositional object. Chains of conjuncts are also split, and each argument is attached to the parent. In addition, auxiliary verbs, negation particles, and noun-phrase-internal elements[2] are collapsed into their parent nodes. Figure 3a shows the abbreviated dependency representations of the input sentences from Figure 1.

Then, a sentence graph is created by merging nodes that share a common lemma and part-of-

[1] As part of the CoreNLP suite: `http://nlp.stanford.edu/software/corenlp.shtml`

[2] As indicated by the dependency edge label *nn*.

speech tag. In addition, we allow synonyms to be merged, defined as being in the same WordNet synset. Merging is blocked if the word is a stop word, which includes function words as well as a number of very common verbs (e.g., *be*, *have*, *do*). Throughout the sentence graph creation and expansion process, the algorithm disallows the addition of edges that would result in a cycle in the graph.

3.3 Sentence graph expansion

The initial sentence graph is expanded by merging in subtrees from dependency parses of non-core sentences drawn from the source text. First, expansion candidates are identified for each node in the sentence graph by finding all of the dependency edges in the source text from non-core sentences in which the governor of the edge shares the same lemma and POS tag as the node in the sentence graph.

Then, these candidate edges are pruned according to two heuristics. The first is to keep only one candidate edge of each dependency relation type according to the edge that has the highest informativeness score (Section 3.4.1), with ties being broken according to which edge has a subtree with a fewer number of nodes. The second is to perform event coreference in order to prune away those candidate edges which are unlikely to be describing the same event as the core sentences, as explained in the next section. Finally, any remaining candidate edges are fused into the sentence graph, and the subtree rooted at the dependent of the candidate edge is added to the sentence graph as well. See Figure 3b for an example of sentence graph creation and expansion.

3.3.1 Event coreference

One problem of sentence fusion is that the different inputs of the fusion may not refer to the same event, resulting in an incorrect merging of information, as would be the case in the following example:

S1: *Officers pled not guilty but **risked** 25 years to life.*

S2: *Officers recklessly engaged in conduct which seriously **risked** the lives of others.*

Here, the first usage of *risk* refers to the potential sentence imposed if the officers are convicted in a trial, whereas the second refers to the potential harm caused by the officer.

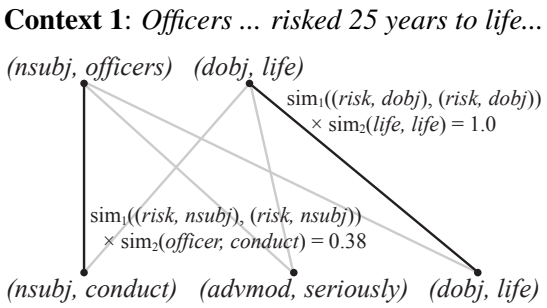


Figure 4: Event coreference resolution as a maximum-weight bipartite graph matching problem. All the nodes share the predicate *risk*.

In order to ensure that sentence enhancement does not lead to the merging of such incompatible events, we designed a simple method to approximate event coreference resolution that does not require event coreference labels. This method is based on the intuition that different mentions of an event should contain many of the same participants. Thus, by measuring the similarity of the arguments and the syntactic contexts between the node in the sentence graph and the candidate edge, we can have a measure of the likelihood that they refer to the same event.

We would be interested in integrating existing event coreference resolution systems into this step in the future, such as the unsupervised method of Bejan and Harabagiu (2010). Existing event coreference systems tend to focus on cases with different heads (e.g., *X kicked Y*, then *Y was injured*), which could increase the possibilities for sentence enhancement, if the event coreference module is sufficiently accurate. However, since our method currently only merges identical heads, we require a more fine-grained method based on distributional measures of similarity.

We measure the similarity of these syntactic contexts by aligning the arguments in the syntactic contexts and computing the similarity of the aligned arguments. These problems can be jointly solved as a maximum-weight bipartite graph matching problem (Figure 4). Formally, let a syntactic context be a list of dependency triples (h, r, a), consisting of a governor or head node h and a dependent argument a in the dependency relation r, where head node h is fixed across each

element of the list. Then, each of the two input syntactic contexts forms one of the two disjoint sets in a complete weighted bipartite graph where each node corresponds to one dependency triple. We define the edge weights according to the similarities of the edge's incident nodes; i.e., between two dependency triples (h_1, r_1, a_1) and (h_2, r_2, a_2). We also decompose the similarity into the similarities between the head and relation types ((h_1, r_1) and (h_2, r_2)), and between the arguments (a_1 and a_2). The edge weight function is thus:

$$\text{sim}((h_1, r_1, a_1), (h_2, r_2, a_2)) = \text{sim}_1((h_1, r_1), (h_2, r_2)) \times \text{sim}_2(a_1, a_2), \quad (1)$$

where sim_1 and sim_2 are binary functions that represent the similarities between governor-relation pairs and dependents, respectively. We train models of distributional semantics using a large background corpus; namely, the Annotated Gigaword corpus (Napoles et al., 2012). For sim_1, we create a vector of counts of the arguments that are seen filling each (h, r) pair, and define the similarity between two such pairs to be the cosine similarity between their argument vectors. For sim_2, we create a basic vector-space representation of a word d according to words that are found in the context of word d within a five-word context window, and likewise compute the cosine similarity between the word vectors. These methods of computing distributional similarity are well attested in lexical semantics for measuring the relatedness of words and syntactic structures (Turney and Pantel, 2010), and similar methods have been applied in text-to-text generation by Ganitkevitch et al. (2012), though the focus of that work is to use paraphrase information thus learned to improve sentence compression.

The resulting graph matching problem is solved using the NetworkX package for Python[3]. The final similarity score is an average of the similarity scores from Equation 1 that participate in the selected matching, weighted by the product of the IDF scores of the dependent nodes of each edge. This final score is used as a threshold that candidate contexts from the source text must meet in order to be eligible for being merged into the sentence graph. This threshold was tuned by cross-validation, and can remain constant, although retuning to different domains (a weakly supervised alternative) is likely to be beneficial.

[3] http://networkx.github.io/

3.4 Tree generation

The next major step of the algorithm is to extract an output dependency tree from the expanded sentence graph. We formulate this as an integer linear program, in which variables correspond to edges of the sentence graph, and a solution to the linear program determines the structure of an output dependency tree. We use ILOG CPLEX to solve all of the integer linear programs in our experiments.

A good dependency tree must at once express the salient or important information present in the input text as well as be grammatically correct and of a manageable length. These desiderata are encoded into the linear program as constraints or as part of the objective function.

3.4.1 Objective function

We designed an objective function that considers the importance of the words and syntactic relations that are selected as well as accounts for redundancy in the output sentence. Let X be the set of variables in the program, and let each variable in X take the form $x_{h,r,a}$, a binary variable that represents whether an edge in the sentence graph from a head node with lemma h to an argument with lemma a in relation r is selected. For a lexicon Σ, our objective function is:

$$\max \sum_{w \in \Sigma} \max_{x_{h,r,a} \in X \, s.t. \, a = w} (x_{h,r,w} \cdot P(r|h) \cdot I(w)), \quad (2)$$

where $P(r|h)$ is the probability that head h projects the dependency relation r, and $I(w)$ is the informativeness score for word w as defined by Clarke and Lapata (2008). This formulation encourages the selection of words that are informative according to $I(w)$ and syntactic relations that are probable. The inner $\max$ function for each w in the lexicon encourages non-redundancy, as each word may only contribute once to the objective value. This function can be rewritten into a form compatible with a standard linear program by the addition of auxiliary variables and constraints. For more details of how this and other aspects of the linear program are implemented, see the supplementary document.

3.4.2 Constraints

Well-formedness constraints, taken directly from F&S, ensure that the set of selected edges pro-

duces a tree. Another constraint limits the number of content nodes in the tree to 11, which corresponds to the average number of content nodes in human-written summary sentences in the data set. Syntactic constraints aim to ensure grammaticality of the output sentence. In addition to the constraint proposed by F&S regarding subordinating conjunctions, we propose two other ones. The first ensures that a nominal or adjectival predicate must be selected with a copular construction at the top level of a non-finite clause. The second ensures that transitive verbs retain both of their complements in the output[4]. Semantic constraints ensure that only noun phrases of sufficiently high similarity which are not in a hyperonym-hyponym or holonym-meronym relation with each other may be joined by coordination.

3.5 Linearization

The final step of our method is to linearize the dependency tree from the previous step into the final sequence of words. We implemented our own linearization method to take advantage of the ordering information can be inferred from the original source text sentences.

Our linearization algorithm proceeds top-down from the root of the dependency tree to the leaves. At each node of the tree, linearization consists of realizing the previously collapsed elements such as prepositions, determiners and noun compound elements, then ordering the dependent nodes with respect to the root node and each other. Restoring the collapsed elements is accomplished by simple heuristics. For example, prepositions and determiners precede their accompanying noun phrase.

The dependent nodes are ordered by a sorting algorithm, where the order between two syntactic relations and dependent nodes (r_1, a_1) and (r_2, a_2) is determined as follows. First, if a_1 and a_2 originated from the same source text sentence, then they are ordered according to their order of appearance in the source text. Otherwise, we consider the probability $P(r_1 \text{ precedes } r_2)$, and order a_1 before a_2 iff $P(r_1 \text{ precedes } r_2) > 0.5$. This distribution, $P(r_1 \text{ precedes } r_2)$, is estimated by counting and normalizing the order of the relation types in the source text corpus. For the purposes of ordering, the governor node is treated as if it were a dependent node with a special syntactic relation label *self*. This algorithm always produces an output ordering with a projective dependency tree, which is a reasonable assumption for English.

[4]We did not experiment with changing the grammatical voice in the output tree, such as introducing a passive construction if only a direct object is selected, but this is one possible extension of the algorithm.

4 Experiments

4.1 Method

Recent approaches to sentence fusion have often been evaluated as isolated components. For example, F&S evaluate the output sentences by asking human judges to rate the sentences' informativeness and grammaticality according to a 1–5 Likert scale rating. Thadani and McKeown (2013) combine grammaticality ratings with an automatic evaluation which compares the system output against gold-standard sentences drawn from summarization data sets. However, this evaluation setting still does not reflect the utility of sentence fusion in summarization, because the input sentences come from human-written summaries rather than the original source text.

We adopt a more realistic setting of using sentence fusion in automatic summarization by drawing the input or core sentences automatically from the source text, then evaluating the output of the fusion and expansion algorithm directly as one-sentence summaries according to standard summarization evaluation measures of content quality.

Data preparation. Our experiments are conducted on the TAC 2010 and TAC 2011 Guided Summarization corpus (Owczarzak and Dang, 2010), on the initial summarization task. Each document cluster is summarized by one sentence, generated from an initial cluster of core sentences as described in Section 3.1.

Evaluation measures. We evaluate summary content quality using the word-overlap measures ROUGE-1 and ROUGE-2, as is standard in the summarization community. We also measure the quality of sentences at a syntactic or shallow semantic level that operates at the level of dependency triples by a measure that we call **Pyramid BE**. Specifically, we extract all of the dependency triples of the form $t = (h, r, a)$ from the sentence under evaluation and the gold-standard summaries, where h and a are the lemmata of the head and the argument, and r is the syntactic relation, normalized for grammatical voice and excluding the collapsed edges which are mostly noun-phrase-internal elements and grammatical

Method	Pyramid BE	ROUGE-1	ROUGE-2	Log Likelihood	Oracle Pyramid BE
Fusion (F&S)	10.61	**10.07**	**2.15**	-159.31	28.00
Expansion	8.82	9.41	1.82	-157.46	52.97
+Event coref	**11.00**	9.76	1.93	**-156.20**	40.30

Table 1: Results of the sentence enhancement and fusion experiments.

particles. Then, we perform a matching between the set of triples in the sentence under evaluation and in a reference summary following the Transformed BE method of Tratz and Hovy (2008) with the *total* weighting scheme. This matching is performed between the sentence and every gold-standard summary, and the maximum of these scores is taken. This score is then divided by the maximum score that is achievable using the number of triples present in the input sentence, as inspired by the Pyramid method. This denominator is more appropriate than the one used in Transformed BE, which is designed for the case where the evaluated summary and the reference summaries are of comparable length.

For grammaticality, we parse the output sentences using the Stanford parser[5], and use the log likelihood of the most likely parse of the sentence as a coarse estimate of grammaticality. Parse log likelihoods have been shown to be useful in determining grammaticality (Wagner et al., 2009), and many of the problems associated with using it do not apply in our evaluation, because our sentences have a fixed number of content nodes, and contain similar content. While we could have conducted a user study to elicit Likert-scale grammaticality judgements, such results are difficult to interpret and the scores depend heavily on the set of judges and the precise evaluation setting, as is the case for sentence compression (Napoles et al., 2011).

4.2 Results and discussion

As shown in Table 1, sentence enhancement with coreference outperforms the sentence fusion algorithm of F&S in terms of the Pyramid BE measure and the baseline expansion algorithm, though only the latter difference is statistically significant ($p = 0.019$[6]). In terms of the ROUGE word overlap measures, fusion achieves a better performance, but it only outperforms the expansion baseline significantly (ROUGE-1: $p = 0.021$, ROUGE-2: $p = 0.012$). Note that the ROUGE scores are low because they involve comparing a one-sentence summary against a paragraph-long gold standard. The average log likelihood result suggests that sentence enhancement with event coreference produces sentences that are more grammatical than traditional fusion does, and this difference is statistically significant ($p = 0.044$). These results show that sentence enhancement with event coreference is competitive with a strong previous sentence fusion method in terms of content, despite having to combine information from more diverse sentences. This does not come at the expense of grammaticality; in fact, it seems that having a greater possible range of output sentences may even improve the grammaticality of the output sentences.

Oracle score. To examine the potential of sentence enhancement, we computed an oracle score that provides an upper bound to the best possible sentence that may be extracted from the sentence graph. First, we ranked all of dependency triples found in each gold-standard summary by their score (i.e., the number of gold-standard summaries they appear in). Then, we took the highest scoring triples from this ranking that are found in the sentence graph until the length limit was reached, and divided by the Pyramid-based denominator as above[7]. The oracle score is the maximum of these scores over the gold-standard summaries. The resulting oracle scores are shown in the rightmost column of Table 1. While it is no surprise that the oracle score improves after the sentence graph is expanded, the large increase in the oracle score indicates the potential of sentence enhancement for generating high-quality summary sentences.

[5]The likelihoods are obtained by the PCFG model of CoreNLP version 1.3.2. We experimented with the Berkeley parser (Petrov et al., 2006) as well, with similar results that favour the sentence enhancement with event coreference method, but because the parser failed to parse a number of cases, we do not report those results here.

[6]All statistical significance results in this section are for Wilcoxon signed-rank tests.

[7]There is no guarantee that these dependency triples form a tree structure. Hence, this is an upper bound.

Grammaticality. There is still room for improvement in the grammaticality of the generated sentences, which will require modelling contexts larger than individual predicates and their arguments. Consider the following output of the sentence enhancement with event coreference system:

(3) *The government has launched an investigation into Soeharto's wealth by the Attorney General's office on the wealth of former government officials.*

This sentence suffers from coherence problems because two pieces of information are duplicated. The first is the subject of the investigation, which is expressed by two prepositional objects of *investigation* with the prepositions *into* and *on*. The second, more subtle incoherence concerns the body that is responsible for the investigation, which is expressed both by the subject of *launch* (***The government*** *has launched an investigation*), and the *by*-prepositional object of *investigation* (an investigation ... ***by the Attorney General's office***). Clearly, a model that makes fewer independence assumptions about the relation between different edges in the sentence graph is needed.

5 A Study of Source-External Elements

The sentence enhancement algorithm presented above demonstrates that it is possible to use the entire source text to produce an informative sentence. Yet it is still limited by the particular predicates and dependency relations that are found in the source. The next step towards developing abstractive systems that exhibit human-like behaviour is to try to incorporate elements into the summary that are not found in the source text at all.

Despite its apparent difficulty, there is reason to be hopeful for text-to-text generation techniques even in such a scenario. In particular, we showed in earlier work that almost all of the **caseframes**, or pairs of governors and relations, in human-written summaries can be found in the source text or in a small set of additional related articles that belong to the same domain as the source text (e.g., natural disasters) (Cheung and Penn, 2013). What that study lacks, however, is a detailed analysis of the factors surrounding why human summary writers use non-source-text elements in their summaries, and how these may be automatically identified in the in-domain text. In this section, we supply such an analysis and provide evidence that human summary writers actually do incorporate elements external to the source text for a reason, namely, that these elements are more specific to the semantic content that they wish to convey. We also identify a number of features that may be useful for automatically determining the appropriateness of these in-domain elements in a summary.

5.1 Method

We performed our analysis on the predicates present in text, such as *kill* and *computer*. We also analyzed predicate-relation pairs (**PR pairs**) such as $(kill, nsubj)$ or $(computer, amod)$. This choice is similar to the caseframes used by Cheung and Penn (2013), and we similarly apply transformations to normalize for grammatical voice and other syntactic alternations, but we consider PR pairs of all relation types, unlike caseframes, which only consider verb complements and prepositional objects. PR pairs are extracted from the preprocessed corpus. We use the TAC 2010 Guided Summarization data set for our analyses, which we organize into two sub-studies. In the provenance study, we divide the PR pairs in human-written summaries according to whether they are found in the source text (**source-internal**) or not (**source-external**). In the domain study, we divide in-domain but source-external predicate-relation pairs according to whether they are used in a human-written summary (**gold-standard**) or not (**non-gold-standard**).

5.2 Provenance Study

In the first study, we compare the characteristics of gold-standard predicates and PR pairs according to their provenance; that is, are they found in the source text itself? The question that we try to answer is why human summarizers need to look beyond the source text at all when writing their summaries. We will provide evidence that they do so because they can find predicates that are more appropriate to the content that is being expressed according to two quantitative measures.

Predicate provenance. Source-external PR pairs may be external to the source text for two reasons. Either the predicate (i.e., the actual word) is found in the source text, but the dependency relation (i.e., the semantic predication that holds between the predicate and its arguments) is not found with that particular predicate, or the

	Average freq (millions)
Source-internal	1.77 (1.57, 2.08)
Source-external	1.15 (0.99, 1.50)

(a) The average predicate frequency of source-internal vs. source-external gold-standard predicates in an external corpus.

	Arg entropy
Source-internal	7.94 (7.90, 7.97)
Source-external	7.42 (7.37, 7.48)

(b) The average argument entropy of source-internal vs. source-external PR pairs in bits.

Table 2: Results of the provenance study. 95% confidence intervals are estimated by the bootstrap method and indicated in parentheses.

predicate itself may be external to the source text altogether. If the former is true, then a generalized version of the sentence enhancement algorithm presented in this paper could in principle capture these PR-pairs. We thus compute the proportion of source-external PR pairs where the predicate already exists in the source text.

We find that 2413 of the 4745 source-external PR pairs, or 51% have a predicate that can be found in the source text. This indicates that an extension of the sentence enhancement with event coreference approach presented in this paper could capture a substantial portion of the source-external PR pairs in its hypothesis space already.

Predicate frequency. What factors then can account for the remaining predicates that are not found in the source text at all? The first such factor we identify is the frequency of the predicates. Here, we take frequency to be the number of occurrences of the predicate in an external corpus; namely the Annotated Gigaword, which gives us a proxy for the specificity or informativeness of a word. In this comparison, we take the set of predicates in human-written summaries, divide them according to whether they are found in the source text or not, and then look up their frequency of appearance in the Annotated Gigaword corpus.

As Table 2a shows, the predicates that are not found in the source text consist of significantly less frequent words on average (Wilcoxon rank-sums test, $p < 10^{-17}$). This suggests that human summary writers are motivated to use source-external predicates, because they are able to find a more informative or apposite predicate than the ones that are available in the source text.

Entropy of argument distribution. Another measure of the informativeness or appropriateness of a predicate is to examine the range of arguments that it tends to take. A more generic word would be expected to take a wider range of arguments, whereas a more particular word would take a narrower range of arguments, for example those of a specific entity type. We formalize this notion by measuring the entropy of the distribution of arguments that a predicate-relation pair takes as observed in Annotated Gigaword. Given frequency statistics $f(h, r, a)$ of predicate head h taking an argument word a in relation r, we define the argument distribution of predicate-relation pair (h, r) as:

$$P(a|h, r) = f(h, r, a) / \sum_{a'} f(h, r, a') \quad (4)$$

We then compute the entropy of $P(a|h, r)$ for the gold-standard predicate-relation pairs, and compare the average argument entropies of the source-internal and the source-external subsets.

Table 2b shows the result of this comparison. Source-external PR pairs exhibit a lower average argument entropy, taking a narrower range of possible arguments. Together these two findings indicate that human summary writers look beyond the source text not just for the sake of diversity or to avoid copying the source text; they do so because they can find predicates that more specifically convey some desired semantic content.

5.3 Domain study

The second study examines how to distinguish those source-external predicates and PR pairs in in-domain articles that are used in a summary from those that are not. For this study, we rely on the topic category divisions in the TAC 2010 data set, and define the in-domain text to be the documents that belong to the same topic category as the target document cluster (but not including the target document cluster itself). This study demonstrates the importance of better semantic understanding for developing a text-to-text generation system that uses in-domain text, and identifies potentially useful features for training such a system.

Nearest neighbour similarity. In the event-coreference step of the sentence enhancement algorithm, we relied on distributional semantics to

	N	NN sim
GS	2202	0.493 (0.486, 0.501)
Non-GS	789K	0.443 (0.442, 0.443)

(a) Average similarity of gold-standard (GS) and non-gold-standard (non-GS) PR pairs to the nearest neighbour in the source text.

	N	Freq. (millions)	Fecundity
GS	1568	2.44 (2.05, 2.94)	21.6 (20.8, 22.5)
non-GS	268K	0.85 (0.83, 0.87)	6.43 (6.41, 6.47)

(b) Average frequency and fecundity of GS and non-GS predicates in an external corpus. The differences are statistically significant ($p < 10^{-10}$).

Table 3: Results of the domain study. 95% confidence intervals are given in parentheses.

measure the similarity of arguments. Here, we examine how well distributional similarity determines the appropriateness of a source-external PR pair in a summary. Specifically, we measure its similarity to the nearest PR pair in the source text. To determine the similarity between two PR pairs, we compute the cosine similarity between their vector representations. The vector representation of a PR pair is the concatenation of a context vector for the predicate itself and a selectional preferences vector for the PR pair; that is, the vector of counts with elements $f(h, r, a)$ for fixed h and r. These vectors are trained from the Annotated Gigaword corpus.

The average nearest-neighbour similarities of PR pairs are shown in Table 3a. While the difference between the gold-standard and non-gold-standard PR pairs is indeed statistically significant, the magnitude of the difference is not large. This illustrates the challenge of mining source-external text for abstractive summarization, and demonstrates the need for a more structured or detailed semantic representation in order to determine the PR pairs that would be appropriate. In other words, the kind of simple event coreference method based solely on distributional semantics that we used in Section 3.3.1 is unlikely to be sufficient when moving beyond the source text.

Frequency and fecundity. We also explore several features that would be relevant to identifying predicates in in-domain text that are used in the automatic summary. This is a difficult problem, as less than 0.6% of such predicates are actually used in the source text. As a first step, we consider several simple measures of the frequency and characteristics of the predicates.

The first measure that we compute is the average predicate frequency of the gold-standard and non-gold-standard predicates in an external corpus, as in Section 5.2. A second, related measure is to compute the number of possible relations that may occur with a given predicate. We call this measure the **fecundity** of a predicate. Both of these are computed with respect to the external Annotated Gigaword corpus, as before.

As shown in Table 3b, there is a dramatic difference in both measures between gold-standard and non-gold-standard predicates in in-domain articles. Gold-standard predicates tend to be more common words compared to non-gold-standard ones. This result is not in conflict with the result in the provenance study that source-external predicates are less common words. Rather, it is a reminder that the background frequencies of the predicates matter, and must be considered together with the semantic appropriateness of the candidate word.

6 Conclusions

This paper introduced sentence enhancement as a method to incorporate information from multiple points in the source text into one output sentence in a fashion that is more flexible than previous sentence fusion algorithms. Our results show that sentence enhancement improves the content and grammaticality of summary sentences compared to previous syntax-based sentence fusion approaches. Then, we presented studies on the components of human-written summaries that are external to the source text. Our analyses suggest that human summary writers look beyond the source text to find predicates and relations that more precisely express some target semantic content, and that more sophisticated semantic techniques are needed in order to exploit in-domain articles for text-to-text generation in summarization.

Acknowledgments

We would like to thank the anonymous reviewers for valuable suggestions. The first author was supported by a Facebook PhD Fellowship during the completion of this research.

References

Regina Barzilay and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Cosmin A. Bejan and Sanda Harabagiu. 2010. Unsupervised event coreference resolution with rich linguistic features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1412–1422. Association for Computational Linguistics.

Jackie Chi Kit Cheung and Gerald Penn. 2013. Towards robust abstractive multi-document summarization: A caseframe analysis of centrality and domain. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1233–1242, August.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research(JAIR)*, 31:399–429.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 137–144, Manchester, UK, August. Coling 2008 Organizing Committee.

Micha Elsner and Deepak Santhanam. 2011. Learning to fuse disparate sentences. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 54–63. Association for Computational Linguistics.

Katja Filippova and Michael Strube. 2008. Sentence fusion via dependency graph compression. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 177–185, Honolulu, Hawaii, October. Association for Computational Linguistics.

Katja Filippova. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 322–330, Beijing, China, August. Coling 2010 Organizing Committee.

Michel Galley and Kathleen McKeown. 2007. Lexicalized Markov grammars for sentence compression. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 180–187.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2012. Monolingual distributional similarity for text-to-text generation. In *Proceedings of *SEM 2012: The First Joint Conference on Lexical and Computational Semantics*, pages 256–264, Montréal, Canada, June. Association for Computational Linguistics.

Hongyan Jing and Kathleen R. McKeown. 2000. Cut and paste based text summarization. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, pages 178–185.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization—step one: Sentence compression. In *Proceedings of the National Conference on Artificial Intelligence*, pages 703–710.

Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*, COLING '00, pages 495–501, Stroudsburg, PA, USA. Association for Computational Linguistics.

Erwin Marsi and Emiel Krahmer. 2005. Explorations in sentence fusion. In *Proceedings of the European Workshop on Natural Language Generation*, pages 109–117.

Ryan T. McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Courtney Napoles, Benjamin Van Durme, and Chris Callison-Burch. 2011. Evaluating sentence compression: Pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 91–97. Association for Computational Linguistics.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the NAACL-HLT Joint Workshop on Automatic Knowledge Base Construction & Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100.

Karolina Owczarzak and Hoa T. Dang. 2010. TAC 2010 guided summarization task guidelines.

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

Horacio Saggion and Guy Lapalme. 2002. Generating indicative-informative summaries with SumUM. *Computational Linguistics*, 28(4):497–526.

Kapil Thadani and Kathleen McKeown. 2013. Supervised sentence fusion with single-stage inference. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1410–1418, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Stephen Tratz and Eduard Hovy. 2008. Summarization evaluation using transformed Basic Elements. In *Proceedings of the First Text Analysis Conference (TAC)*.

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

Joachim Wagner, Jennifer Foster, and Josef van Genabith. 2009. Judging grammaticality: Experiments in sentence classification. *CALICO Journal*, 26(3):474–490.

Stephen Wan, Robert Dale, Mark Dras, and Cecile Paris. 2008. Seed and grow: Augmenting statistically generated summary sentences using schematic word patterns. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 543–552, Honolulu, Hawaii, October. Association for Computational Linguistics.

ReferItGame: Referring to Objects in Photographs of Natural Scenes

Sahar Kazemzadeh[1*] **Vicente Ordonez**[1*] **Mark Matten**[2] **Tamara L. Berg**[1]
[1]University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA
[2]The Bishop's School, San Diego, CA 92037, USA
vicente@cs.unc.edu,tlberg@cs.unc.edu

Abstract

In this paper we introduce a new game to crowd-source natural language referring expressions. By designing a two player game, we can both collect and verify referring expressions directly within the game. To date, the game has produced a dataset containing 130,525 expressions, referring to 96,654 distinct objects, in 19,894 photographs of natural scenes. This dataset is larger and more varied than previous REG datasets and allows us to study referring expressions in real-world scenes. We provide an in depth analysis of the resulting dataset. Based on our findings, we design a new optimization based model for generating referring expressions and perform experimental evaluations on 3 test sets.

1 Introduction

Much of everyday language and discourse concerns the visual world around us, making understanding the relationship between objects in the physical world and language describing those objects an important challenge problem for AI. From robotics, to image search, to situated language learning, and natural language grounding, there are a number of research areas that would benefit from a better understanding of how people refer to physical entities in the world.

Recent advances in automatic computer vision methods have started to make technologies for recognizing thousands of object categories a near reality (Perronnin et al., 2012; Deng et al., 2012; Deng et al., 2010; Krizhevsky et al., 2012). As a result, there has been a spurt of recent work trying to estimate higher level semantics, including exciting efforts to automatically produce natural language descriptions of images and video (Farhadi et al., 2010; Kulkarni et al., 2011; Yang et al., 2011; Ordonez et al., 2011; Kuznetsova et al., 2012; Feng and Lapata, 2013). Common challenges encountered in these pursuits include the fact that descriptions can be highly task dependent, open-ended, and difficult to evaluate automatically.

Therefore, we look at the related, but more focused problem of referring expression generation (REG). Previous work on REG has made significant progress toward understanding how people generate expressions to refer to objects (a recent survey of techniques is provided in Krahmer and van Deemter (2012)). In this paper, we study the relatively unexplored setting of how people refer to objects in *complex photographs of real-world cluttered scenes*. One initial stumbling block to examining this scenario is lack of existing relevant datasets, as previous collections for studying REG have used relatively focused domains such as graphics generated objects (van Deemter et al., 2006; Viethen and Dale, 2008), crafts (Mitchell et al., 2010), or small everyday (home and office) objects arrayed on a simple background (Mitchell et al., 2013a; FitzGerald et al., 2013).

In this paper, we collect a new large-scale corpus, currently containing 130,525 expressions, referring to 96,654 distinct objects, in 19,894 photographs of real world scenes. Some examples from our dataset are shown in Figure 5. To construct this corpus efficiently, we design a new two player referring expression game (ReferItGame) to crowd-source the data collection. Popularized by efforts like the ESP game (von Ahn and Dabbish, 2004) and Peekaboom (von Ahn et al., 2006b), Human Computation based games can be an effective way to engage users and collect large amounts of data inexpensively. Two player games can also automate verification of human provided annotations.

Our resulting corpus is both more real-world and much bigger than previous datasets, allowing

[*]Indicates equal author contribution.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 787–798,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

us to examine referring expression generation in a new setting at large scale. To understand and quantify this new dataset, we perform an extensive set of analyses. One significant difference from previous work is that we study how referring expressions vary for different categories. We find that an object's category greatly influences the types of attributes used in their referring expression (e.g. people use color words to describe cars more often than mountains). Additionally, we find that references to an object are sometimes made with respect to other nearby objects, e.g. "the ball to left of the man". Interestingly, the types of reference objects (i.e. "the man") used in referring expressions is also biased toward some categories. Finally, we find that the word used to refer to the object category itself displays consistencies across people. This notion is related to ideas of entry-level categories from Psychology (Rosch, 1978).

Given these findings, we propose an optimization model for generating referring expressions that jointly selects which attributes to include in the expression, and what attribute values to generate. This model incorporates both visual models for selecting attribute-values and object category specific priors. Experimental evaluations indicate that our proposed model produces reasonable results for REG.

In summary, contributions of our paper include:

- A two player online game to collect and verify natural language referring expressions.
- A new large-scale dataset containing natural language expressions referring to objects in photographs of real world scenes.
- Analyses of the collected dataset, including studying category-specific variations in referring expressions.
- An optimization based model to generate referring expressions for objects in real-world scenes with experimental evaluations on three labeled test sets.

The rest of the paper is organized as follows. First we outline related work from the vision and language communities (§2). Then we describe our online game for collecting referring expressions (§3) and provide an analysis of our new ReferItGame Dataset (§4). Finally, we present and evaluate our model for generating referring expressions (§5) and discuss conclusions and future work (§6).

2 Related Work

Referring Expression Generation: There has been a long history of research on understanding how people generate referring expressions, dating back to the 1970s (Winograd, 1972). One common approach is the Incremental Algorithm (Dale and Reiter, 1995; Dale and Reiter, 2000) which uses logical expressions for generation. Much work in REG follows the Gricean maxims (Grice, 1975) which provide principles for how people will behave in conversation.

Recently, there has been progress examining other aspects of the referring expression problem such as understanding what types of attributes are used (Mitchell et al., 2013a), modeling variations between speakers (Viethen and Dale, 2010; Viethen et al., 2013; Van Deemter et al., 2012; Mitchell et al., 2013b), incorporating visual classifiers (Mitchell et al., 2011), producing algorithms to refer to object sets (Ren et al., 2010; FitzGerald et al., 2013), or examining impoverished perception REG (Fang et al., 2013). A good survey of work in this area is provided in Krahmer and van Deemter (2012). We build on past work, extending models to generate attributes jointly in a category specific framework.

Referring Expression Datasets: Some initial datasets in REG used graphics engines to produce images of objects (van Deemter et al., 2006; Viethen and Dale, 2008). Recently more realistic datasets have been introduced, consisting of craft objects like pipecleaners, ribbons, and feathers (Mitchell et al., 2010), or everyday home and office objects such as staplers, combs, or rulers (Mitchell et al., 2013a), arrayed on a simple background. These datasets helped moved referring expression generation research into the domain of real world objects. We seek to further these pursuits by constructing a dataset of natural objects in photographs of the real world.

Image & Video Description Generation: Recent research on automatic image description has followed two main directions. Retrieval based methods (Aker and Gaizauskas, 2010; Farhadi et al., 2010; Ordonez et al., 2011; Feng and Lapata, 2010; Feng and Lapata, 2013) retrieve existing captions or phrases to describe a query image. Bottom up methods (Kulkarni et al., 2011; Yang et al., 2011; Yao et al., 2010) rely on visual classifiers to first recognize image content and then construct captions from scratch, perhaps with some

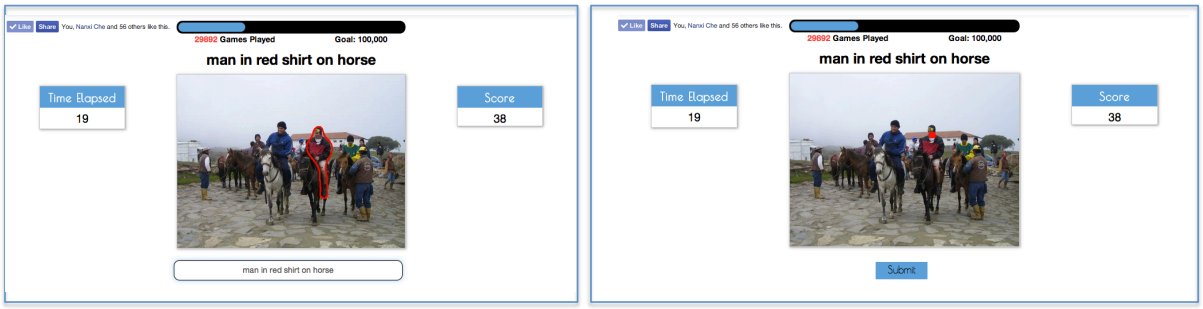


Figure 1: An example game. Player 1 (*left*) sees an image with an object outlined in red (the man) and provides a referring expression for the object ("man in red shirt on horse"). Player 2 (*right*) sees the image and the expression from Player 1 and must localize the correct object by clicking on it (click indicated by the red square). Elapsed time and current scores are also provided.

input from natural language statistics. Very recently, these ideas have been extended to produce descriptions for videos (Guadarrama et al., 2013; Barbu et al., 2012). Like these methods, we generate descriptions for natural scenes, but focus on referring to particular objects rather than providing an overall description of an image or video.

Human Computation Games: Games can be a useful tool for collecting large amounts of labeled data quickly. Human Computation Games were first introduced by Luis von Ahn in the ESP game (von Ahn and Dabbish, 2004) for image labeling, and later extended to segment objects (von Ahn et al., 2006b), collect common-sense knowledge (von Ahn et al., 2006a), or disambiguate words (Seemakurty et al., 2010). Recently, crowd games have also been introduced into the computer vision community for tasks like fine grained category recognition (Deng et al., 2013). These games can be released publicly on the web or used on Mechanical Turk to enhance and encourage turker participation (Deng et al., 2013). Inspired by the success of previous games, we create a game to collect and verify natural language expressions referring to objects in natural scenes.

3 Referring Expression Game (ReferItGame)

In this section we describe our referring expression game (ReferItGame*), a simple two player game where players alternate between generating expressions referring to objects in images of natural scenes, and clicking on the locations of described objects. An example game is shown in Figure 1.

*Available online at http://referitgame.com

3.1 Game Play

Player 1: is shown an image with an object outlined in red and provided with a text box in which to write a referring expression. *Player 2:* is shown the same image and the referring expression written by Player 1 and must click on the location of the described object (note, Player 2 does not see the object segmentation). If Player 2 clicks on the correct object, then both players receive game points and the Player 1 and Player 2 roles swap for the next image. If Player 2 does not click on the correct object then no points are received and the players remain in their current roles.

This provides us with referring expressions for our dataset and verification that the expressions are valid since they led to correct object localizations. Expressions written for games where the object was not correctly localized are kept and released with the dataset for future study, but are not included in our final dataset analyses or statistics. A game timer encourages players to write expressions quickly, resulting in more natural expressions. Also, IP addresses are filtered to prevent people from simultaneously playing both roles.

3.2 Playing Against the Computer

To promote engagement, we implement a single player version of the game. When a player connects, if there is another player online then the two people are paired. If there are currently no other available players, then the person plays a "canned" game against the computer. If at any point another person connects, the canned game ends and the player is paired with the new person.

To implement canned games we seed the game with 5000 pre-recorded referring expression games (5 referring expressions and resulting clicks

for each of 1000 objects) collected using Amazon's Mechanical Turk service. Implementing an automated version of Player 1 is simple; we just show the person one of the pre-collected referring expressions and they click as usual.

Automating the role of Player 2 is a bit more complicated. In this case, we compare the person's written expression against the pre-recorded expressions for the same object. For this comparison we use a parser to lemmatize the words in an expression and then compute cosine similarity between expressions with a bag of words representation. Based on this measure the closest matching expression is determined. If there is no similarity between the newly generated expression and the canned expressions, the expression is deemed incorrect and a random click location (outside of the object) is generated. If there is a successful match with a previously generated expression, then the canned click from the most similar pre-recorded game is used. More complex similarities could be used, but since we require real-time performance in our game setting we use this simple implementation which works well for our expressions.

4 ReferItGame Dataset

In this section we describe the ReferItGame dataset[†], including images and labels, processing the dataset, and analysis of the collection.

4.1 Images and Labels

We build our dataset of referring expressions on top of the ImageCLEF IAPR image retrieval dataset (Grubinger et al., 2006). This dataset is a collection of 20,000 images available free of charge without copyright restrictions, depicting a variety of aspects of everyday life, from sports, to animals, to cities, and landscapes. Crucial for our purposes, the SAIAPR TC-12 expansion (Escalante et al., 2010) includes segmentations of each image into regions indicating the locations of constituent objects. 238 different object categories are labeled, including animals, people, buildings, objects, and background elements like grass or sky. This provides us with information regarding object category, object location, and object size, as well as the location and categories of other objects present in the same image.

[†] Available at http://tamaraberg.com/referitgame

4.2 Collecting the Dataset

From the ImageCLEF dataset, we created a total of over 100k distinct games (one per object labeled in the dataset). For the games we imposed an ordering to allow for collecting the most interesting expressions first. Initially we prioritized games for objects in images with multiple objects of the same category. Once these games were completed, we prioritized ordering based on object category to include a comprehensive range of objects. Finally, after successfully collecting referring expressions from the prioritized games, we posted games for the remaining objects. In order to evaluate consistency of expression generation across people, we also include a probability of repeating previously played games during collection.

To date, we have collected 130,525 successfully completed games. This includes 10,431 canned games (a person playing against the computer, not including the initial seed set) and 120,094 real games (two people playing). 96,654 distinct objects from 19,984 photographs are represented in the dataset. This covers almost all of the objects present in the IAPR corpus. The remaining objects from the collection were either too small or too ambiguous to result in successful games.

For data collection, we posted the game online for anyone on the web to play and encouraged participation through social media and the survey section of reddit. In this manner we collected over 4 thousand referring expressions over a period of 3 weeks. To speed up data collection, we also posted the game on Mechanical Turk. Turkers were paid upon completion of 10 correct games (games where Player 2 clicks on the correct object of interest). Turkers were pre-screened to have approval ratings above 80% and to be located in the US for language consistency.

4.3 Processing the Dataset

Because of the size of the dataset, hand annotation of all referring expressions is prohibitive. Therefore, similar to past work (FitzGerald et al., 2013), we design an automatic method to pre-process the expressions and extract object and attribute mentions. These automatically processed expressions are used only for analysis and model training. We also fully hand label portions of the dataset for evaluation (§5.2).

By examining the expressions in the collected dataset, we define a set of attributes with broad

$$
\begin{aligned}
S &::= subject\_word \\
color\_word' &::= rel(S, color\_word)_{color\_word'=color\_word} \mid \\
&\quad prep\_in(S, color\_word)_{color\_word'=color\_word} \\
size\_word' &::= rel(S, size\_word)_{size\_word'=size\_word} \\
abs\_loc\_word' &::= rel(S, abs\_loc\_word)_{\ abs\_loc\_word'=abs\_loc\_word} \mid \\
&\quad prep\_on(S, orientation\_word) \wedge \neg prep\_of(S, \_)_{abs\_loc\_word'=on+orientation\_word} \\
rel\_loc\_word' &::= RL \\
RL &::= prep\_rel\_loc\_word(S, object\_word)_{RL=rel\_loc\_word} \mid \\
&\quad prep\_on(S, orientation\_word) \wedge prep\_of(S, object\_word)_{\ RL=on\_orientation\_word} \mid \\
&\quad prep\_to(S, orientation\_word) \wedge prep\_of(S, object\_word)_{\ RL=to\_orientation\_word} \mid \\
&\quad prep\_at(S, orientation\_word) \wedge prep\_of(S, object\_word)_{\ RL=at\_orientation\_word} \\
generic\_word' &::= amod(S, generic\_word)
\end{aligned}
$$

Figure 2: Templates for parsing attributes from referring expressions (§4.3).

coverage of the attribute types used in the referring expressions. We define the set of attributes for a referring expression as a 7-tuple $R = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$:

- r_1 is an entry-level category attribute,
- r_2 is a color attribute,
- r_3 is a size attribute,
- r_4 is an absolute location attribute,
- r_5 is a relative location relation attribute,
- r_6 is a relative location object attribute,
- r_7 is a generic attribute,

Color and *size* attributes refer to the object color (e.g. "blue") and object size (e.g. "tiny") respectively. *Absolute location* refers to the location of the object in the image (e.g. "top of the image"). *Relative location relation* and *relative location object* attributes allow for referring expressions that localize the object with respect to another object in the picture (e.g. "the car to the left of the tree"). *Generic attributes* cover all less frequently observed attribute types (e.g. "wooden" or "round").

The *entry-level category attribute* is related to the concept of entry-level categories first proposed by Psychologists in the 1970s (Rosch, 1978) and recently explored in visual recognition (Ordonez et al., 2013). The idea of entry-level categories is that an object can belong to many different categories; an indigo bunting is an oscine, a bird, a vertebrate, a chordate, and so on. But, a person looking at a picture of one would probably call it a bird (unless they are very familiar with ornithology). Therefore, we include this attribute to capture how people name object categories in referring expressions.

Parsing the referring expressions: We parse the expressions using the most recent version of the StanfordCoreNLP parser (Socher et al., 2013). We begin by traversing the parse tree in a breadth-first manner and selecting the head noun of the sentence to determine the object of the referring expression, denoted as $subject\_word$. We pre-define a dictionary of attribute-values ($color\_word$, $size\_word$, $abs\_location\_word$, $rel\_location\_word$) for each of the attributes based on the observed data using a combination of POS-tagging and manual labeling.

We then apply a template-based approach on the collapsed dependency relations to recover the set of attributes (the main template rules are shown in Figure 2). The relationship rel indicates any linguistic binary relationship between the subject word S and another word, including the $amod$ relationship. $Orientation\_word$ captures the words like left, right, top and bottom. For $generic\_word$ we consider any modifier words other than those captured by our other attributes (color, size, location).

Using this template-based parser we can for instance parse the following expression: "Red flower on top of pedestal". The first rule would match the $prep(S, color\_word)$ relation, effectively recovering the attribute $color\_word'$ as "red". The second rule would match the $prep\_on(S, orientation\_word) \wedge prep\_of(S, object\_word)$ relations, recovering rel_loc_word' as "on top of " and $object\_word$ as "pedestal".

The accuracy of our parser based processing is 91%. This was evaluated on 4,500 expressions

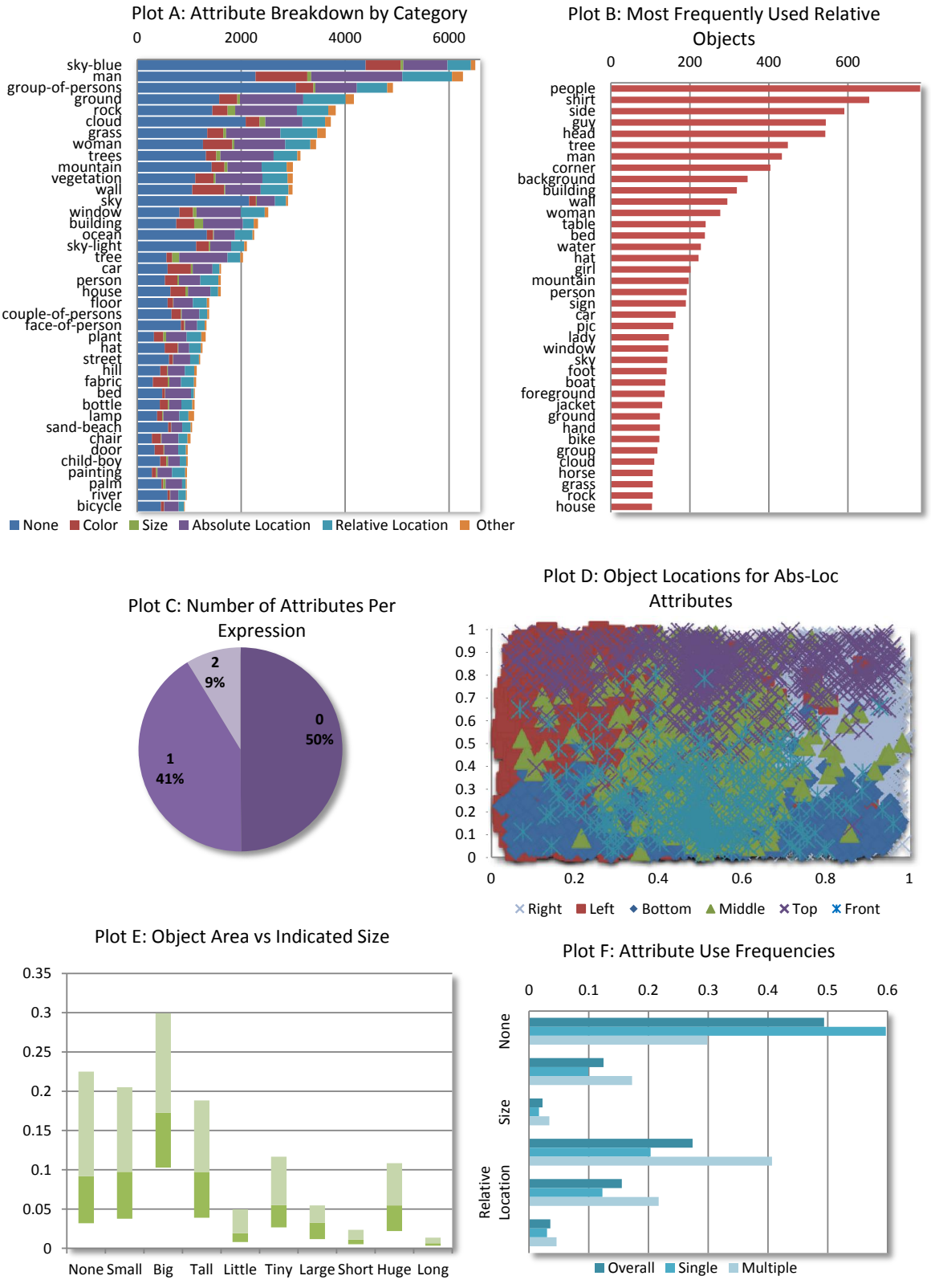


Figure 3: Analyses of the ReferItGame Dataset. **Plot A** shows frequency and attribute occurrence for common object categories. **Plot B** shows objects frequently used as reference points, ie "to the left of the man". **Plot C** shows frequencies of using 0, 1 or 2 attributes within the same expression. **Plot D** shows object locations vs location words used. **Plot E** shows normalized object size vs size words used (bars show 1^{st} through 3^{rd} quartiles). **Plot F** shows the frequency of usage of each attribute type for images containing either a *single* instance of the object category or *multiple* instances of the category.

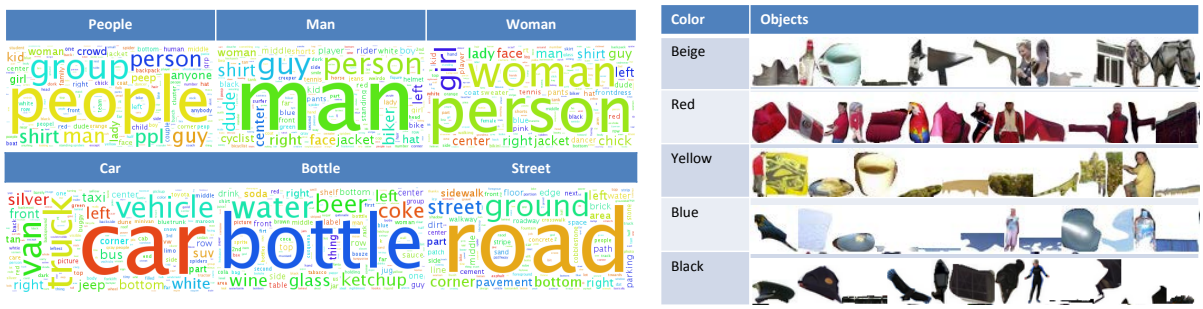


Figure 4: **Left:** Tag clouds showing entry-Level category words used in referring expressions to name various object categories, with word size indicating frequency. For example, this indicates that "streets" are often called "road", sometimes "ground", sometimes "roadway", etc. **Right:** example objects predicted to portray some of our color attribute values. Note sometimes our color predictor is quite accurate, and sometimes it makes mistakes (see the man in a red shirt predicted as "yellow").

that were manually parsed by a human annotator.

4.4 Dataset Analysis

In the resulting dataset, we have a range of coverage over objects. For 10,304 of the objects we have 2 or more referring expressions while for the rest of the objects we have collected only one expression. This creates a dataset that emphasizes breadth while also containing enough data to study speaker variation.

Multiple attribute analyses are provided in Figure 3. We find that most expressions use 0, 1, or 2 attributes (in addition to the entry-level attribute object word), with very few expressions containing more than 2 attributes (frequencies are shown in Fig 3c). We also examine what types of attributes are used most frequently, according to object category in Fig 3a, and when associated with single or multiple occurrences of the same object category in an image in Fig 3f. The frequency of attribute usage in images containing multiple objects of the same type increases for all types, compared to single object occurrences. Perhaps more interestingly, the use of different attributes is highly category dependent. People use more attribute words overall to describe some categories, like "man", "woman", or "plant", and the distribution of attribute types also varies by category. For example, color attributes are used more frequently for categories like "car" or "woman" than for categories like "sky" or "rock".

We also examine which objects are most frequently used as points of reference, e.g.,"the chair next to the *man*" in Fig 3b. We observe that people and some background categories like "tree" or "wall" are often used to help localize objects in referring expressions. Additionally, we provide plots showing the relationship between object location in the image and use of absolute location words, Fig 3d, as well as size words vs object area, Fig 3e.

Finally, we study entry-level category attribute-values to understand how people name objects in referring expressions. Tag clouds indicating the frequencies of words used to name various object categories are provided in Fig 4 (left). Objects like "street" are usually referred to as "road", but sometimes they are called "ground", "roadway", etc. "Bottles" are usually called "bottle", but sometimes referred to as "coke" or "beer". Interestingly, "man" is usually called "man" while "woman" is most often called "person" in the referring expressions.

5 Generating Referring Expressions

In this section we describe our proposed generation model and provide experimental evaluations on three test sets.

5.1 Generation Model

Given an input tuple $I = \{P, S\}$, where P is a target object and S is a scene (image containing multiple objects), our goal is to generate an output referring expression, R. For instance, the representation R for the referring expression: *The big old white cabin beside the tree* would be $R = \{cabin, white, big, \varnothing, beside, tree, old\}$.

To generate referring expressions we construct vocabularies V_{r_i} with candidate values for each attribute $r_i \in R$, where attribute vocabulary V_{r_i} contains the set of words observed in our parsed referring expressions for attribute r_i plus an additional

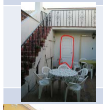
Image	Human Expressions	Generated Expressions
	picture on the wall picture picture	**Baseline:**[picture, white, , right, , ,] **Full:** [picture, , , , prep_on, wall,]
	Door white door middle white door	**Baseline:**[door, white, , right, , ,] **Full:**[door, white, , right, , ,]
	big gated window on right of white section black big window right brown railings on right	**Baseline:**[window, white, , right, , ,] **Full:**[window, brown, , right, , ,]
	white shirt man white shirt on right man on right	**Baseline:**[man, white, , right, , ,] **Full:**[man, white, , right, , ,]
	building on right behind guys blue right building building on right	**Baseline:**[building, white, , right, , ,] **Full:**[building, white, , right, , ,]

Image	Human Expressions	Generated Expressions
	picture santa the santa picture	**Baseline:**[picture, white, , right, , ,] **Full:**[picture, , , , prep_on, plant,]
	right doorway right brown door right door	**Baseline:**[door, , , right, prep_on, person,] **Full:**[door, , , right, prep_above, person,]
	with flag window top 2nd left 2nd window top left	**Baseline:**[window, , , right, prep_on, person,] **Full:**[window, , , left, prep_above, door,]
	red guy left sitting left bottom guy red shirt lef	**Baseline:**[man, , , right, prep_on, wall,] **Full:**[man, , , left, prep_in, woman,]
	buildings buildings buildings	**Baseline:**[building, white, , right, , ,] **Full:**[building, brown, , middle, , ,

Figure 5: Example results, including human generated expressions, baseline and full model generated expressions. For some images the model does well at mimicking human expressions (left). For others it does not generate the correct attributes (right).

ε value indicating that the attribute should be omitted from the referring expression entirely.

In this way, our framework can jointly determine which attributes to include in the expression (e.g.,"size" and "color") and what attribute values to generate (e.g.,"small" and "blue") from the list of all possible values. We enforce a constraint to always include an "entry-level category" attribute (e.g. "boy") so that we always generate a word referring to the object.

We pose our problem as an optimization where we map a tuple $\{P, S\}$ to a referring expression R^* as:

$$R^* = \underset{R}{\text{argmax}} E(R, P, S) \quad \text{s. t. } f_i(R) \leq b_i \tag{1}$$

Where the objective function E is decomposed as:

$$E(R,P,S) = \alpha \sum_{i=2}^{6} \phi_i(r_i, P, S) + \beta \sum_{i=1}^{7} \psi_i(r_i, type(P)) + \sum_{i>j} \psi_{i,j}(r_i, r_j) \tag{2}$$

Where ϕ_i is the compatibility function between an attribute-value for r_i and the properties of the observed scene S and object P (described in §5.1.1). The terms ψ_i and $\psi_{i,j}$ are unary and pairwise priors computed based on observed co-occurrence statistics of attribute-values for r_i with categories (where $type(P)$ denotes the type or category of an object) and between pairs of attribute-values (described in §5.1.2). Attributes r_1 and r_7 are modeled only in the priors since we do not have visual models for these attributes.

The constraints $f_i(R) \leq b_i$ are restricted to be linear constraints and are used to impose hard constraints on the solution. The first such constraint is used to control the verbosity (length) of the generated referring expression using a constraint function that imposes a minimum attribute length requirement by restricting the number of entries r_i that can take value ε in the solution.

$$\sum_i \mathbb{1}[r_i = \varepsilon] \leq 7 - \gamma(P, S) \tag{3}$$

Where $\mathbb{1}[.]$ is the indicator function and $\gamma(P, S)$ is a term that allows us to change the length requirement based on the object and scene (so that images with a larger number of objects of the same type have a larger length requirement).

Finally we add hard constraints such that $r_5 = \varepsilon \iff r_6 = \varepsilon$, so that relative location and relative object attributes are produced together.

5.1.1 Content-based potentials

Potentials ϕ_i are defined for attributes r_2 to r_6. Attribute r_7 represents a variety of different attributes, e.g. material or shape attributes, but we lack sufficient data to train visual models for these infrequent attribute terms. Therefore, we model these attributes using only prior statistics-based potentials (§5.1.2). Visual recognition models for recognizing entry-level object categories

could also be incorporated for modeling r_1, but we leave this as future work.

Color attribute:

$$\phi_2(r_2 = c_k, P, S) = sim(hist_{c_k}, hist(P))$$

Where $hist(P)$ is the HSV color histogram of the object P. We compute similarity sim using cosine similarity, and $hist_{c_k}$ is the mean histogram of all objects in our training data that were referred to with color attribute-value $c_k \in V_{r_2}$.

Size attribute:

$$\phi_3(r_3 = s_k, P, S) = \frac{1}{\sigma_{s_k}\sqrt{2\pi}} e^{-\left(size(P)-\mu_{s_k}\right)^2 / 2\sigma_{s_k}^2}$$

Where $size(P)$ is the size of object P normalized by image size. We model the probabilities of each size word $s_k \in V_{r_3}$ as a Gaussian learned on our training set.

Absolute-location attribute:

$$\phi_4(r_4 = a_k, P, S) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_{a_k}|}} e^{-\frac{1}{2}(loc(P)-\mu_{a_k})^T \Sigma_{a_k}{}^{-1} (loc(P)-\mu_{a_k})}$$

Where $loc(P)$ are the 2-dimensional coordinates of the object P normalized to be $\in [0-1]$. Parameters μ_{a_k} and Σ_{a_k} are estimated from training data for each absolute location word $a_k \in V_{r_4}$.

Relative-location and Relative object:

$$\phi_5(r_5 = l_k, P, S) = \mathbb{1}[l_k = \varepsilon] \cdot g(count(type(P), S))$$

If there are a larger number of objects of the same type in the image we find that the probability of using a relative-location-object increases (e.g., "the car to the right of the man"). For images where P was the only object of that category type, the probability of using a relative-location-object is 0.12. This increases to 0.22 when there were two objects of the same type and further increases to 0.26 for additional objects of the same type. Therefore, we model the probability of selecting relative location value $l_k \in V_{r_5}$ as a function g, where $count(type(P), S)$ counts the number of objects in the scene S of the same category type as the object P.

$$\phi_6(r_6 = o_k, P, S) = \mathbb{1}[o_k \in objectsnear(location(P), S)]$$

The above expression filters out potential relative objects $o_k \in V_{r_6}$ that are not located in sufficient proximity to object P or are not present in the image at all.

5.1.2 Prior statistics-based potentials

Prior statistics-based potentials are modeled for all of the attributes r_1 - r_7. Note that these potentials do not depend on specific attribute-values but only on the given object category $type(P)$.

Unary prior potentials ψ_i are defined as:

$$\psi_i(r_i, type(P)) = \frac{\sum_{j=1}^{|D|} \mathbb{1}[(r_i^{(j)} \neq \epsilon) \wedge (type(P^{(j)}) = type(P))]}{\sum_{j=1}^{|D|} \mathbb{1}[type(P^{(j)}) = type(P)]} + \frac{\sum_{j=1}^{|D|} \mathbb{1}[r_i^{(j)} \neq \epsilon]}{|D|} + \lambda$$

Where $D = \{P^{(j)}, S^{(j)}, R^{(j)}\}$ is our training dataset and λ is a small additive smoothing term. The two terms in the above expression represent *category-specific* counts and *global* counts of the number of times a given attribute r_i was output in a referring expression in training data. Pairwise prior potentials $\psi_{i,j}$ are defined as:

$$\sum_{i<j} \psi_{i,j}(r_i, r_j) = \sum_{i<j} \psi_{i,j}^{(1)}(r_i, r_j) + \psi_{5,6}^{(2)}(r_5, r_6)$$

$$\psi_{i,j}^{(1)}(r_i, r_j) = \begin{cases} 1 & \text{if } r_i = r_j = \varepsilon \\ C + \lambda & o.w. \end{cases}$$

$$\psi_{5,6}^{(2)}(r_5 = a, r_6 = b) = \frac{\sum_{t=1}^{|D|} \mathbb{1}[(r_5^{(t)} = a) \wedge (r_6^{(t)} = b)]}{|D|}$$

where $C = \frac{\sum_{t=1}^{|D|} \mathbb{1}[(r_i^{(t)} \neq \epsilon) \wedge (r_j^{(t)} \neq \epsilon)]}{|D|}$. The pairwise potential $\psi_{i,j}^{(1)}$ captures the pairwise statistics of how frequently people use pairs of attribute types.

SOURCE	PREC(%)	RECALL(%)
Baseline - A	27.92	43.27
Full Model - A	**36.28**	**53.44**
Baseline - B	29.87	50.57
Full Model - B	**36.68**	**59.80**
Baseline - C	28.85	37.41
Full Model - C	**37.73**	**48.54**

Table 1: Baseline Model & Full Model performance on the three test sets (A,B,C).

For instance how frequently people use both color and size attributes to refer to an object. The pairwise potential $\psi_{i,j}^{(2)}$ produces a cohesion score between relative-location words and relative-object words based on global dataset statistics.

5.2 Experiments

We implement the proposed model using commercial binary integer linear programming software (IBM ILOG CPLEX). This requires introducing a set of indicator variables for each of our multi-valued attributes and another set of indicator variables to model pairwise interactions between our variables, as well as incorporating additional consistency constraints between variables. Model parameters (α and β) are tuned on data randomly sampled from the training set.

Test Sets: We evaluate our model on three test sets, each containing 500 objects. For each object in the test sets we collect 3 referring expressions using the ReferItGame and manually label the attributes mentioned in each expression. We find human agreement to be 72.31% on our dataset (where we measure agreement as mean matching accuracy of attribute values for pairs of users across images in our test sets). The three test sets are created to evaluate different aspects of our data.

Test Set A contains objects sampled randomly from the entire dataset. This test set is meant to closely resemble the full dataset distribution. The goal of the other two test sets is to sample expressions for "interesting" objects. We first identify categories that are mainly related to background content elements, e.g. "sky, ground, floor, sand, sidewalk, etc". We consider these categories to be potentially less interesting for study than categories like people, animals, cars, etc. *Test Set B* contains objects sampled from the most frequently occurring object categories in the dataset, selected to contain a balanced number of objects from each category, excluding the less interesting categories. *Test Set C* contains objects sampled from images that contain at least 2 objects of the same category, excluding the less interesting categories.

Results: *Qualitative examples* are shown in Fig 5 comparing our results to the human produced expressions. For some images (left) we do quite well at predicting the correct attributes and values. For others we do less well (right). We also show example objects predicted for some color words in Fig 4 (right). We see that our model can fail in several ways, such as generating the wrong attribute-value due to inaccurate predictions by visual models or selecting incorrect attributes to include in the generated expression.

Quantitative results: precision and recall measures for the 3 test sets are reported in Table 1, including evaluation of a baseline version of our model which incorporates only the prior potentials (§5.1.2) without any content based estimates. We see that our model performs reasonably on both measures, and outperforms the baseline by a large margin on all test sets, with highest performance on the broadly sampled interesting category test set. Note that our problem is somewhat different than traditional REG where the input is often attribute-value pairs and the task is to select which pairs to include in the expression. Our goal is to jointly select which attributes to include and what values to predict from a list of all possible values for the attribute.

6 Conclusions & Future Work

In this paper we have introduced a new game to crowd-source referring expressions for objects in natural scenes. We have used this game to produce a new large-scale dataset with analysis. We have also proposed an optimization based model for REG and performed experimental evaluations. Future work includes developing fully automatic visual recognition methods for REG in real world scenes, and incorporating linguistically inspired models for entry-level category prediction.

Acknowledgments

This work was funded by NSF Awards #1417991 and #1444234. M.M. was supported by the Stony Brook Simons Summer Research Program for High School students. We also thank Alex Berg for many helpful discussions.

References

Ahmet Aker and Robert Gaizauskas. 2010. Generating image descriptions using dependency relational patterns. In *Association for Computational Linguistics (ACL)*.

Andrei Barbu, Alexander Bridge, Zachary Burchill, Dan Coroian, Sven J. Dickinson, Sanja Fidler, Aaron Michaux, Sam Mussman, Siddharth Narayanaswamy, Dhaval Salvi, Lara Schmidt, Jiangnan Shangguan, Jeffrey Mark Siskind, Jarrell W. Waggoner, Song Wang, Jinlian Wei, Yifan Yin, and Zhiqi Zhang. 2012. Video in sentences out. In *Uncertainty in Artificial Intelligence (UAI)*.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science (CogSci)*, 19:233264.

Robert Dale and Ehud Reiter. 2000. Building natural language generation systems. In *Cambridge University Press*.

Jia Deng, Alexander C. Berg, Kai Li, and Fei-Fei Li. 2010. What does classifying more than 10,000 image categories tell us? In *European Conference on Computer Vision (ECCV)*.

Jia Deng, Alex Berg, Sanjeev Satheesh, Hao Su, Aditya Khosla, and Fei-Fei Li. 2012. Large scale visual recognition challenge. In *http://www.image-net.org/challenges/LSVRC/2012/index*.

Jia Deng, Jonathan Krause, and Li Fei-Fei. 2013. Fine-grained crowdsourcing for fine-grained recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hugo Jair Escalante, Carlos A. Hernandez, Jesus A. Gonzalez, A. Lopez-Lopez, Manuel Montes, Eduardo F. Morales, L. Enrique Sucar, Luis Villasenor, and Michael Grubinger. 2010. The segmented and annotated iapr tc-12 benchmark. *Computer Vision and Image Understanding (CVIU)*.

Rui Fang, Changsong Liu, Lanbo She, and Joyce Chai. 2013. Towards situated dialogue: Revisiting referring expression generation. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: generating sentences for images. In *European Conference on Computer Vision (ECCV)*.

Yansong Feng and Mirella Lapata. 2010. How many words is a picture worth? automatic caption generation for news images. In *Association for Computational Linguistics (ACL)*.

Yansong Feng and Mirella Lapata. 2013. Automatic caption generation for news images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(4):797–812.

Nicholas FitzGerald, Yoav Artzi, and Luke Zettlemoyer. 2013. Learning distributions over logical forms for referring expression generation. In *Empirical Methods on Natural Language Processing (EMNLP)*.

H. Paul Grice. 1975. Logic and conversation. page 4158.

Michael Grubinger, Paul D. Clough, Henning Muller, and Thomas Deselaers. 2006. The iapr benchmark: A new evaluation resource for visual information systems. In *Proceedings of the International Workshop OntoImage (LREC)*.

Sergio Guadarrama, Niveda Krishnamoorthy, Girish Malkarnenkar, Subhashini Venugopalan, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2013. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *International Conference on Computer Vision (ICCV)*.

Emiel Krahmer and Kees van Deemter. 2012. Computational generation of referring expressions: A survey. In *Computational Linguistics*, volume 38, page 173218.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems (NIPS)*.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. 2011. Babytalk: Understanding and generating simple image descriptions. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.

Polina Kuznetsova, Vicente Ordonez, Alex Berg, Tamara L Berg, and Yejin Choi. 2012. Collective generation of natural image descriptions. In *Association for Computational Linguistics (ACL)*.

Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2010. Natural reference to objects in a visual domain. In *International Natural Language Generation Conference (INLG)*.

Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2011. Two approaches for generating size modifiers. In *European Workshop on Natural Language Generation*.

Margaret Mitchell, Ehud Reiter, and Kees van Deemter. 2013a. Typicality and object reference. In *Cognitive Science (CogSci)*.

Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2013b. Generating expressions that refer to visible objects. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *Neural Information Processing Systems (NIPS)*.

Vicente Ordonez, Jia Deng, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. 2013. From large scale image categorization to entry-level categories. In *International Conference on Computer Vision (ICCV)*.

Florent Perronnin, Zeynep Akata, Zaid Harchaoui, and Cordelia Schmid. 2012. Towards good practice in large-scale learning for image classification. In *Computer Vision and Pattern Recognition (CVPR)*.

Yuan Ren, Kees Van Deemter, and Jeff Z Pan. 2010. Charting the potential of description logic for the generation of referring expressions. In *International Natural Language Generation Conference (INLG)*.

Eleanor Rosch. 1978. Principles of categorization. *Cognition and Categorization*, page 2748.

Nitin Seemakurty, Jonathan Chu, Luis von Ahn, and Anthony Tomasic. 2010. Word sense disambiguation via human computation. In *Human Computation Workshop*.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing With Compositional Vector Grammars. In *Association for Computational Linguistics (ACL)*.

Kees van Deemter, Ielka van der Sluis, and Albert Gatt. 2006. Building a semantically transparent corpus for the generation of referring expressions. In *International Conference on Natural Language Generation (INLG)*.

Kees Van Deemter, Albert Gatt, Roger PG van Gompel, and Emiel Krahmer. 2012. Toward a computational psycholinguistics of reference production. In *Topics in Cognitive Science*, volume 4(2), page 166183.

Jette Viethen and Robert Dale. 2008. The use of spatial relations in referring expression generation. In *International Natural Language Generation Conference (INLG)*.

Jette Viethen and Robert Dale. 2010. Speaker-dependent variation in content selection for referring expression generation. In *Australasian Language Technology Workshop*.

Jette Viethen, Margaret Mitchell, and Emiel Krahmer. 2013. Graphs and spatial relations in the generation of referring expressions. In *European Workshop on Natural Language Generation*.

Luis von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *ACM Conf. on Human Factors in Computing Systems (CHI)*.

Luis von Ahn, Mihir Kedia, and Manuel Blum. 2006a. Verbosity: A game for collecting common-sense knowledge. In *ACM Conference on Human Factors in Computing Systems (CHI)*.

Luis von Ahn, Ruoran Liu, and Manuel Blum. 2006b. Peekaboom: A game for locating objects in images. In *ACM Conference on Human Factors in Computing Systems (CHI)*.

Terry Winograd. 1972. Understanding natural language. *Cognitive Psychology*, 3(1):1191.

Yezhou Yang, Ching Lik Teo, Hal Daume III, and Yiannis Aloimonos. 2011. Corpus-guided sentence generation of natural images. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Benjamin Z. Yao, Xiong Yang, Liang Lin, Mun Wai Lee, and Song-Chun Zhu. 2010. I2t: Image parsing to text description. *Proc. IEEE*, 98(8).

Unsupervised Template Mining for Semantic Category Understanding

Lei Shi[1,2]*, Shuming Shi[3], Chin-Yew Lin[3], Yi-Dong Shen[1], Yong Rui[3]
[1]State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]Microsoft Research
{shilei,ydshen}@ios.ac.cn
{shumings,cyl,yongrui}@microsoft.com

Abstract

We propose an unsupervised approach to constructing templates from a large collection of semantic category names, and use the templates as the semantic representation of categories. The main challenge is that many terms have multiple meanings, resulting in a lot of wrong templates. Statistical data and semantic knowledge are extracted from a web corpus to improve template generation. A nonlinear scoring function is proposed and demonstrated to be effective. Experiments show that our approach achieves significantly better results than baseline methods. As an immediate application, we apply the extracted templates to the cleaning of a category collection and see promising results (precision improved from 81% to 89%).

1 Introduction

A semantic category is a collection of items sharing common semantic properties. For example, all cities in Germany form a semantic category named "city in Germany" or "German city". In Wikipedia, the category names of an entity are manually edited and displayed at the end of the page for the entity. There have been quite a lot of approaches (Hearst, 1992; Pantel and Ravichandran, 2004; Van Durme and Pasca, 2008; Zhang et al., 2011) in the literature to automatically extracting category names and instances (also called is-a or hypernymy relations) from the web.

Most existing work simply treats a category name as a text string containing one or multiple words, without caring about its internal structure. In this paper, we explore the *semantic* structure of category names (or simply called "categories"). For example, both "CEO of General Motors" and "CEO of Yahoo" have structure "CEO of [company]". We call such a structure a *category template*. Taking a large collection of open-domain categories as input, we construct a list of *category templates* and build a mapping from categories to templates. Figure 1 shows some example semantic categories and their corresponding templates.

Templates can be treated as additional features of semantic categories. The new features can be exploited to improve some upper-layer applications like web search and question answering. In addition, by linking categories to templates, it is possible (for a computer program) to infer the semantic meaning of the categories. For example in Figure 1, from the two templates linking to category "symptom of insulin deficiency", it is reasonable to interpret the category as: "a symptom of a medical condition called insulin deficiency which is about the deficiency of one type of hormone called insulin." In this way, our knowledge about a category can go beyond a simple string and its member entities. An immediate application of templates is removing invalid category names from a noisy category collection. Promising results are observed for this application in our experiments.

An intuitive approach to this task (i.e., extracting templates from a collection of category names)

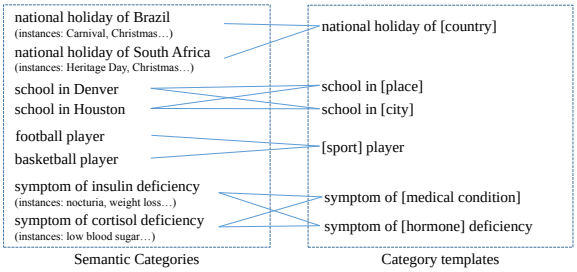


Figure 1: Examples of semantic categories and their corresponding templates.

*This work was performed when the first author was visiting Microsoft Research Asia.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 799–809,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

contains two stages: category labeling, and template scoring.

Category labeling: Divide a category name into multiple segments and replace some key segments with its hypernyms. As an example, assume "CEO of Delphinus" is divided to three segments "CEO + of + Delphinus"; and the last segment (Delphinus) has hypernyms "constellation", "company", etc. By replacing this segment with its hypernyms, we get candidate templates "CEO of [constellation]" (a wrong template), "CEO of [company]", and the like.

Template scoring: Compute the score of each candidate template by aggregating the information obtained in the first phase.

A major challenge here is that many segments (like "Delphinus" in the above example) have multiple meanings. As a result, wrong hypernyms may be adopted to generate incorrect candidate templates (like "CEO of [constellation]"). In this paper, we focus on improving the template scoring stage, with the goal of assigning lower scores to bad templates and larger scores to high-quality ones.

There have been some research efforts (Third, 2012; Fernandez-Breis et al., 2010; Quesada-Martınez et al., 2012) on exploring the structure of category names by building patterns. However, we automatically assign semantic types to the pattern variables (or called arguments) while they do not. For example, our template has the form of "city in [country]" while their patterns are like "city in [X]". More details are given in the related work section.

A similar task is query understanding, including query tagging and query template mining. Query tagging (Li et al., 2009; Reisinger and Pasca, 2011) corresponds to the category labeling stage described above. It is different from template generation because the results are for one query only, without merging the information of all queries to generate the final templates. Category template construction are slightly different from query template construction. First, some useful features such as query click-through is not available in category template construction. Second, categories should be valid natural language phrases, while queries need not. For example, "city Germany" is a query but not a valid category name. We discuss in more details in the related work section.

Our major contributions are as follows.

1) To the best of our knowledge, this is the first work of template generation specifically for categories in unsupervised manner.

2) We extract semantic knowledge and statistical information from a web corpus for improving template generation. Significant performance improvement is obtained in our experiments.

3) We study the characteristics of the scoring function from the viewpoint of probabilistic evidence combination and demonstrate that nonlinear functions are more effective in this task.

4) We employ the output templates to clean our category collection mined from the web, and get apparent quality improvement (precision improved from 81% to 89%).

After discussing related work in Section 2, we define the problem and describe one baseline approach in Section 3. Then we introduce our approach in Section 4. Experimental results are reported and analyzed in Section 5. We conclude the paper in Section 6.

2 Related work

Several kinds of work are related to ours.

Hypernymy relation extraction: Hypernymy relation extraction is an important task in text mining. There have been a lot of efforts (Hearst, 1992; Pantel and Ravichandran, 2004; Van Durme and Pasca, 2008; Zhang et al., 2011) in the literature to extract hypernymy (or is-a) relations from the web. Our target here is not hypernymy extraction, but discovering the semantic structure of hypernyms (or category names).

Category name exploration: Category name patterns are explored and built in some existing research work. Third (2012) proposed to find axiom patterns among category names on an existing ontology. For example, infer axiom pattern "SubClassOf(AB, B)" from "SubClassOf(junior_school school)" and "SubClassOf(domestic_mammal mammal)". Fernandez-Breis et al. (2010) and Quesada-Martınez et al. (2012) proposed to find lexical patterns in category names to define axioms (in medical domain). One example pattern mentioned in their papers is "[X] binding". They need manual intervention to determine what X means. The main difference between the above work and ours is that we automatically assign semantic types to the pattern variables (or called arguments) while they do not.

Template mining for IE: Some research work in information extraction (IE) involves patterns. Yangarber (2003) and Stevenson and Greenwood (2005) proposed to learn patterns which were in the form of [subject, verb, object]. The category names and learned templates in our work are not in this form. Another difference between our work and their work is that, their methods need a supervised name classifer to generate the candidate patterns while our approach is unsupervised. Chambers and Jurafsky (2011) leverage templates to describe an event while the templates in our work are for understanding category names (a kind of short text).

Query tagging/labeling: Some research work in recent years focuses on segmenting web search queries and assigning semantic tags to key segments. Li et al. (2009) and Li (2010) employed CRF (Conditional Random Field) or semi-CRF models for query tagging. A crowdsourcing-assisted method was proposed by Han et al. (2013) for query structure interpretation. These supervised or semi-supervised approaches require much manual annotation effort. Unsupervised methods were proposed by Sarkas et al. (2010) and Reisinger and Pasca (2011). As been discussed in the introduction section, query tagging is only one of the two stages of template generation. The tagging results are for one query only, without aggregating the global information of all queries to generate the final templates.

Query template construction: Some existing work leveraged query templates or patterns for query understanding. A semi-supervised random walk based method was proposed by Agarwal et al. (2010) to generate a ranked templates list which are relevant to a domain of interest. A predefined domain schema and seed information is needed for this method. Pandey and Punera (2012) proposed an unsupervised method based on graphical models to mine query templates. The above methods are either domain-specific (i.e., generating templates for a specific domain), or have some degree of supervision (supervised or semi-supervised). Cheung and Li (2012) proposed an unsupervised method to generate query templates by the aid of knowledge bases. An approach was proposed in (Szpektor et al., 2011) to improve query recommendation via query templates. Query session information (which is not available in our task) is needed in this approach for templates generation. Li et al. (2013) proposed an clustering algorithm to group existing query templates by search intents of users.

Compared to the open-domain unsupervised methods for query template construction, our approach improves on two aspects. First, we propose to incorporate multiple types of semantic knowledge (e.g., term peer similarity and term clusters) to improve template generation. Second, we propose a nonlinear template scoring function which is demonstrated to be more effective.

3 Problem Definition and Analysis

3.1 Problem definition

The goal of this paper is to construct a list of category templates from a collection of open-domain category names.

Input: The input is a collection of category names, which can either be manually compiled (like Wikipedia categories) or be automatically extracted. The categories used in our experiments were automatically mined from the web, by following existing work (Hearst, 1992, Pantel and Ravichandran 2004; Snow et al., 2005; Talukdar et al., 2008; Zhang et al., 2011). Specifically, we applied Hearst patterns (e.g., "NP [,] (such as | including) $\{NP,\}^*$ {and|or} NP") and is-a patterns ("NP (is|are|was|were|being) (a|an|the) NP") to a large corpus containing 3 billion English web pages. As a result, we obtained a term→hypernym bi-partite graph containing 40 million terms, 74 million hypernyms (i.e., category names), and 321 million edges (e.g., one example edge is "Berlin"→"city in Germany", where "Berlin" is a term and "city in Germany" is the corresponding hypernym). Then all the multi-word hypernyms are used as the input category collection.

Output: The output is a list of templates, each having a score indicating how likely it is valid. A template is a multi-word string with one headword and at least one argument. For example, in template "national holiday of [country]", "holiday" is the headword, and "[country]" is the argument. We only consider one-argument templates in this paper, and the case of multiple arguments is left as future work. A template is valid if it is syntactically and semantically correct. "CEO of [constellation]" (wrongly generated from "CEO of Delphinus", "CEO of Aquila", etc.) is not valid because it is semantically unreasonable.

3.2 Baseline approach

An intuitive approach to this task contains two stages: category labeling and template scoring. Figure 2 shows its workflow with simple examples.

3.2.1 Phase-1: Category labeling

At this stage, each category name is automatically segmented and labeled; and some *candidate template tuples* (CTTs) are derived based on the labeling results. This can be done in the following steps.

Category segmentation: Divide each category name into multiple segments (e.g., "holiday of South Africa" to "holiday + of + South Africa"). Each segment is one word or a phrase appearing in an entity dictionary. The dictionary used in this paper is comprised of all Freebase (www.freebase.com) entities.

Segment to hypernym: Find hypernyms for every segment (except for the headword and some trivial segments like prepositions and articles), by referring to a term→hypernym mapping graph. Following most existing query labeling work, we derive the term→hypernym graph from a dump of Freebase. Below are some examples of Freebase types (hypernyms),

German city (id: /location/de_city)

Italian province (id: /location/it_province)

Poem character (id: /book/poem_character)

Book (id: /book/book)

To avoid generating too fine-grained templates like "mayor of [Germany city]" and "mayor of [Italian city]" (semantically "mayor of [city]" is more desirable), we discard type modifiers and map terms to the headwords of Freebase types. For example, "Berlin" is mapped to "city". In this way, we build our basic version of term→hypernym mapping which contains 16.13 million terms and 696 hypernyms. Since "South Africa" is both a country and a book name in Freebase, hypernyms "country", "book", and others are assigned to the segment "South Africa" in this step.

CTT generation: Construct CTTs by choosing one segment (called the *target segment*) each time and replacing the segment with its hypernyms. An CTT is formed by the candidate template (with one argument), the target segment (as an *argument value*), and the tuple score (indicating tuple quality). Below are example CTTs obtained after the last segment of "holiday + of + South Africa" is processed,

U_1: (holiday of [country], South Africa, w_1)

U_2: (holiday of [book], South Africa, w_2)

3.2.2 Phase-2: Template scoring

The main objective of this stage is to merge all the CTTs obtained from the previous stage and to compute a final score for each template. In this stage, the CTTs are first grouped by the first element (i.e., the template string). For example, tuples for "holiday of [country]" may include,

U_1: (holiday of [country], South Africa, w_1)

U_2: (holiday of [country], Brazil, w_2)

U_3: (holiday of [country], Germany, w_3)

...

Then a scoring function is employed to calculate the template score from the tuple scores. Formally, given n tuples $\vec{U}$=(U_1, U_2..., U_n) for a template, the goal is to find a score fusion function $F(\vec{U})$ which yields large values for high-quality templates and small (or zero) values for invalid ones.

Borrowing the idea of TF-IDF from information retrieval, a reasonable scoring function is,

$$F(\vec{U}) = \sum_{i=1}^{n} w_i \cdot IDF(h) \tag{1}$$

where h is the argument type (i.e., the hypernym of the argument value) of each tuple. TF means the "term frequency" and IDF means the "inverse document frequency". An IDF function assigns lower scores to common hypernyms (like person and music track which contain a lot of entities). Let $DF(h)$ be the number of entities having hypernym h, we test two IDF functions in our experiments,

$$\begin{aligned} IDF_1(h) &= log\frac{1+N}{1+DF(h)} \\ IDF_2(h) &= 1/sqrt(DF(h)) \end{aligned} \tag{2}$$

where N is total number of entities in the entity dictionary.

The next problem is estimating tuple score w_i. Please note that there is no weight or score information in the term→hypernym mapping of Freebase. So we have to set w_i to be constant in the baseline,

$$w_i = 1 \tag{3}$$

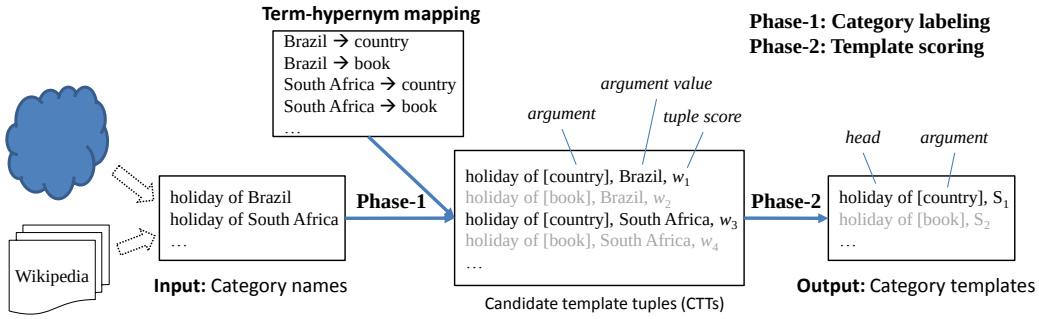


Figure 2: Problem definition and baseline approach.

4 Approach: Enhancing Template Scoring

In our approach, we follow the same framework as in the above baseline approach, and focus on improving the template scoring phase (i.e., phase-2).

We try three techniques: First, a better tuple score w_i is calculated in Section 4.1 by performing statistics on a large corpus. The corpus is a collection of 3 billion web pages crawled in early 2013 by ourselves. During this paper, we use "our web corpus" or "our corpus" to refer to this corpus.

Second, a nonlinear function is adopted in Section 4.2 to replace the baseline tuple fusion function (Formula 1). Third, we extract term peer similarity and term clusters from our corpus and use them as additional semantic knowledge to refine template scores.

4.1 Enhancing tuple scoring

Let's examine the following two template tuples,

U_1: (holiday of [country], South Africa, w_1)

U_2: (holiday of [book], South Africa, w_2)

Intuitively, "South Africa" is more likely to be a country than a book when it appears in text. So for a reasonable tuple scoring formula, we should have $w_1 > w_2$.

The main idea is to automatically calculate the popularity of a hypernym given a term, by referring to a large corpus. Then by adding the popularity information to (the edges of) the term→hypernym graph of Freebase, we obtain a *weighted* term→hypernym graph. The weighted graph is then employed to enhance the estimation of w_i.

For popularity calculation, we apply Hearst patterns (Hearst, 1992) and is-a patterns ("NP (is|are|was|were|being) (a|an|the) NP") to every sentence of our web corpus. For a (term, hypernym) pair, its popularity F is calculated as the number of sentences in which the term and the hypernym co-occur and also follow at least one of the patterns.

For a template tuple U_i with argument type h and argument value v, we test two ways of estimating the tuple score w_i,

$$w_i = \log\left(1 + F(v,h)\right) \tag{4}$$

$$w_i = \frac{F(v,h))}{\lambda + \sum_{h_j \in H} F(v,h_j)} \tag{5}$$

where $F(v,h)$ is the popularity of the (v, h) pair in our corpus, H is the set of all hypernyms for v in the weighted term→hypernym graph. Parameter λ (=1.0 in our experiments) is introduced for smoothing purpose. Note that the second formula is the conditional probability of hypernym h given term v.

Since it is intuitive to estimate tuple scores with their frequencies in a corpus, we treat the approach with the improved w_i as another baseline (our strong baseline).

4.2 Enhancing tuple combination function

Now we study the possibility of improving the tuple combination function (Formula 1), by examining the tuple fusion problem from the viewpoint of probabilistic evidence combination. We first demonstrate that the linear function in Formula 1 corresponds to the conditional independence assumption of the tuples. Then we propose to adopt a series of nonlinear functions for combining tuple scores.

We define the following events:

T: Template T is a valid template;

$\overline{T}$: T is an invalid template;

E_i: The observation of tuple U_i.

Let's compute the posterior odds of event T, given two tuples U_1 and U_2. Assuming E_1 and E_2 are conditionally independent given T or $\overline{T}$, according to the Bayes rule, we have,

$$\begin{aligned}\frac{P(T|E_1,E_2)}{P(\overline{T}|E_1,E_2)} &= \frac{P(E_1,E_2|T)\cdot P(T)}{P(E_1,E_2|\overline{T})\cdot P(\overline{T})} \\ &= \frac{P(E_1|T)}{P(E_1|\overline{T})}\cdot\frac{P(E_2|T)}{P(E_2|\overline{T})}\cdot\frac{P(T)}{P(\overline{T})} \\ = \frac{P(T|E_1)\cdot P(\overline{T})}{P(\overline{T}|E_1)\cdot P(T)}\cdot\frac{P(T|E_2)\cdot P(\overline{T})}{P(\overline{T}|E_2)\cdot P(T)}&\cdot\frac{P(T)}{P(\overline{T})}\end{aligned} \quad (6)$$

Define the log-odds-gain of T given E as,

$$G(T|E) = \log\frac{P(T|E)}{P(\overline{T}|E)} - \log\frac{P(T)}{P(\overline{T})} \quad (7)$$

Here G means the gain of the log-odds of T after E occurs. By combining formulas 6 and 7, we get

$$G(T|E_1,E_2) = G(T|E_1) + G(T|E_2) \quad (8)$$

It is easy to prove that the above conclusion holds true when $n > 2$, i.e.,

$$G(T|E_1,...,E_n) = \sum_{i=1}^{n} G(T|E_i) \quad (9)$$

If we treat $G(T|E_i)$ as the score of template T when only U_i is observed, and $G(T|E_1,...,E_n)$ as the template score after the n tuples are observed, then the above equation means that the combined template score should be the sum of $w_i \cdot IDF(h)$, which is exactly Formula 1. Please keep in mind that Equation 9 is based on the assumption that the tuples are conditional independent. This assumption, however, may not hold in reality. The case of conditional dependence was studied in (Zhang et al., 2011), where a group of nonlinear combination functions were proposed and achieved good performance in their task of hypernymy extraction. We choose p-Norm as our nonlinear fusion functions, as below,

$$F(\vec{U}) = \sqrt[p]{\sum_{i=1}^{n} w_i^p}\cdot IDF(h) \qquad (p>1) \quad (10)$$

where p (=2 in experiments) is a parameter.

Experiments show that the above nonlinear function performs better than the linear function of Formula 1. Let's use an example to show the intuition. Consider a good template "city of [country]" corresponding to CTTs $\vec{U}_A$ and a wrong template "city of [book]" having tuples $\vec{U}_B$. Suppose $|\vec{U}_A| = 200$ (including most countries in the world) and $|\vec{U}_B| = 1000$ (considering that many place names have already been used as book names). We observe that each tuple score corresponding to "city of [country]" is larger than the tuple score corresponding to "city of [book]". For simplicity, we assume each tuple in $\vec{U}_A$ has score 1.0 and each tuple in $\vec{U}_B$ has score 0.2. With the linear and nonlinear (p=2) fusion functions, we can get,

Linear:

$$\begin{aligned}F(\vec{U}_A) &= 200 * 1.0 = 200 \\ F(\vec{U}_B) &= 1000 * 0.2 = 200\end{aligned} \quad (11)$$

Nonlinear:

$$\begin{aligned}F(\vec{U}_A) &= 14.1 \\ F(\vec{U}_B) &= 6.32\end{aligned} \quad (12)$$

In the above settings the nonlinear function yields a much higher score for the good template (than for the invalid template), while the linear one does not.

4.3 Refinement with term similarity and term clusters

The above techniques neglect the similarity among terms, which has a high potential to improve the template scoring process. Intuitively, for a toy set {"city in Brazil", "city in South Africa","city in China", "city in Japan"}, since "Brazil", "South Africa", "China" and "Japan" are very similar to each other and they all have a large probability to be a "country", so we have more confidence that "city in [country]" is a good template. In this section, we propose to leverage the term similarity information to improve the template scoring process.

We start with building a large group of *small* and *overlapped* clusters from our web corpus.

4.3.1 Building term clusters

Term clusters are built in three steps.

Mining term peer similarity: Two terms are peers if they share a common hypernym and they are semantically correlated. For example, "dog" and "cat" should have a high peer similarity score. Following existing work (Hearst, 1992; Kozareva

et al., 2008; Shi et al., 2010; Agirre et al., 2009; Pantel et al., 2009), we built a peer similarity graph containing about 40.5 million nodes and 1.33 billion edges.

Clustering: For each term, choose its top-30 neighbors from the peer similarity graph and run a hierarchical clustering algorithm, resulting in one or multiple clusters. Then we merge highly duplicated clusters. The algorithm is similar to the first part of CBC (Pantel and Lin, 2002), with the difference that a very high merging threshold is adopted here in order to generate small and overlapped clusters. Please note that one term may be included in many clusters.

Assigning top hypernyms: Up to two hypernyms are assigned for each term cluster by majority voting of its member terms, with the aid of the weighted term→hypernym graph of Section 4.1. To be an eligible hypernym for the cluster, it has to be the hypernym of at least 70% of terms in the cluster. The score of each hypernym is the average of the term→hypernym weights over all the member terms.

4.3.2 Template score refinement

With term clusters at hand, now we describe the score refinement procedure for a template T having argument type h and supporting tuples $\vec{U}$=(U_1, U_2..., U_n). Denote $V = \{V_1, V_2, ..., V_n\}$ to be the set of argument values for the tuples (where V_i is the argument value of U_i).

By computing the intersection of V and every term cluster, we can get a distribution of the argument values in the clusters. We find that for a good template like "holiday in [country]", we can often find at least one cluster (one of the country clusters in this example) which has hypernym h and also contains many elements in V. However, for invalid templates like "holiday of [book]", every cluster having hypernym h (="book" here) only contains a few elements in V. Inspired by such an observation, our score refinement algorithm for template T is as follows,

Step-1. Calculating supporting scores: For each term cluster C having hypernym h, compute its supporting score to T as follows:

$$S(C, T) = k(C, V) \cdot w(C, h) \qquad (13)$$

where $k(C, V)$ is the number of elements shared by C and V, and $w(C, h)$ is hypernym score of h to C (computed in the last step of building clusters).

Step-2. Calculating the final template score: Let term cluster C^* has the maximal supporting score to T, the final template score is computed as,

$$S(T) = F(\vec{U}) \cdot S(C^*, T) \qquad (14)$$

where $F(\vec{U})$ is the template score before refinement.

5 Experiments

5.1 Experimental setup

5.1.1 Methods for comparison

We make a comparison among 10 methods.

SC: The method is proposed in (Cheung and Li, 2012) to construct templates from queries. The method firstly represents a query as a matrix based on Freebase data. Then a hierarchical clustering algorithm is employed to group queries having the same structure and meaning. Then an intent summarization algorithm is employed to create templates for each query group.

Base: The linear function in Formula 1 is adopted to combine the tuple scores. We use IDF_2 here because it achieves higher precision than IDF_1 in this setting.

LW: The linear function in Formula 1 is adopted to combine the tuple scores generated by Formula 4. IDF_1 is used rather than IDF_2 for better performance.

LP: The linear function in Formula 1 is adopted to combine the tuple scores generated by Formula 5. IDF_2 is used rather than IDF_1 for better performance.

NLW: The nonlinear fusion function in Formula 10 is used. Other settings are the same as LW.

NLP: The nonlinear fusion function in Formula 10 is used. Other settings are the same as LP.

LW+C, LP+C, NLW+C, NLP+C: All the settings of LW, LP, NLW, NLP respectively, with the refinement technology in Section 4.3 applied.

5.1.2 Data sets, annotation and evaluation metrics

The input category names for experiments are automatically extracted from a web corpus (Section 3.1). Two test-sets are built for evaluation from the output templates of various methods.

Subsets: In order to conveniently compare the performance of different methods, we create 20 sub-collections (called subsets) from the whole input category collection. Each subset contains all

the categories having the same headword (e.g., "symptom of insulin deficiency" and "depression symptom" are in the same subset because they share the same headword "symptom"). To choose the 20 headwords, we first sample 100 at random from the set of all headwords; then manually choose 20 for diversity. The headwords include symptom, school, food, gem, hero, weapon, model, etc. We run the 10 methods on these subsets and sort the output templates by their scores. Top-30 templates from each method on each subset are selected and mixed together for annotation.

Fullset: We run method NLP+C (which has the best performance according to our subsets experiments) on the input categories and sort the output templates by their scores. Then we split the templates into 9 sections according to their ranking position. The sections are: [1∼100], (100∼1K], (1K∼10K], (10K∼100K], (100K,120K], (120K∼140K], (140K∼160K], (160K∼180K], (180K∼200K]. Then 40 templates are randomly chosen from each section and mixed together for annotation.

The selected templates (from subsets and the fullset) are annotated by six annotators, with each template assigned to two annotators. A template is assigned a label of "good", "fair", or "bad" by an annotator. The percentage agreement between the annotators is 80.2%, with kappa 0.624.

For the subset experiments, we adopt Precision@k (k=10,20,30) to evaluate the top templates generated by each method. The scores for "good", "fair", and "bad" are 1, 0.5, and 0. The score of each template is the average annotation score over two annotators (e.g., if a template is annotated "good" by one annotator and "fair" by another, its score is (1.0+0.5)/2=0.75). The evaluation score of a method is the average over the 20 subsets. For the fullset experiments, we report the precision for each section.

5.2 Experimental results

5.2.1 Results for subsets

The results of each method on the 20 subsets are presented in Table 1. A few observations can be made. First, by comparing the performance of baseline-1 (Base) and the methods adopting term→hypernym weight (LW and LP), we can see big performance improvement. The bad performance of baseline-1 is mainly due to the lack of weight (or frequency) information on term→hypernym edges. The results demonstrate that edge scores are critical for generating high quality templates. Manually built semantic resources typically lack such kinds of scores. Therefore, it is very important to enhance them by deriving statistical data from a large corpus. Since it is relatively easy to have the idea of adopting a weighted term→hypernym graph, we treat LW and LP as another (stronger) baseline named baseline-2.

Method		P@10	P@20	P@30
Base (baseline-1)		0.359	0.361	0.358
SC (Cheung and Li, 2012)		0.382	0.366	0.371
Weighted (baseline-2)	LW	0.633	0.582	0.559
	LP	0.771	0.734	0.707
Nonlinear	NLW	0.711	0.671	0.638
	NLP	0.818	0.791	0.765
Term cluster	LW+C	0.813	0.786	0.754
	NLW+C	0.854	0.833	0.808
	LP+C	0.818	0.788	0.778
	NLP+C	0.868	0.839	0.788

Table 1: Performance comparison among the methods on subset.

As the second observation, the results show that the nonlinear methods (NLP and NLW) achieve performance improvement over their linear versions (LW and LP).

Third, let's examine the methods with template scores refined by term similarity and term clusters (LW+C, NLW+C, LP+C, NLP+C). It is shown that the refine-by-cluster technology brings additional performance gains on all the four settings (linear and nonlinear, two different ways of calculating tuple scores). So we can conclude that the peer similarity and term clusters are quite effective in improving template generation.

Fourth, the best performance is achieved when the three techniques (i.e., term→hypernym weight, nonlinear fusion function, and refine-by-cluster) are combined together. For instance, by comparing the P@20 scores of baseline-2 and NLP+C, we see a performance improvement of 14.3% (from 0.734 to 0.839). Therefore every technique studied in this paper has its own merit in template generation.

Finally, by comparing the method SC (Cheung and Li, 2012) with other methods, we can see that SC is slightly better than baseline-1, but has much lower performance than others. The major reason may be that this method did not employ a weighted term→hypernym graph or term peer similarity information in template construction.

		Base	SC	LP	NLP	LP+C
P@10	SC	$\sim$				
	LP	$>$ **	$>$ **			
	NLP	$>$ **	$>$ **	$>$		
	LP+C	$>$ **	$>$ **	$>$ **	$\sim$	
	NLP+C	$>$ **	$>$ **	$>$ **	$>$ **	$>$
		Base	**SC**	**LP**	**NLP**	**LP+C**
P@20	SC	$\sim$				
	LP	$>$ **	$>$ **			
	NLP	$>$ **	$>$ **	$>$ **		
	LP+C	$>$ **	$>$ **	$>$ **	$\sim$	
	NLP+C	$>$ **	$>$ **	$>$ **	$>$ **	$>$ **
		Base	**SC**	**LP**	**NLP**	**LP+C**
P@30	SC	$\sim$				
	LP	$>$ **	$>$ **			
	NLP	$>$ **	$>$ **	$>$ **		
	LP+C	$>$ **	$>$ **	$>$ **	$\sim$	
	NLP+C	$>$ **	$>$ **	$>$ **	$>$	$\sim$

Table 2: Paired t-test results on subsets.

		Base	SC	LW	NLW	LW+C
P@10	SC	$\sim$				
	LW	$>$ **	$>$ **			
	NLW	$>$ **	$>$ **	$>$ *		
	LW+C	$>$ **	$>$ **	$>$ **	$>$ **	
	NLW+C	$>$ **	$>$ **	$>$ **	$>$ **	$>$ *
		Base	**SC**	**LW**	**NLW**	**LW+C**
P@20	SC	$\sim$				
	LW	$>$ **	$>$ **			
	NLW	$>$ **	$>$ **	$>$ **		
	LW+C	$>$ **	$>$ **	$>$ **	$>$ **	
	NLW+C	$>$ **	$>$ **	$>$ **	$>$ **	$>$ **
		Base	**SC**	**LW**	**NLW**	**LW+C**
P@30	SC	$\sim$				
	LW	$>$ **	$>$ **			
	NLW	$>$ **	$>$ **	$>$ **		
	LW+C	$>$ **	$>$ **	$>$ **	$>$ **	
	NLW+C	$>$ **	$>$ **	$>$ **	$>$ **	$>$ **

Table 3: Paired t-test results on subsets.

Are the performance differences between methods significant enough for us to say that one is better than the other? To answer this question, we run paired two-tailed t-test on every pair of methods. We report the t-test values among methods in tables 2, 3 and 4.

The meaning of the symbols in the tables are,

$\sim$: The method on the row and the one on the column have similar performance.

$>$: The method on the row outperforms the method on the column, but the performance difference is not statistically significant ($0.05 \leq P < 0.1$ in two-tailed t-test).

$> *$: The performance difference is statistically significant ($P < 0.05$ in two-tailed t-test).

$> **$: The performance difference is statistically highly significant ($P < 0.01$ in two-tailed t-test).

	P@10	P@20	P@30
LP V.S. **LW**	$>$ **	$>$ **	$>$ **
NLP V.S. **NLW**	$>$ **	$>$ **	$>$ **
LP+C V.S. **LW+C**	$\sim$	$\sim$	$\sim$
NLP+C V.S. **NLW+C**	$\sim$	$\sim$	$\sim$

Table 4: Paired t-test results on subsets.

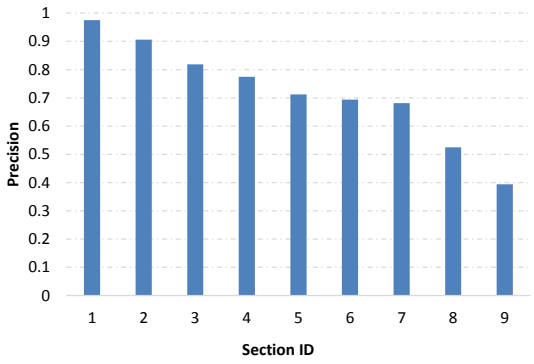


Figure 3: Precision by section in the fullset.

5.2.2 Fullset results

As described in the Section 5.1.2, for the *fullset* experiments, we conduct a section-wise evaluation, selecting 40 templates from each of the 9 sections of the NLP+C results. The results are shown in Figure 3. It can be observed that the precision for each section decreases when the section ID increases. The results indicate the effectiveness of our approach, since it can rank good templates in top sections and bad templates in bottom sections. According to the section-wise precision data, we are able to determine the template score threshold for choosing different numbers of top templates in different applications.

5.2.3 Templates for category collection cleaning

Since our input category collection is automatically constructed from the web, some wrong or invalid category names is inevitably contained. In this subsection, we apply our category templates to clean the category collection. The basic idea is that if a category can match a template, it is more likely to be correct. We compute a new score for every category name H as follows,

$$S_{new}(H) = \log(1 + S(H)) \cdot S(T^*) \qquad (15)$$

where $S(H)$ is the existing category score, determined by its frequency in the corpus. Here $S(T^*)$ is the score of template T^*, the best template (i.e., the template with the highest score) for the category.

Then we re-rank the categories according to their new scores to get a re-ranked category list. We randomly sampled 150 category names from the top 2 million categories of each list (the old list and the new list) and asked annotators to judge the

quality of the categories. The annotation results show that, after re-ranking, the precision increases from 0.81 to 0.89 (i.e., the percent of invalid category names decreases from 19% to 11%).

6 Conclusion

In this paper, we studied the problem of building templates for a large collection of category names. We tested three techniques (tuple scoring by weighted term→hypernym mapping, nonlinear score fusion, refinement by term clusters) and found that all of them are very effective and their combination achieves the best performance. By employing the output templates to clean our category collection mined from the web, we get apparent quality improvement. Future work includes supporting multi-argument templates, disambiguating headwords of category names and applying our approach to general short text template mining.

Acknowledgments

We would like to thank the annotators for their efforts in annotating the templates. Thanks to the anonymous reviewers for their helpful comments and suggestions. This work is supported in part by China National 973 program 2014CB340301 and NSFC grant 61379043.

References

Ganesh Agarwal, Govind Kabra, and Kevin Chen-Chuan Chang. 2010. Towards rich query interpretation: walking back and forth for mining query templates. In *Proceedings of the 19th international conference on World wide web*, pages 1–10. ACM.

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 976–986. Association for Computational Linguistics.

Jackie Chi Kit Cheung and Xiao Li. 2012. Sequence clustering and labeling for unsupervised query intent discovery. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 383–392. ACM.

Jesualdo Tomas Fernandez-Breis, Luigi Iannone, Ignazio Palmisano, Alan L Rector, and Robert Stevens. 2010. Enriching the gene ontology via the dissection of labels using the ontology pre-processor language. In *Knowledge Engineering and Management by the Masses*, pages 59–73. Springer.

Jun Han, Ju Fan, and Lizhu Zhou. 2013. Crowdsourcing-assisted query structure interpretation. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2092–2098. AAAI Press.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics - Volume 2*, COLING '92, pages 539–545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zornitsa Kozareva, Ellen Riloff, and Eduard H Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *ACL*, volume 8, pages 1048–1056.

Xiao Li, Ye-Yi Wang, and Alex Acero. 2009. Extracting structured information from user queries with semi-supervised conditional random fields. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 572–579. ACM.

Yanen Li, Bo-June Paul Hsu, and ChengXiang Zhai. 2013. Unsupervised identification of synonymous query intent templates for attribute intents. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2029–2038. ACM.

Xiao Li. 2010. Understanding the semantic structure of noun phrase queries. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1337–1345. Association for Computational Linguistics.

Sandeep Pandey and Kunal Punera. 2012. Unsupervised extraction of template structure in web search queries. In *Proceedings of the 21st international conference on World Wide Web*, pages 409–418. ACM.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619. ACM.

Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *HLT-NAACL*, volume 4, pages 321–328.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 938–947. Association for Computational Linguistics.

Manuel Quesada-Martınez, Jesualdo Tomás Fernández-Breis, and Robert Stevens. 2012. Enrichment of owl ontologies: a method for defining axioms from labels. In *Proceedings of the First International Workshop on Capturing and Refining Knowledge in the Medical Domain (K-MED 2012), Galway, Ireland*, pages 1–10.

Joseph Reisinger and Marius Pasca. 2011. Fine-grained class label markup of search queries. In *ACL*, pages 1200–1209.

Nikos Sarkas, Stelios Paparizos, and Panayiotis Tsaparas. 2010. Structured annotations of web queries. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 771–782. ACM.

Shuming Shi, Huibin Zhang, Xiaojie Yuan, and Ji-Rong Wen. 2010. Corpus-based semantic class mining: distributional vs. pattern-based approaches. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 993–1001. Association for Computational Linguistics.

Mark Stevenson and Mark A Greenwood. 2005. A semantic approach to ie pattern induction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 379–386. Association for Computational Linguistics.

Idan Szpektor, Aristides Gionis, and Yoelle Maarek. 2011. Improving recommendation for long-tail queries via templates. In *Proceedings of the 20th international conference on World wide web*, pages 47–56. ACM.

Allan Third. 2012. Hidden semantics: what can we learn from the names in an ontology? In *Proceedings of the Seventh International Natural Language Generation Conference*, pages 67–75. Association for Computational Linguistics.

Benjamin Van Durme and Marius Pasca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *AAAI*, volume 8, pages 1243–1248.

Roman Yangarber. 2003. Counter-training in discovery of semantic patterns. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 343–350. Association for Computational Linguistics.

Fan Zhang, Shuming Shi, Jing Liu, Shuqi Sun, and Chin-Yew Lin. 2011. Nonlinear evidence fusion and propagation for hyponymy relation mining. In *ACL*, volume 11, pages 1159–1168.

Taxonomy Construction Using Syntactic Contextual Evidence

Luu Anh Tuan [#1], **Jung-jae Kim** [#2], **Ng See Kiong** [*3]

[#] *School of Computer Engineering, Nanyang Technological University, Singapore*

[1] anhtuan001@e.ntu.edu.sg, [2] jungjae.kim@ntu.edu.sg

[*] *Institute for Infocomm Research, Agency for Science, Technology and Research, Singapore*

[3] skng@i2r.a-star.edu.sg

Abstract

Taxonomies are the backbone of many structured, semantic knowledge resources. Recent works for extracting taxonomic relations from text focused on collecting lexical-syntactic patterns to extract the taxonomic relations by matching the patterns to text. These approaches, however, often show low coverage due to the lack of contextual analysis across sentences. To address this issue, we propose a novel approach that collectively utilizes contextual information of terms in syntactic structures such that if the set of contexts of a term includes most of contexts of another term, a subsumption relation between the two terms is inferred. We apply this method to the task of taxonomy construction from scratch, where we introduce another novel graph-based algorithm for taxonomic structure induction. Our experiment results show that the proposed method is well complementary with previous methods of linguistic pattern matching and significantly improves recall and thus F-measure.

1 Introduction

Taxonomies that are backbone of structured ontology knowledge have been found to be useful for many areas such as question answering (Harabagiu et al., 2003), document clustering (Fodeh et al., 2011) and textual entailment (Geffet and Dagan, 2005). There have been an increasing number of hand-crafted, well-structured taxonomies publicly available, including WordNet (Miller, 1995), OpenCyc (Matuszek et al., 2006), and Freebase (Bollacker et al., 2008). However, the manual curation of those taxonomies is time-consuming and human experts may miss relevant terms. As such, there are still needs to extend existing taxonomies or even to construct new taxonomies from scratch.

The previous methods for identifying taxonomic relations (i.e. is-a relations) from text can be generally classified into two categories: statistical and linguistic approaches. The former includes co-occurrence analysis (Budanitsky, 1999), term subsumption (Fotzo and Gallinari, 2004) and clustering (Wong et al., 2007). The main idea behinds these techniques is that the terms that frequently co-occur may have taxonomic relationships. Such approaches, however, usually suffer from low accuracy, though relatively high coverage, and heavily depend on the choice of feature types and datasets. Most previous methods of the linguistic approach, on the other hand, rely on the lexical-syntactic patterns (e.g. *A is a B*, *A such as B*) (Hearst, 1992). Those patterns can be manually created (Kozareva et al., 2008; Wentao et al., 2012), chosen via automatic bootstrapping (Widdows and Dorow, 2002; Girju et al., 2003) or identified from machine-learned classifiers (Navigli et al., 2011). The pattern matching methods generally achieve high precision, but low coverage due to the lack of contextual analysis across sentences. In this paper, we introduce a novel statistical method and shows that when combined with a pattern matching method, it shows significant performance improvement.

The proposed statistical method, called syntactic contextual subsumption (SCS), compares the syntactic contexts of terms for the taxonomic relation identification, instead of the usage of bag-of-words model by the previous statistical methods. We observe that the terms in taxonomic relations may not occur in the same sentences, but in similar syntactic structures of different sentences, and that the contexts of a specific term are often found in the contexts of a general term but not vice versa. By context of a term, we mean the set of words frequently have a particular syntactic relation (e.g. *Subject-Verb-Object*) with the term in a

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 810–819,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

given corpus. Given two terms, the SCS method collects from the Web pre-defined syntactic relations of each of the terms and checks if the syntactic contexts of a term properly includes that of the other term in order to determine their taxonomic relation. The method scores each taxonomic relation candidate based on the two measures of Web-based evidence and contextual set inclusion, and as such, is able to find implicit subsumption relations between terms across sentences. The SCS shows itself (Section 3.1) to be complementary to linguistic pattern matching.

After the relation identification, the identified taxonomic relations should be integrated into a graph for the task of taxonomy construction from scratch or associated with existing concepts of a given taxonomy via is-a relations (Snow et al., 2006). In this step of taxonomic structure construction, there is a need for pruning incorrect and redundant relations. Previous methods for the pruning task (Kozareva and Hovy, 2010; Velardi et al., 2012) treat the identified taxonomic relations equally, and the pruning task is thus reduced to finding the best trade-off between path length and the connectivity of traversed nodes. This assumption, however, is not always true due to the fact that the identified taxonomic relations may have different confidence values, and the relations with high confidence values can be incorrectly eliminated during the pruning process. We thus propose a novel method for the taxonomy induction by utilizing the evidence scores from the relation identification method and the topological properties of the graph. We show that it can effectively prune redundant edges and remove loops while preserving the correct edges of taxonomy.

We apply the proposed methods of taxonomic relation identification and taxonomy induction to the task of constructing a taxonomy from a given text collection from scratch. The resultant system consists of three modules: Term extraction and filtering (Section 2.1), taxonomic relation identification (Section 2.2), and taxonomy induction (Section 2.3). The outputs of the term extraction/filtering module are used as inputs of the taxonomic relation identification, such that the taxonomic relation identification module checks if there is a taxonomic relation between each pair of terms from the term extraction/filtering module. The taxonomy induction module gets the identified taxonomic relation set as the input, and outputs the final optimal taxonomy by pruning redundant and incorrect relations.

2 Methodology

2.1 Term Extraction and Filtering

The first step to construct taxonomies is to collect candidate terms from text documents in the domain of interest. Like most of linguistic approaches, we use pre-defined linguistic filters to extract candidate terms, including single-word terms and multi-word terms which are noun or noun phrases in sentences. These terms are then preprocessed by removing determiners and lemmatization.

The candidate terms collected are then filtered to select the terms that are most relevant to the domain of interest. Many statistical techniques are developed for the filtering, such as $TF\text{-}IDF$, domain relevance (DR), and domain consensus (DC) (Navigli and Velardi, 2004). DR measures the amount of information that a term t captures within a domain of interest D_i, compared to other contrasting domains (D_j), while DC measures the distributed use of a term t across documents d in a domain D_i. Since three measures have pros and cons, and might be complementary to each other, our term filtering method is thus the linear combination of them:

$$\begin{aligned} TS(t, D_i) = \alpha \times TFIDF(t, D_i) \\ + \beta \times DR(t, D_i) + \gamma \times DC(t, D_i) \end{aligned} \quad (1)$$

We experimented (see Section 3) with different values of α, β and γ, and found that the method shows the best performance when the values for α and β are 0.2 and 0.8 and the value for γ is between 0.15 and 0.35, depending on the size of the domain corpus.

2.2 Taxonomic Relation Identification

In this section, we present three taxonomic relation identification methods which are adopted in our system. First, two methods of string inclusion with WordNet and lexical-syntactic pattern matching, which were commonly used in the literature will be introduced with some modifications. Then, a novel syntactic contextual subsumption method to find implicit relations between terms across sentences by using contextual evidence from syntactic structures and Web data will be proposed. Finally, these three methods will be linearly combined to

Notation	Meaning
$t_1 \gg t_2$	t_1 is a hypernym of t_2
$t_1 \approx t_2$	t_1 semantically equals or is similar to t_2
$t_1 \gg_{WN} t_2$	t_1 is a direct or inherited hypernym of t_2 according to WordNet
$t_1 \approx_{WN} t_2$	t_1 and t_2 belong to the same synset of WordNet

Table 1: Notations

form an integrating solution for taxonomic relation identification. Given two terms t_1 and t_2, Table 1 summarizes important notations used in this paper.

2.2.1 String Inclusion with WordNet (SIWN)

One simple way to check taxonomic relation is to test string inclusion. For example, "terrorist organization" is a hypernym of "foreign terrorist organization", as the former is a substring of the latter. We propose an algorithm to extend the string inclusion test by using WordNet, which will be named SIWN. Given a candidate general term t_g and a candidate specific term t_s, the SIWN algorithm examines t_g from left to right (designating each word in t_g to be examined as w_g) to check if there is any word (w_s) in t_s such that $w_g \approx_{WN} w_s$ or $w_g \gg_{WN} w_s$, and identifies the taxonomic relation between two terms if every word of t_g has a corresponding word in t_s (with at least one $\gg_{WN}$ relation). For example, consider two terms: "suicide attack" and "world trade center self-destruction bombing". Because "attack" $\gg_{WN}$ "bombing" and "suicide" $\approx_{WN}$ "self-destruction", according to SIWN algorithm, we conclude that "suicide attack" is the hypernym of "world trade center self-destruction bombing".

Given two terms t_1 and t_2, the evidence score for SIWN algorithm is calculated as follows:

$$Score_{SIWN}(t_1, t_2) = \begin{cases} 1 & \text{if } t1 \gg t_2 \text{ via SIWN} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

2.2.2 Lexical-syntactic Pattern

Extending the ideas of Kozareva and Hovy (2010) and Navigli et al. (2011), we propose a method of extracting taxonomic relations by matching lexical-syntactic patterns to the Web data.

Definition 1 (Syntactic patterns)**.** *Given two terms t_1 and t_2, Pat(t_1, t_2) is defined as the set of the following patterns:*

- *"t_1 such as t_2"*
- *"t_1, including t_2"*
- *"t_2 is [a|an] t_1"*
- *"t_2 is a [kind|type] of t_1"*
- *"t_2, [and|or] other t_1"*

, where t_1 and t_2 are replaced with actual terms and [a|b] denotes a choice between a and b.

Given candidate general term t_1 and candidate specific term t_2, the lexical-syntactic pattern (LSP) method works as follows:

1. Submit each phrase in *Pat*(t_1, t_2) to a Web search engine as a query. The number of the search results of the query is denoted as *WH*(t_1, t_2).
2. Calculate the following evidence score:

$$Score_{LSP}(t_1, t_2) = \frac{\log(WH(t_1, t_2))}{1 + \log(WH(t_2, t_1))} \quad (3)$$

3. If $Score_{LSP}(t_1, t_2)$ is greater than a threshold value then $t_1 \gg t_2$.

While most lexical-syntactic pattern methods in the literature only consider the value of *WH*(t_1, t_2) in checking $t_1 \gg t_2$ (Wentao et al., 2012), we take into account both *WH*(t_1, t_2) and *WH*(t_2, t_1). The intuition of formula (3) is that if $t1$ is a hypernym of $t2$ then the size of *WH*(t_1, t_2) will be much larger than that of *WH*(t_2, t_1), which means the lexical-syntactic patterns are more applicable for the ordered pair (t_1, t_2) than (t_2, t_1).

2.2.3 Syntactic Contextual Subsumption

The LSP method performs well in recognizing the taxonomic relations between terms in the sentences containing those pre-defined syntactic patterns. This method, however, has a major shortcoming: it cannot derive taxonomic relations between two terms occurring in two different sentences. We thus propose a novel syntactic contextual subsumption (SCS) method which utilizes contextual information of terms in syntactic structure (i.e. *Subject-Verb-Object* in this study) and Web data to infer implicit taxonomic relations

between terms across sentences. Note that the chosen syntactic structure *Subject-Verb-Object* is identical to the definition of non-taxonomic relations in the literature (Buitelaar et al., 2004), where the *Verb* indicate non-taxonomic relations between *Subject* and *Object*. In this subsection, we first present the method to collect those non-taxonomic relations. Then we present in detail the ideas of the SCS method and how we can use it to derive taxonomic relations in practice.

A. Non-taxonomic Relation Identification

Following previous approaches to non-taxonomic relation identification, e.g. (Ciaramita et al., 2005), we use the Stanford parser (Klein and Manning, 2003) to identify the syntactic structures of sentences and extract triples of *(Subject, Verb, Object)*, where *Subject* and *Object* are noun phrases.

We further consider the following issues: First, if a term (or noun phrase) includes a preposition, we remove the prepositional phrase. However, if the headword of a term is a quantitative noun like "lot", "many" or "dozen" and it is modified by the preposition "of", we replace it with the headword of the object of the preposition "of". For example, we can extract the triples $(people, need, food)$ and $(people, like, snow)$ from the following sentences, respectively:

- "People in poor countries need food"
- "A lot of people like snow"

Second, if the object of a verb is in a verb form, we replace it with, if any, the object of the embedded verb. For example, we can extract the triple $(soldier, attack, terrorist)$ from the following sentence:

- "The soldiers continue to attack terrorists"

Third, if a term has a coordinate structure with a conjunction like "and" or "or", we split it into all coordinated noun phrases and duplicate the triple by replacing the term with each of the coordinated noun phrases. For example, we can extract the triples of $R(girl, like, dog)$ and $R(girl, like, cat)$ from the following sentence:

- "The girl likes both dogs and cats"

Given two terms t_1, t_2 and a non-taxonomic relation r, some notations which will be used hereafter are shown below:

- $R(t_1, r, t_2)$: t_1, r, and t_2 have a *(Subject, Verb, Object)* triple.
- $\Theta(t_1, t_2)$: the set of relations r such that there exists $R(t_1, r, t_2)$ or $R(t_2, r, t_1)$.

B. Syntactic Contextual Subsumption Method

The idea of the SCS method derived from the following two observations.

Observation 1. *Given three terms t_1, t_2, t_3, and a non-taxonomic relation r, if we have two triples $R(t_1, r, t_3)$ and $R(t_2, r, t_3)$ (or $R(t_3, r, t_1)$ and $R(t_3, r, t_2)$), t_1 and t_2 may be in taxonomic relation.*

For example, given two triples *R(Al-Qaeda, attack, American)* and *R(Terrorist group, attack, American)*, a taxonomic relation *Terrorist group* $\gg$ *Al-Qaeda* can be induced. However, it is not always guaranteed to induce a taxonomic relations from such a pair of triples, for example from *R(animal, eat, meat)* and *R(animal, eat, grass)*. The second observation introduced hereafter will provide more chance to infer taxonomic relationship.

Definition 2 (Contextual set of a term)**.** *Given a term t_1 and a non-taxonomic relation r, $S(t_1, r, \text{“}subj\text{”})$ denotes the set of terms t_2 such that there exists triple $R(t_1, r, t_2)$. Similarly, $S(t_1, r, \text{“}obj\text{”})$ is the set of terms t_2 such that there exists triple $R(t_2, r, t_1)$.*

Observation 2. *Given two terms t_1, t_2, and a non-taxonomic relation r, if $S(t_1, r, \text{“}subj\text{”})$ mostly contains $S(t_2, r, \text{“}subj\text{”})$ but not vice versa, then most likely t_1 is a hypernym of t_2. Similarly, if $S(t_1, r, \text{“}obj\text{”})$ mostly contains $S(t_2, r, \text{‘}obj\text{”})$ but not vice versa, then most likely t_1 is a hypernym of t_2.*

For example, assume that *S(animal, eat, "subj")* = {*grass, potato, mouse, insects, meat, wild boar, deer, buffalo*} and *S(tiger, eat, "subj")* = {*meat, wild boar, deer, buffalo*}. Since *S(animal, eat, "subj")* properly contains *S(tiger, eat, "subj")*, we can induce *animal* $\gg$ *tiger*.

Based on Observation 2, our strategy to infer taxonomic relations is to first find the contextual set of terms via the evidence of syntactic structures and Web data, and then compute the score of the set inclusion. The detail of the method is presented hereafter.

Definition 3. *Given two terms t_1, t_2 and a non-taxonomic relation r, $C(t_1, t_2, r, \text{“}subj\text{”})$ denotes the number of terms t_3 such that there exists both triples $R(t_1, r, t_3)$ and $R(t_2, r, t_3)$. Similarly, $C(t_1, t_2, r, \text{“}obj\text{”})$ is the number of terms t_3 such that there exists both relations $R(t_3, r, t_1)$ and $R(t_3, r, t_2)$.*

Given the pair of a candidate general term t_1 and a candidate specific term t_2, we extract their non-taxonomic relations from corpora extracted from the Web, and use them to determine the taxonomic relation between t_1 and t_2 as follows:

1. Find from a domain corpus the relation r and type Γ such that:

$$C(t_1, t_2, r, \Gamma) = \max_{\substack{r' \in \Theta(t_1, t_2) \\ \Gamma' \in \{\text{“}subj\text{”}, \text{“}obj\text{”}\}}} C(t_1, t_2, r', \Gamma')$$

2. If type Γ is “subj”, collect the first 1,000 search results of the query “t_1 r” using the Google search engine, designated as $Corpus^{\Gamma}_{t_1}$. In the same way, construct $Corpus^{\Gamma}_{t_2}$ with the query “t_2 r”. If Γ is “obj”, two queries “r t_1” and “r t_2” are submitted instead to collect $Corpus^{\Gamma}_{t_1}$ and $Corpus^{\Gamma}_{t_2}$, respectively.

3. Find the sets of $S(t_1, r, \Gamma)$ and $S(t_2, r, \Gamma)$ from $Corpus^{\Gamma}_{t_1}$ and $Corpus^{\Gamma}_{t_2}$, respectively, using the non-taxonomic relation identification method above.

4. Calculate the following evidence score for SCS method:

$$Score_{SCS} = \left[\frac{|S(t_1, r, \Gamma) \bigcap S(t_2, r, \Gamma)|}{|S(t_2, r, \Gamma)|} + \left(1 - \frac{|S(t_1, r, \Gamma) \bigcap S(t_2, r, \Gamma)|}{|S(t_1, r, \Gamma)|}\right)\right] \times \log(|S(t_1, r, \Gamma)| + |S(t_2, r, \Gamma)|) \quad (4)$$

   The basic idea of the contextual subsumption score in our method is that if t_1 is a hypernym of t_2 then the set $S(t_1, r, \Gamma)$ will mostly contain $S(t_2, r, \Gamma)$ but not vice versa. The intuition of formula (5) is inspired by Jaccard similarity coefficient. We then multiply the score with the log value of total size of two sets to avoid the bias of small set inclusion.

5. If $Score_{SCS}(t_1, t_2)$ is greater than a threshold value, then we have $t1 \gg t2$.

2.2.4 Combined Method

In our study, we linearly combine three methods as follows:

1. For each ordered pair of terms (t_1, t_2) calculate the total evidence score:

$$\begin{aligned} Score(t_1, t_2) = \ & \alpha \times Score_{SIWN}(t_1, t_2) \\ & + \beta \times Score_{LSP}(t_1, t_2) \\ & + \gamma \times Score_{SCS}(t_1, t_2) \end{aligned} \quad (5)$$

2. If $Score(t_1, t_2)$ is greater than a threshold value, then we have $t_1 \gg t_2$.

We experimented with various combinations of values for α, β and γ, and found that the method shows the best performance when the value of α is 0.5, β is between 0.35 and 0.45, and γ is between 0.15 and 0.25, depending on the domain corpus size.

2.3 Taxonomy Induction

The output of the taxonomic relation identification module is a set of taxonomic relations T. In this section, we will introduce a graph-based algorithm (Algorithm 1) to convert this set into an optimal tree-structured taxonomy, as well as to eliminate incorrect and redundant relations. Denote $e(t_1, t_2)$ as an directed edge from t_1 to t_2, the algorithm consists of three steps which will be described hereafter with the corresponding lines in Algorithm 1.

Step 1: Initial hypernym graph creation (line 1 - 16) This step is to construct a connected directed graph from the list of taxonomic relations. The idea is to add each taxonomic relation $t_1 \gg t_2$ as a directed edge from parent node t_1 to child node t_2, and if t_1 does not have any hypernym term, t_1 will become a child node of $ROOT$ node. The result of this step is a connected graph containing all taxonomic relations with the common $ROOT$ node.

Step 2: Edge weighting (line 17) This step is to calculate the weight of each edge in the hypernym graph. Unlike the algorithm of Velardi et al. (2012) and Kozareva and Hovy (2010) where every taxonomic relation is treated equally, we assume the confidence of each taxonomic relation is different, depending on the amount of

Algorithm 1 Taxonomy Induction Algorithm

Input: T : the taxonomic relation set
Output: V: the vertex set of resultant taxonomy;
E: the edge set of resultant taxonomy;

1: Initialize $V = \{ROOT\}$, $E = \emptyset$;
2: **for** each taxonomic relation $(t_1 \gg t_2) \in T$ **do**
3: $E = E \cup \{e(t_1, t_2)\}$
4: **if** $t_1 \notin V$ **then**
5: $V = V \cup \{t_1\}$
6: **end if**
7: **if** $t_2 \notin V$ **then**
8: $V = V \cup \{t_2\}$
9: **end if**
10: **if** $\nexists\, e(t_3, t_1) \in E$ with $t_3 \neq ROOT$ **then**
11: $E = E \cup \{e(ROOT, t_1)\}$
12: **end if**
13: **if** $\exists\, e(ROOT, t_2) \in E$ **then**
14: $E = E \setminus \{e(ROOT, t_2)\}$
15: **end if**
16: **end for**
17: edgeWeighting(V, E);
18: graphPruning(V, E);

evidence it has. Thus, the hypernym graph edges will be weighted as follows:

$$w(e(t_1, t_2)) = \begin{cases} 1 & \text{if } t_1 = ROOT \\ Score(t_1, t_2) & \text{otherwise} \end{cases} \quad (6)$$

Note that the $Score$ value in formula (6) is determined by the taxonomic relation identification process described in Section 2.2.4.

Step 3: Graph pruning (line 18) The hypernym graph generated in Step 1 is not an optimal taxonomy as it may contain many redundant edges or incorrect edges which together form in a loop. In this step, we aim at producing an optimal taxonomy by pruning the graph based on our edge weighting strategy. A maximum spanning tree algorithm, however, cannot be applied as the graph is directed. For this purpose, we apply Edmonds' algorithm (Edmonds, 1967) for finding a maximum optimum branching of a weighted directed graph. Using this algorithm, we can find a subset of the current edge set, which is the optimized taxonomy where every non-root node has in-degree 1 and the sum of the edge weights is maximized. Figure 1 shows an example of the taxonomy induction process.

3 Experiment Results

We evaluated our methods for taxonomy construction against the following text collections of five domains:

- Artificial Intelligence (AI) domain: 4,119 papers extracted from the IJCAI proceedings from 1969 to 2011 and the ACL archives from year 1979 to 2010. The same dataset used in the work of Velardi et al. (2012).
- Terrorism domain: 104 reports of the US state department, titled "Patterns of Global Terrorism (1991-2002)" [1]. A report contains about 1,500 words.
- Animals, Plants and Vehicles domains: Collections of Web pages crawled by using the bootstrapping algorithm described by Kozareva et al. (2008). Navigli et al. (2011) and Kozareva and Hovy (2010) used these datasets to compare their outputs against WordNet sub-hierarchies.

There are two experiments performed in this section: 1) Evaluating the construction of new taxonomies for Terrorism and AI domains, and 2) Comparing our results with the gold-standard WordNet sub-hierarchies. Note that in the experiments, the threshold value we used for $Score_{LSP}$ is 1.9, $Score_{SCS}$ is 1.5 and $Score$ is 2.1.

3.1 Constructing new taxonomies for AI and Terrorism domains

Referential taxonomy structures such as WordNet or OpenCyc are widely used in semantic analytics applications. However, their coverage is limited to common well-known areas, and many specific domains like Terrorism and AI are not well covered in those structures. Therefore, an automatic method which can induce taxonomies for those specific domains from scratch can greatly contribute to the process of knowledge discovery.

First, we applied our taxonomy construction system to the AI domain corpus. We compared the taxonomy constructed by our system with that obtained by Velardi et al. (2012), and show the comparison results in Table 2. Notice that in this comparison, to be fair, we use the same set of terms that was used in (Velardi et al., 2012). The result shows that our approach can extract 9.8%

[1] *http://www.fas.org/irp/threat/terror.htm*

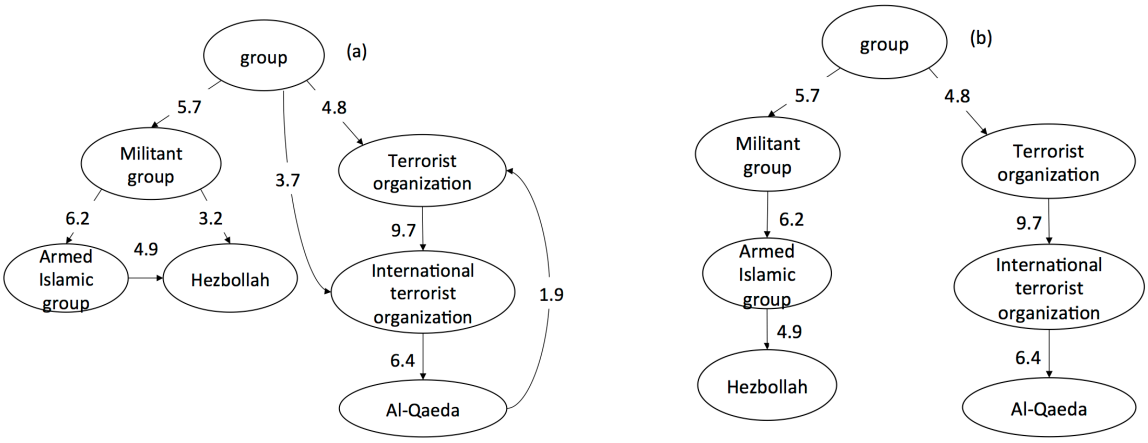


Figure 1: An example of taxonomy induction. (a) Initial weighted hypernym graph. (b) Final optimal taxonomy, where we prune two redundant edges *(group, International terrorist organization)*, *(Militant group, Hezbollah)* and remove the loop by cutting an incorrect edge *(Al-Qaeda, Terrorist organization)*.

more taxonomic relations and achieve 7% better term coverage than Velardi's approach.

	Our system	Velardi's system
#vertex	1839	1675
#edge	1838	1674
Average depth	6.2	6
Max depth	10	10
Term coverage	83%	76%

Table 2: Comparison of our system with (Velardi et al., 2012)

We also applied our system to the Terrorism corpus. The proposed taxonomic relation identification algorithm extracts a total of 976 taxonomic relations, from which the taxonomy induction algorithm builds the optimal taxonomy. The total number of vertices in the taxonomy is 281, and the total number of edges is 280. The average depth of the trees is 3.1, with the maximum depth 6. In addition, term coverage (the ratio of the number of terms in the final optimal trees to the number of terms obtained by the term suggestion/filtering method) is 85%.

To judge the contribution of each of taxonomic relation identification methods described in Section 2.2 to the overall system, we alternately run the system for the AI and Terrorism domains with different combinations of the three methods (i.e. SIWN, LSP, and SCS) as shown in Table 3. Note that we employed only the first two modules of term suggestion/filtering and taxonomic relation identification except the last module of taxonomy induction for this experiment. Table 3 shows the number of the taxonomic relations extracted by each of the combinations. Since SIWN and LSP are commonly used by previous taxonomic relation identification systems, we consider the combination of SIWN + LSP as the baseline of the experiment. The results in Table 3 show that the three methods are all well complementary to each other. In addition, the proposed SCS method can contribute up to about 27% - 29% of all the identified taxonomic relations, which were not discovered by the other two baseline methods.

	No. of extracted relations	
	Terrorism	AI domain
SCS	484	1308
SIWN	301	984
LSP	527	1537
SIWN + LSP	711	2203
SCS + SIWN + LSP	976	3122

Table 3: The number of taxonomic relations extracted by different methods.

	Percentage of correct relations	
	Terrorism	AI domain
SCS	91%	88%
SIWN	96%	91%
LSP	93%	93%
SCS + SIWN + LSP	92%	90%

Table 4: Estimated precision of taxonomic relation identification methods in 100 extracted relations.

	Animals domain			Plants domain			Vehicles domain		
	Our	Kozareva	Navigli	Our	Kozareva	Navigli	Our	Kozareva	Navigli
#Correct relations	2427	1643	N.A.	1243	905	N.A.	281	246	N.A.
Term coverage	96%	N.A.	94%	98%	N.A.	97%	97%	N.A.	96%
Precision	95%	98%	97%	95%	97%	97%	93%	99%	91%
Recall	56%	38%	44%	53%	39%	38%	69%	60%	49%
F-measure	71%	55%	61%	68%	56%	55%	79%	75%	64%

Table 5: Comparison of (Navigli et al., 2011), (Kozareva and Hovy, 2010) and our system against WordNet in three domains: Animals, Plants and Vehicles.

We further evaluated the precision of each individual taxonomic relation identification method. For AI and Terrorism domains, we again run the system with each of the three methods and with all together, and then randomly select 100 extracted taxonomic relations each time. These selected taxonomic relations are then examined by two domain experts to check the correctness. The evaluation results are given in Table 4. Note that only the first two modules of term suggestion/filtering and taxonomic relation identification are employed for this experiment as well. The SIWN and LSP methods achieve high precision because they are based on the gold-standard taxonomy hierarchy WordNet and on the well-defined patterns, respectively. In contrast, the SCS method ambitiously looks for terms pairs that share similar syntactic contexts across sentences, though the contextual evidence is restricted to certain syntactic structures, and thus has a slightly lower precision compared to the other two methods.

In short, the SCS method is complementary to the baseline methods, significantly improving the coverage of the combined methods, when its precision is comparable to those of the baseline methods. We performed next experiments to show that the SCS method overall has synergistic impact to improve the F-measure of the combined methods.

3.2 Evaluation against WordNet

In this experiment, we constructed taxonomies for three domains Animals, Plants and Vehicles, and then checked whether the identified relations can be found in the WordNet, and which relations in WordNet are not found by our method. Note that in this comparison, to be fair, we changed our algorithm to avoid using WordNet in identifying taxonomic relations. Specifically, in the SIWN algorithm, all operations of "$\approx_{WN}$" are replaced with normal string-matching comparison, and all "$\gg_{WN}$" relations are falsified. The evaluation uses the following measures:

$$Precision = \frac{\#relations\ found\ in\ WordNet\ and\ by\ the\ method}{\#relations\ found\ by\ the\ method}$$

$$Recall = \frac{\#relations\ found\ in\ WordNet\ and\ by\ the\ method}{\#relations\ found\ in\ WordNet}$$

We also compared our results with those obtained by the approaches of Navigli et al. (2011) and Kozareva and Hovy (2010), where they also compared their resultant taxonomies against WordNet. In this comparison, all the three approaches (i.e. ours, the two previous methods) use the same corpora and term lists. The comparison results are given in Table 5. "$N.A.$" value means that this parameter is not applicable to the corresponding method. The results show that our approach achieves better performance than the other two approaches, in terms of both the number of correctly extracted taxonomic relations and the term coverage. Our system has a slightly lower precision than that of (Navigli et al., 2011) and (Kozareva and Hovy, 2010) due to the SCS method, but it significantly contributes to improve the recall and eventually the F-measure over the other two systems.

To judge the effectiveness of our proposed taxonomy induction algorithm described in Section 2.3, we compared it with the graph-based algorithm of Velardi et al. (2012). Recall that in this algorithm, they treat all taxonomic relations equally, and the pruning task is reduced to finding the best trade-off between path length and the connectivity of traversed nodes. For each of five domains (i.e. Terrorism, AI, Animals, Plants and Vehicles), we alternately run the two taxonomy induction algorithms over the same taxonomic relation set produced by our taxonomic relation identification process. For Terrorism and AI domains, we randomly pick up 100 edges in each resultant taxon-

omy and ask two domain experts to judge for the correctness. For Animals, Plants and Vehicles domains, we check the correctness of the edges in resultant taxonomies by comparing them against the corresponding sub-hierarchies in WordNet. The evaluation is given in Table 6. The results show that the proposed taxonomy induction algorithm can achieve better performance than the algorithm of Velardi et al. (2012). This may be due to the fact that our algorithm considers the scores of the identified taxonomic relations from the relation identification module, and thus is more precise in eliminating incorrect relations during the pruning process.

	Percentage of correct edges	
	Our algorithm	Velardi's algorithm
Terrorism	94%	90%
AI	93%	88%
Animals	95%	93%
Plants	95%	92%
Vehicles	93%	92%

Table 6: Comparison of our taxonomy induction algorithms and that of Velardi et al. (2012).

In addition, when comparing Tables 4 and 6, we can find that the precision of taxonomic relations after the pruning process is higher than that before the pruning process, which proves that the proposed taxonomy induction algorithm effectively trims the incorrect relations of Terrorism and AI taxonomies, leveraging the percentage of correct relations 2% - 3% up.

For the SCS method, besides the triple *Subject-Verb-Object*, we also explore other syntactic structures like *Noun-Preposition-Noun* and *Noun-Adjective-Noun*. For example, from the sentence *"I visited Microsoft in Washington"*, the triple *(Microsoft, in, Washington)* is extracted using *Noun-Preposition-Noun* structure. Similarly, from the sentence *"Washington is a beautiful city"*, the triple *(Washington, beautiful, city)* is extracted using *Noun-Adjective-Noun* structure. We then use the triples for the contextual subsumption method described in Section 2.2.3, and test the method against the Animals, Plants and Vehicles domains. The results are then compared against WordNet sub hierarchies. The experiment results in Table 7 show that the triples of *Subject-Verb-Object* give the best performance compared to the other syntactic structures. These can be explained as the number of triples of two types *Noun-Preposition-Noun* and *Noun-Adjective-Noun* are smaller than that of *Subject-Verb-Object*, and the number of *Verb* is much greater than number of *Preposition* or *Adjective*.

	S-V-O	*N-P-N*	*N-A-N*
Animals domain			
Precision	95%	68%	72%
Recall	56%	52%	47%
F-measure	71%	59%	57%
Plants domain			
Precision	95%	63%	66%
Recall	53%	41%	43%
F-measure	68%	50%	52%
Vehicles domain			
Precision	93%	59%	60%
Recall	69%	45%	48%
F-measure	79%	51%	53%

Table 7: Comparison of three syntactic structures: *S-V-O* (*Subject-Verb-Object*), *N-P-N* (*Noun-Preposition-Noun*) and *N-A-N* (*Noun-Adjective-Noun*).

All experiment results are available at *http://nlp.sce.ntu.edu.sg/wiki/projects/taxogen*.

4 Conclusion

In this paper, we proposed a novel method of identifying taxonomic relations using contextual evidence from syntactic structure and Web data. This method is proved well complementary with previous method of linguistic pattern matching. We also present a novel graph-based algorithm to induce an optimal taxonomy from a given taxonomic relation set. The experiment results show that our system can generally achieve better performance than the state-of-the-art methods. In the future, we will apply the proposed taxonomy construction method to other domains such as biomedicine and integrate it into other frameworks such as ontology authoring.

References

K. Bollacker, C. Evans, P. Paritosh, T. Sturge and J. Taylor. 2008. *Freebase: a collaboratively created graph database for structuring human knowledge.* In proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 1247-1250.

A. Budanitsky. 1999. *Lexical semantic relatedness*

and its application in natural language processing. Technical Report CSRG-390, Computer Systems Research Group, University of Toronto.

P. Buitelaar, D. Olejnik and M. Sintek. 2004. *A Protégé Plug-in for Ontology Extraction from Text Based on Linguistic Analysis.* In proceedings of the 1st European Semantic Web Symposium, pp. 31-44.

M. Ciaramita, A. Gangemi, E. Ratsch, J. Saric and I. Rojas. 2005. *Unsupervised Learning of Semantic Relations Between Concepts of a Molecular Biology Ontology.* In proceedings of the 19th International Joint Conference on Artificial Intelligence, pp. 659-664.

J. Edmonds. 1967. *Optimum branchings.* Journal of Research of the National Bureau of Standards, 71, pp. 233-240.

S. Fodeh, B. Punch and P. N. Tan. 2011. *On Ontology-driven Document Clustering Using Core Semantic Features.* Knowledge and information systems, 28(2), pp. 395-421.

H. N. Fotzo and P. Gallinari. 2004. *Learning "Generalization/Specialization" Relations between Concepts - Application for Automatically Building Thematic Document Hierarchies.* In proceedings of the 7th International Conference on Computer-Assisted Information Retrieval.

M. Geffet and I. Dagan. 2005. *The Distributional Inclusion Hypotheses and Lexical Entailment.* In proceedings of the 43rd Annual Meeting of the ACL, pp. 107-114.

R. Girju, A. Badulescu, and D. Moldovan. 2003. *Learning Semantic Constraints for the Automatic Discovery of Part-Whole Relations.* In proceedings of the NAACL, pp. 1-8.

S. M. Harabagiu, S. J. Maiorano and M. A. Pasca. 2003. *Open-Domain Textual Question Answering Techniques.* Natural Language Engineering, 9(3): pp. 1-38.

M. A. Hearst. 1992. *Automatic Acquisition of Hyponyms from Large Text Corpora.* In proceedings of the 14th Conference on Computational Linguistics, pp. 539-545.

D. Klein and C. D. Manning. 2003. *Accurate Unlexicalized Parsing.* In proceedings of the 41st Annual Meeting of the ACL, pp. 423-430.

Z. Kozareva, E. Riloff, and E. H. Hovy. 2008. *Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs.* In proceedings of the 46th Annual Meeting of the ACL, pp. 1048-1056.

Z. Kozareva and E. Hovy. 2010. *A Semi-supervised Method to Learn and Construct Taxonomies Using the Web.* In proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1110-1118.

C. Matuszek, J. Cabral, M. J. Witbrock and J. DeOliveira. 2006. *An Introduction to the Syntax and Content of Cyc.* In proceedings of the AAAI Spring Symposium: Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering, pp. 44-49.

G. A. Miller. 1995. *WordNet: a Lexical Database for English.* Communications of the ACM, 38(11), pp. 39-41.

R. Navigli and P. Velardi, 2004. *Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites.* Computational Linguistics, 30(2), pp. 151-179.

R. Navigli, P. Velardi and S. Faralli. 2011. *A Graph-based Algorithm for Inducing Lexical Taxonomies from Scratch.* In proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 1872-1877.

R. Snow, D. Jurafsky and A. Y. Ng. 2006. *Semantic Taxonomy Induction from Heterogenous Evidence.* In proceedings of the 21st International Conference on Computational Linguistics, pp. 801-808.

P. Velardi, S. Faralli and R. Navigli. 2012. *Ontolearn Reloaded: A Graph-based Algorithm for Taxonomy Induction.* Computational Linguistics, 39(3), pp. 665-707.

W. Wentao, L. Hongsong, W. Haixun, and Q. Zhu. 2012. *Probase: A probabilistic taxonomy for text understanding.* In proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 481-492.

D. Widdows and B. Dorow. 2002. *A Graph Model for Unsupervised Lexical Acquisition.* In proceedings of the 19th International Conference on Computational Linguistics, pp. 1-7.

W. Wong, W. Liu and M. Bennamoun. 2007. *Tree-traversing ant algorithm for term clustering based on featureless similarities.* Data Mining and Knowledge Discovery, 15(3), pp. 349-381.

Analysing recall loss in named entity slot filling

Glen Pink Joel Nothman James R. Curran
ə-lab, School of Information Technologies
University of Sydney
NSW 2006, Australia
{glen.pink,joel.nothman,james.r.curran}@sydney.edu.au

Abstract

State-of-the-art fact extraction is heavily constrained by recall, as demonstrated by recent performance in TAC Slot Filling. We isolate this recall loss for NE slots by systematically analysing each stage of the slot filling pipeline as a filter over correct answers. Recall is critical as candidates never generated can never be recovered, whereas precision can always be increased in downstream processing.

We provide precise, empirical confirmation of previously hypothesised sources of recall loss in slot filling. While NE type constraints substantially reduce the search space with only a minor recall penalty, we find that 10% to 39% of slot fills will be entirely ignored by most systems. One in six correct answers are lost if coreference is not used, but this can be mostly retained by simple name matching rules.

1 Introduction

The TAC Knowledge Base Population (KBP) Slot Filling (SF) consists of extracting named attributes from text. Given a query, e.g. John Kerry, a system searches a corpus for documents which contain the entity. It then fills a list of *slots*, named attributes such as (`per:spouse`, Teresa Heinz).

The top TAC SF 2013 (TAC13) system scored 37.3% F-score (Roth et al., 2013), and the median F-score was 16.9% (Surdeanu, 2013). Recall for SF systems is especially low, with many systems using precise extractors with low recall. Precision ranges from 9% to 40% *greater than recall* for the top 5 systems in TAC13, and unsurprisingly, Roth et al. (2013) has the highest recall at 33%. Closing the recall gap without substantially increasing the search space is critical to improving SF results.

Ji and Grishman (2011) and Min and Grishman (2012) identify many of the challenges of SF, and suggest that inference, coreference and named entity recognition (NER) are key sources of error. Min and Grishman categorise the slot fills found by human annotators but not found in the aggregated output of all systems. However, this approach only allows them to hypothesise the likely source of recall loss. For instance, it is impossible to distinguish candidate generation errors from answer merging errors. Roth et al. (2014) categorise these errors at a high level, without specific analysis of candidate generation pipeline components such as coreference.

In this paper, we take this analysis further by performing a systematic recall analysis that allows us to pinpoint the cause of every recall error (candidates lost that can never be recovered) and estimate upper bounds on recall in existing approaches. We implement a collection of naïve SF systems utilizing a set of increasingly restrictive filters over documents and named entities (NEs). TAC has three slot types: NE, string and value slots. We consider only those slots filled by NEs as there are widely-used, high accuracy tools available for NER, and focusing on NEs only allows us to precisely gauge performance of filters. String slots do not have reliable classifiers, and value slots require more normalisation than directly returning a token span. Otherwise, this evaluation is not specifically dependent on the nature of NEs, and we expect similar results for other slot types.

We focus on systems which first generate candidates and then process them, the approach of the majority of TAC systems. Our filters apply hard constraints over NEs commonly used in the literature, accounting for a typical SF candidate generation pipeline—matching the query term, the form of candidate fills and the distance between the query and the candidate—but not performing any further scoring or thresholding. We compare sev-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 820–830,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

eral forms of coreference as filters, motivated by the need for efficient coreference resolution when processing large corpora. Complementing these unsupervised experiments, we implement a maximum recall bootstrap to identify which fills are reachable from training data.

We find ∼10% of recall is ignored by most systems due to NER bounds errors, and despite state-of-the-art coreference, 8% is lost when queries and fills occur in different sentences. Using NE type constraints is very effective, reducing recall by only 2% for a search space reduction of 81%. Without any coreference, 16% of typed fills are lost, but 12% of this recall can be recovered using fast naïve name matching rules, reducing the search space to 59% that of full coreference. 15% of recall is lost if a SF approach, such as a bootstrapping, requires that dependency paths be non-unique in a corpus. We show that most remaining candidates are reachable via bootstrapping from a small number of seeds. Our results provide systematic confirmation that effective coreference and NER are critical to high recall slot filling.

2 Why focus on recall?

In this work, we determine the recall loss caused by candidate generation constraints in SF systems. SF pipelines are typically implemented using a coarse-to-fine approach, where all possible candidates are generated and then filtered by hard constraints and more sophisticated downstream processes. Following this, we maximally generate candidates and assume a high-precision but relatively costly downstream process selects the final extractions. While ultimately any system makes precision-recall trade-offs, the recall of a system's coarse candidate generation process sets a hard upper bound on performance, as candidates that are not generated at all can never be recovered by downstream processes. SF systems could generate every noun phrase in a corpus as potential candidates, but they apply hard candidate generation constraints for efficiency and precision.

We implement these hard constraints as a series of filters, and return every candidate which passes a filter without further ranking or thresholding. These filters are comprised of generic components, such as NER, which are representative of SF pipelines. We are only interested in precision in so much as it corresponds to the size of the search space (the candidates generated), assuming a small, fixed number of answers. The search space determines the workload of later stages responsible for extraction, merging and ranking. Precision can be improved by this post-processing of the candidate set, but recall cannot.

3 Background

Slot filling (SF) is a query-oriented relation extraction (RE) task in the Knowledge Base Population (KBP) track of the Text Analysis Conferences (TAC) (McNamee and Dang, 2009). A SF system is queried with a name and a predefined relation schema, or *slots*, and must seek instances of any relations involving the query entity, and the corresponding slot fills, from a corpus.

Systems typically consist of several pipelined stages (Ji et al., 2011), providing many potential locations for error. The basic pipeline, in Figure 1, consists of four stages (Ji and Grishman, 2011): document retrieval, candidate generation, answer extraction, and answer merging and ranking. The output of the second stage is a set of candidates which are then usually ranked using RE techniques,[1] to precisely pinpoint answers. TAC penalises redundant responses, requiring a final answer merging and ranking stage. The first two stages are the focus of this work, as they inadvertently filter correct answers that cannot be recovered, and they determine the size of the search space for later stages.

Min and Grishman (2012) conducted an analysis of the 140 TAC 2010 SF fills that were found by human annotators but not any system, and manually look for evidence in the reference document and categorise the hypothetical sources of error. They find inference, coreference and NER to be the top sources of error, and that the most studied component (sentence-level RE) is not the dominant problem, contributing only 10% of recall loss. We precisely characterise the contribution of these sources of error.

We follow the SF literature in adopting RE techniques for filtering candidates. RE focuses on identifying relations between entities (or attributes of entities) as mentioned in text. Both relation schema and training data are often provided, and extraction is done using learnt classifiers (Mintz et al., 2009; Surdeanu et al., 2012; Riedel et al.,

[1] We note that question answering techniques have been used directly by SF systems (Byrne and Dunnion, 2011) but RE techniques are the primary method for answer extraction.

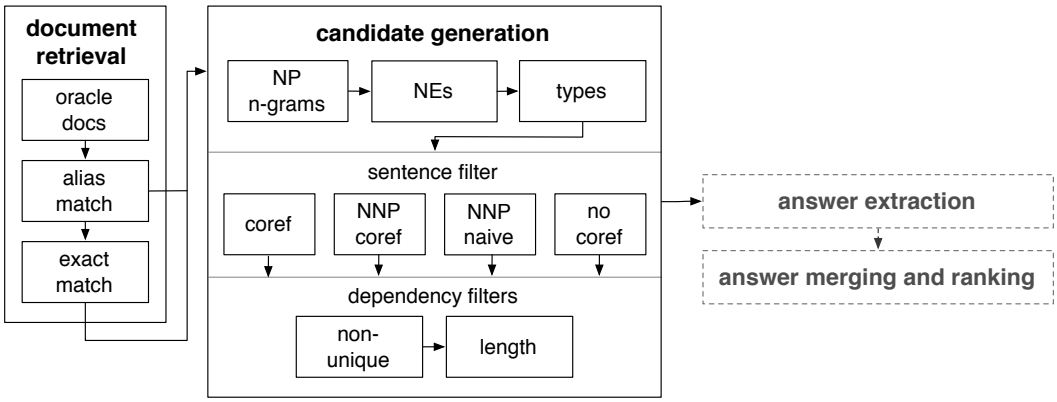


Figure 1: Candidate filters within the standard SF pipeline. Arrows indicate a sequence of filters.

2013; Zhang et al., 2013) or semi-supervised techniques (Agichtein and Gravano, 2000; Wang et al., 2011; Carlson et al., 2010).

Relation phrases or patterns may be identified without labels (Fader et al., 2011; Mausam et al., 2012) or clustered (Yao et al., 2012) into types. Generating candidate entity pairs and using the syntactic or surface path between them to decide whether a relation exists are common threads in RE that also form part of the SF pipeline. In some RE tasks, entities mentioned may already be identified in a document and provided to a RE system; in general, automatic NER is required. Some tasks are defined more generally to include common noun phrases (Fader et al., 2011; Carlson et al., 2010). SF specifically includes slots that can be filled by arbitrary strings such as `per:cause of death`, which make up a large number of slot fills but may require the use of different techniques for extraction, separate from names. NER may be further enhanced by resolving names to a KB (Mintz et al., 2009; Hoffmann et al., 2011; Surdeanu et al., 2012; Wang et al., 2011), reducing noise in learning and extraction processes, but we do not take this step in this work.

Typically, a RE system will only consider entities mentioned together in a sentence. When seeking all instances of a given relation between known entities, coreference resolution is necessary to substantially expand the set of candidate pairs (Gabbard et al., 2011). Coreference resolution may not be necessary where each relation is redundantly mentioned in a large corpus, as in SF; in this vein, "Open" approaches prefer precision and avoid automatic coreference resolution (Banko et al., 2007). Moreover, previous analysis attributed substantial SF error to these tools (Ji and Grishman, 2011). Our work evaluates NER, locality heuristics and coreference within a SF context.

Classification features for RE typically encode: attributes of the entities; the surface form, dependency path, or phrase structure subtree between them; and surrounding context (Zhou et al., 2005; Mintz et al., 2009; Zhang et al., 2013). We evaluate the length of dependency path between entities as a variable affecting SF candidate recall, and apply naïve entity pair bootstrapping (Brin, 1998; Agichtein and Gravano, 2000) to assess the generalisation over dependency paths from examples.

4 Experimental setup

We begin with a set of queries (a query being a NE entity grounded in a mention in a document) and, for each query q, the documents D_q known to contain any slot fill for q, as determined by oracle information retrieval (IR) from human annotation and judged system output. Filling every slot in q with every n-gram in D_q constitutes a system with nearly perfect recall. We apply a series of increasingly restrictive filters over this set. As in Figure 1, SF systems in practice must retrieve relevant documents and generate candidates. We propose filters that allow for analysis of recall lost during these stages. We ignore the remaining stages and evaluate the set of candidates directly.

Filters define what documents or NEs are allowed to pass through, based on constraints imposed by query matching, entity form, and sentence and syntactic context. We combine these filters in series in a number of configurations. The use or absence of coreference varies across our configurations, as the need to identify the query mention and terms that refer to the query mention is critical. Finally, we experiment with a boot-

strapping training process, to reflect constraints implicitly applied by a training approach.

The SF typical system pipeline presented in Section 3 applies to most, but not all SF approaches. The following filters directly apply only to systems that use NER as the method of candidate generation, and where candidate generation is distinct from answer extraction. Fourteen of the eighteen teams participating in TAC13 submitted system reports (Surdeanu, 2013). Eleven of these systems identify NEs with NER and pass these to an answer extraction process. The remaining three systems either do not document whether they rely on or do not rely on NER for candidate generation for `name` slots. We include a high recall baseline based on noun phrases (NPs) to cover these systems.

4.1 Filters

The first step in the SF pipeline is to find a relevant document and the query entity mentioned within that document. We use oracle IR to find documents D_q (ORACLE DOCS in Figure 1) but need to find a reference to q in these documents for other filters and downstream stages (ALIAS MATCH in Figure 1). An exact match to the query name is trivial, but some documents may not contain the query verbatim. This primarily occurs in cases where an alias is used, e.g. where the query Fyffes PLC is only mentioned as Fyffes in a document.

SF systems typically implement a query expansion step prior to searching for relevant documents, generating and extracting aliases based on the corpus and external sources (Ji et al., 2011). For documents that do not mention the query verbatim, we manually annotate the longest token span which refers to the query. All of our filters are applied to this base setup. To measure the effect of our manual aliases on recall, we implement a naïve EXACT MATCH filter, which allows a document only if a NE matches the query verbatim.

Entity form filters are based on the form of the entities extracted from documents. We initially consider all substrings of all NPs for a high-recall, yet tractable, baseline. The NP N-GRAMS filter allows every n-gram of every NP. NES allows NEs only; and for TYPES, fill NEs must be of a NER type defined by the slot, e.g. for `per:city of birth` only LOC NEs are allowed.

Sentence filters require the query mention and fill to be in the same sentence, or to have mentions in the same sentence. Sentence filters are COREF: the query and the fill must be mentioned in the same sentence; COREF NNP: as for COREF, but the query and the fill must have coreferent proper noun mentions in the same sentence; NAÏVE NNP: as for COREF NNP, but instead of using a full coreference system and identifying proper noun mentions, we use a naïve proper noun coreference process; and NOCOREF: the verbatim query and the fill must be named in the same sentence.

As dependency paths are often a key feature for extracting relations, we apply further syntactic filters based on dependency paths between NEs and mentions in sentences. Where we use dependencies, we use the Stanford collapsed and propagated representation (de Marneffe and Manning, 2008), e.g. in Alice is an employee of Bob and Charlie the collapsed and propagated dependency path between Alice and Charlie is $\rightarrow$`nsubj`$\rightarrow$employee$\leftarrow$`prep_of`$\leftarrow$.

Syntactic filters roughly capture the complexity of the syntactic configuration between query and filler: LENGTH $\leq$ N requires that the query and fill are separated by a dependency path of at most N arcs, e.g. the above dependency path is two arcs; VERB requires a verb to be present in the dependency path between the query and fill mentions or names; and NON-UNIQUE requires the dependency path between the query and fill to occur more than once in a corpus, modelling a hard constraint on bootstrapping and other learning processes that require a shared dependency context between training and test examples.

4.2 Bootstrapping reachability

In addition to the upper bound set by these explicit hard constraints, we want to reflect constraints that are implicitly applied by an extraction process—are there fills that are never learnable given a set of features and a set of training data? We extend our evaluation to include a training process in a semi-supervised setting. We treat this as a bootstrapping task (Agichtein and Gravano, 2000): given training pairs of NEs in text (each pair effectively a query entity and a candidate slot fill, or vice-versa), extract the context of each pair, and find other pairs in the corpus that share that context. A pair is reachable, and hence learnable, if it can be found by iterating this process. We continue to evaluate maximum recall and do not apply thresholding or ranking that would typically be utilised in a bootstrapping process. We simply output all

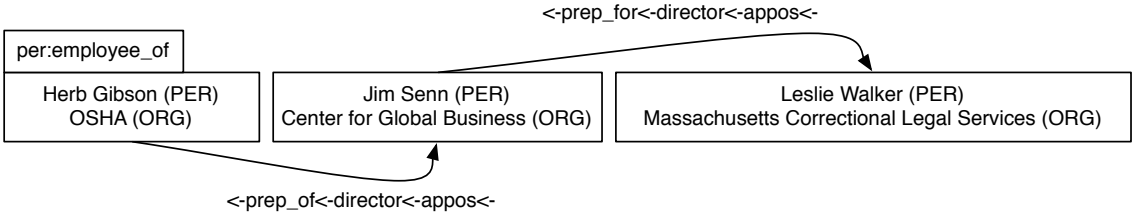


Figure 2: Bootstrapping. The rightmost vertex is labelled with `per:employee of` after two iterations.

possible candidates in order to measure recall loss: as with hard constraints applied by filters, if recall is lost it can never be recovered.

Given a set of training data, we identify if we can reach a test instance by bootstrapping, no matter how remotely it is connected to training instances. We use lemmatised dependency paths as the context for this process as they are relatively precise and discriminative, compared to other features used for SF. In order to simplify processing, we construct a graph of all pairs and paths in the corpus first, and then bootstrap from training instances over this graph. Bootstrapping more general features (e.g. bag-of-words) results in the graph becoming too large to process on our computing resources.

The graph is constructed as follows. Each vertex represents a typed pair of NEs that occur in the same sentence in the TAC KBP Source Data (LDC, 2010), collapsing vertices that have equal names and types into a single vertex. An edge exists between pairs that are connected at least once by the same dependency path. The constructed graph is equivalent to the EXACT MATCH + NOCOREF + NON-UNIQUE filter. Constructing a graph for COREF (which requires many more edges than NOCOREF) was impractical.

Initially, pairs in training data are labelled with their corresponding slots (see Figure 2). In each bootstrap iteration, the labels of each vertex are added to its neighbouring vertices. There is no filtering or competition between labels on a vertex, they are all added. We analyse performance after each iteration, evaluating by mapping the labelled graph back to the equivalent SF queries. This enables us to determine what fills are recoverable from the bootstrapping process.

5 Evaluation

We evaluate our filters on the TAC KBP English Slot Filling 2011 corpus, queries and task specification. As we aim to determine recall upper bounds and recall loss, we use only the documents D from the TAC KBP Source Data (LDC, 2010) that are known to contain at least one correct slot fill in the TAC KBP 2011 English Slot Filling Assessment Results (LDC, 2011).

We restrict the assessment results and the evaluation process to all slot types that are filled by `name` content types as opposed to `value` or `string`. We also do not evaluate the `per:alternate names` or `org:alternate names` slots, as extraction of fills for these slots typically falls outside the RE task: while X also known as Y or similar may appear in text, X and Y are typically mentioned independently across documents.

There are 100 TAC11 queries, 50 PER and 50 ORG. There are 535 fills in our reduced evaluation, 1,171 correct responses over these fills: 56% of the original evaluation slots. The distribution of fills per slot is listed in Table 1. The number of fills per query ranges from 0 (one query has no `name` fills) to 71, with a median of 17. D is comprised of 1,351 documents. The number of documents per query ranges from 0 to 63, with a median of 15.5. We use TAC 2009 and 2010 results and annotations as training data for bootstrapping, with 4,647 relevant training examples.

We evaluate ignoring case and without requiring a specific source document: `nocase` and `anydoc` in SF evaluation. Note that each slot fill is an equivalence class of responses: e.g. for `org:founded by` the correct fills Clifford S. Asness and Clifford Asness are equivalent. Consistent with SF evaluation, we identify at what constraint an entire equivalence class no longer has any member proposed as a fill.

We process documents with Stanford CoreNLP: tokenisation, POS tagging (Toutanova et al., 2003), NER (Finkel et al., 2005), parsing (Klein and Manning, 2003), and coreference resolution (Lee et al., 2011), and these annotations form the relevant components of our filters. Where we use dependency paths, we lemmatise tokens on the

slot	#	slot	#	slot	#
org:top members,employees	118	per:cities of residence	17	per:other family	6
per:employee of	71	per:children	17	per:city of birth	6
per:member of	47	org:stateorprovince of headquarters	17	per:parents	3
org:subsidiaries	32	per:schools attended	16	per:country of birth	3
org:parents	24	per:stateorprovinces of residence	11	org:political,religious affiliation	2
per:origin	23	org:member of	11	per:stateorprovince of birth	1
org:country of headquarters	22	per:spouse	8	per:country of death	1
per:countries of residence	20	org:members	8	per:city of death	1
org:city of headquarters	19	org:founded by	7		
org:shareholders	18	per:siblings	6		

Table 1: Number of fills for slots in the evaluation.

path to increase generality and recall in further analysis. For example, for Alice employs Bob we extract the path ←nsubj←employ→dobj→.

The COREF NNP filter uses CoreNLP coreference, limited to mentions which are headed by NNPs. For NAÏVE NNP we use a naïve rule-based coreference process (Pink et al., 2013), motivated by efficiency reasons, as the full CoreNLP requires parsing and a more complex model. The rules do not require deep processing and can run quickly over large volumes of text. All NEs from a document are matched by processing in decreasing length order. Two names are marked coreferent where, ignoring titles and case: they match exactly; they have a matching final word; they have a matching initial word; or one is an acronym of the other. If multiple conditions are matched, the earliest (the most strict match) is used.

The NON-UNIQUE filter requires that a dependency path occurs more than once between NEs in the full TAC KBP Source Data (LDC, 2010), comprised of 1.8M documents and 318M NE pairs. There are 38.6M distinct lemmatised dependency paths, 5M of which occur more than once.

6 Results

We now analyse where the filters lose recall. Results for non-syntactic filters are listed in Table 2. Figure 3 illustrates our main pipeline which contains filters that would typically be implemented.

NP n-grams We choose all n-grams of NPs (from the CoreNLP constituency parser) to be our highest recall filter, and so our highest baseline has 3% recall loss. We identify the reasons for loss at this filter. There are four errors due to the fill not existing verbatim in text, e.g. Pinellas and Pasco counties does not contain Pinellas County verbatim. Four errors occur where an NP is not correctly identified, which occurs in two different cases: where there is genuine error or where the sentence being parsed is actually a list or other semi-structured data as opposed to an actual sentence. four errors are where a correct answer has not been annotated as correct, we refer to this as ANNOTATION error below, and one case where an incorrect response has been annotated as correct.

While 97% recall is an excellent starting point, 53M candidates is a huge, likely intractable search space for any downstream process. Hence NER is commonly used as the starting point for SF.

NEs Most errors here are due to NER errors, and these errors result in nearly a 10% recall loss. 25 errors are caused where no token in the fill has been tagged as part of a NE (NO NER); and 13 where some tokens were missed (NER BOUNDS). There are two additional cases of ANNOTATION due to determiners not being included in an NE, where they perhaps should have also been annotated. Hence, in agreement with previous analyses, NER error has a large impact on SF.

On this data set we have 10% recall loss that most SF or RE approaches would never be able to extract. However, it is still fairly unconstrained and a high recall bound in comparison to the following filters. Recall errors could be substantially reduced if SF approaches were to take into consideration all NEs in documents as a set of candidates, and take a more document-based approach to RE as opposed to sentence-based. While there has been some work in extracting relations across sentences without coreference (Swampillai and Stevenson, 2011), RE across sentence boundaries is effectively limited to coreference chains between sentences. Currently whole document extraction is not a research focus for SF, and the implementation of whole document techniques throughout SF pipelines would likely be beneficial.

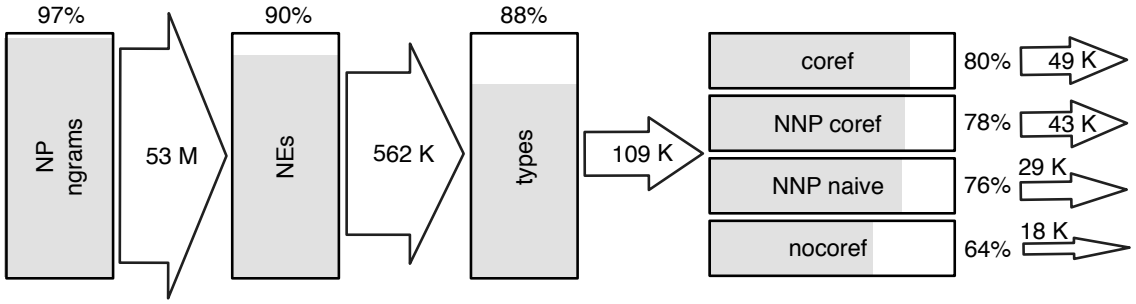


Figure 3: Results for NP N-GRAMS + NES + TYPES, followed by sentence filters with a range of coreference configurations. Grey fill and % indicates recall after each filter, and the number in the arrow is the size of the result set passed to the next filter or to the downstream process.

experiment	R (%)	|search space|
NP N-GRAMS	97	53966773
...+ NES	90	562318
...+ TYPES (1)	88	109241
...+ EXACT MATCH (2)	85	105764
(1) + COREF	80	49170
(1) + NNP COREF	78	43476
(1) + NNP NAÏVE	76	29171
(1) + NOCOREF	64	18331
(2) + COREF	77	47439
(2) + NNP COREF	73	30089
(2) + NNP NAÏVE	73	27770
(2) + NOCOREF	61	16978
(1) + COREF + NON-UNIQUE	65	19958
(1) + NNP COREF + NON-UNIQUE	62	17692
(1) + NNP NAÏVE + NON-UNIQUE	61	13960
(1) + NOCOREF + NON-UNIQUE	48	8084
(2) + COREF + NON-UNIQUE	63	18953
(2) + NNP COREF + NON-UNIQUE	60	16712
(2) + NNP NAÏVE + NON-UNIQUE	56	13064
(2) + NOCOREF + NON-UNIQUE	43	7236

Table 2: Results on D given sets of filters configurations. The ellipses indicate the previous line.

Exact match Requiring that the query name is exactly matched (EXACT MATCH) loses a further 2% recall. Effectively this is the recall error created by the IR component of SF. Five error cases occur when an alias is required, e.g. Quds Force for IRGC-QF; Chris Bentley for Christopher Bentley. Eight errors occur where the query term is a reference to an entity but not its name, all pertaining to the query GMAC's Residential Capital LLC.

Types All errors created by the TYPES filter are due to incorrect NER types on mentions proposed by CoreNLP. We do not aggregate the NE type over the coreference chain. Applying this filter cuts down the search space substantially, with minimal loss to recall. Adding TYPES results in a recall loss of 2%, but cuts down the search space by 80%.

Coref This filter is the starting point for many recent SF approaches: we consider entities that are either named or mentioned in the same sentence. Table 3 shows that coreference is the largest category of recall error created by the COREF filter. NN COREF, NNP COREF and PRP COREF indicate failure to resolve common noun, proper noun and pronoun coreference.

The remainder of the errors are cases where mentions of the fills do not occur in the same sentence. ROLE INF indicates that an individual's role is mentioned, e.g. Gene Roberts, the executive editor, where The Inquirer is mentioned in a previous sentence. LOC INF where additional location knowledge is required: a French company is headquartered in France. The search space has been substantially reduced, by a further 55% to 0.1% of the original space. However, the recall upper bound has dropped to 80% of all fills.

Coref NNP and naive NNP While coreference is important for high recall, more difficult coreference cases (common noun and pronoun coreference) may generate a large number of spurious cases. Using COREF NNP as the sentence filter loses 2% recall, to an upper bound of 78%, for a 12% reduction in the search space. However, using a full coreference system generates may more candidates than using simple NNP coreference. NAÏVE NNP has an upper bound of 76%. This is only 4% lower recall than COREF, but for a 41% reduction in search space. In addition, CoreNLP coreference is much more expensive than our naïve approach as it requires parsing.

No coref Errors for NOCOREF are listed in Table 3. INF indicates that inference or more sophisticated analysis is required to find the fill, such as correctly identifying the relation between entities

Experiment	NN COREF	NNP COREF	PRP COREF	ROLE INF	LOC INF	INF	NO NER	ANNOTATION
COREF	9	6	13	4	3	0	8	1
NOCOREF	16	52	20	4	3	2	14	3

Table 3: Error types for COREF and NOCOREF.

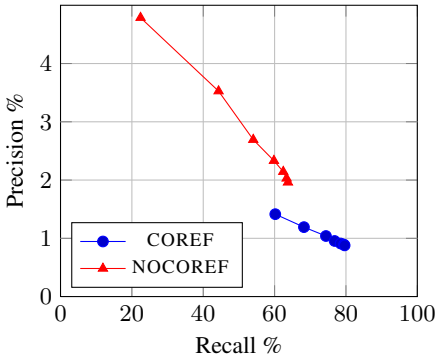


Figure 4: Effect of COREF.

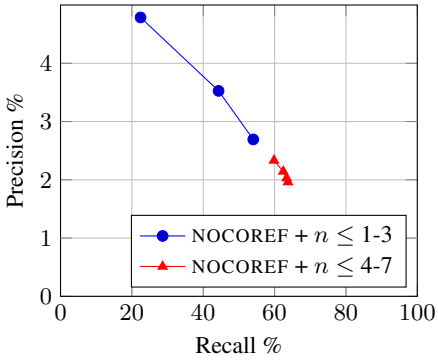


Figure 5: Effect of short dependency paths, taking the NOCOREF points from Figure 4.

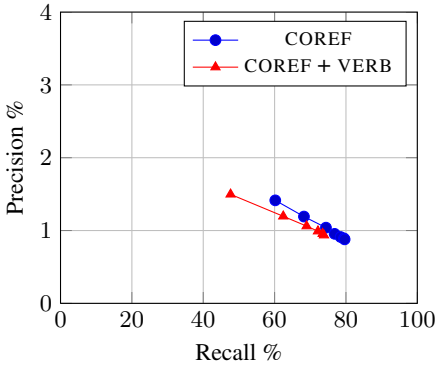


Figure 6: Effect of the VERB filter.

referred to in an interview. NOCOREF results in a recall upper bound of 64%. While this gives us a small search space, we are now losing a substantial proportion of the correct fills.

Precision-recall curves for the dependency path filters are given in Figures 4, 5 and 6. We choose to report precision for simplicity, and note that the downstream search space is the inverse of precision multiplied by the number of correct fills. Dots from low recall to high recall indicate maximum dependency path length from $n = 1$ to $n = 7$. Dependency paths of length 7 give maximum recall in our experiments. Results for the addition of the NON-UNIQUE constraint are given in Table 2.

Use of coreference While critical for recall, use of coreference generates a large number of candidates and presents a key trade-off for SF, as indicated by Figure 4. At maximum dependency path length, coreference gives 16% greater recall at a cost of 1.1% precision, roughly half the precision of no coreference.

Higher precision indicates that fewer candidates are generated. Fewer candidates allows for SF approaches to be scaled to larger amounts of data, and enables techniques that take advantage of redundancy or clustering to be used. Hence the higher precision no coreference approach may allow for more precise learning methods to be used, which may provide better results overall than an approach using coreference.

Short dependency paths In all of our filter configurations, a short dependency path length is sufficient for extracting the majority of slot fills for that particular configuration. Improving precision of fills found on short dependency paths may be a more effective and scalable approach to improving F-score rather than focusing on long paths.

In Figure 5 we consider NOCOREF. Limiting the dependency path length to three loses 11% recall, but gains 0.7% precision. While this loss of recall is high, the reduction in unique dependency paths is substantial. For maximum path length three there are 10,732 paths (1,551 unique); for all paths there are 17,394 paths (2,863 unique).

Verb Figure 6 shows the VERB filters has less impact or recall or precision than some other dependency filters. For COREF with all paths, adding the VERB filter loses 6% recall for a 0.1% gain in precision. Some slots not included in this analysis, such as `per:title`, tend to be described

by shorter paths that often do not include verbs. These slots are also frequent in the TAC11 dataset.

Non-unique The frequency of a dependency path may be a critical feature for learning, as paths that occur only once will not been seen by a bootstrapping process or may not be considered by other machine learning approaches. Applying the NON-UNIQUE filter (Table 2) has a large effect on recall: COREF loses 15% recall for a 41% reduction in the size of the search space; NOCOREF loses 15% recall for a 44% reduction in search space. To recover this recall, the strictness of this filter could be relaxed by further generalising dependency paths or using a different similarity metric to direct match of paths. However, this is the upper bound for approaches which consider only exact dependency paths as a feature.

Bootstrapping A small amount of training data quickly finds slot fills via bootstrapping. One iteration has a recall of 24%, with 7,665 candidates generated. Two to four iterations have recall of 37%–39% (maximum recall), with 31,702–37,797 candidates. The recall upper bound for these configurations is 43%—more training data will allow for better precision, but will only minimally improve recall in this setup. We note that limiting bootstrap to one or two iterations is ideal for the best trade-off between recall and search space. However, closer analysis of discriminative paths is required for a full SF system.

Note that even when bootstrapping through every dependency path in the corpus, there is an upper bound on recall of 39%. Even if we used the test data as additional training data the recall would still be limited to 43%. This demonstrates that systems need distributional features, dependency tree kernels or other similarity comparison as opposed to exact feature matching if dependency paths are to be a useful feature for SF.

7 Discussion

We present an analysis of SF recall bounds given hard constraints applied by standard system components. Pipeline error is common across all NLP tasks. Our analysis suggests that high-precision naïve tools, e.g. naïve coreference, can lead to state-of-the-art performance.

However, the SF task is not strictly an exhaustive evaluation for each query, as the evaluation data is comprised of the time-limited human annotation plus aggregated system output only. There may be fills that are missed in the evaluation results but are correct and returned by our high recall filters—affecting our reported precisions.

We manually evaluate a small sample of the queries, the first five person and the first five organization queries, to identify missed fills in the COREF output (2,903 of 49,170 total fills, or 5.9%). For these fills, there were 29 fills in the assessment data. Of these fills, 21 are returned by COREF, however there are two correct fills found by COREF that are not in the assessment data. One of these two errors would be identified with correct coreference, and the other requires complex long range inference. These additional correct fills that are identified will not have a large impact on the absolute precision, as there are two of 2,903 more fills. However, the relative difference in true positives, 21 to 23, results in some uncertainty in results when comparing them relatively.

8 Conclusion

Recent TAC KBP Slot Filling results have shown that state-of-the-art systems are substantially limited by low recall. In this work, we perform a maximum recall analysis of slot filling, providing a comprehensive analysis of recall error created in the document retrieval and candidate generation stages. We focus on recall error in candidate generation as a performance limitation, as candidates that are lost in the pipeline cannot be recovered by downstream processes.

We find ∼10% of recall is ignored by most slot filling systems due to NER error, and while state-of-the-art coreference provides a substantial recall gain over no coreference, 8% of recall is still lost when queries and fills occur in different sentences. Using NE type constraints is very effective, reducing recall by only 2% for a search space reduction of 81%. Without coreference, a further 16% of fills are lost, but 12% of this recall can be regained using efficient naïve name matching rules, while still reducing the search space by 41%, making such an approach possibly preferable over full coreference. We confirm that coreference and accurate NER are critical to high recall slot filling.

We find that using maximum recall bootstrapping, 39% of test slots fills are reachable from the TAC09 and TAC10 training data, limited by an upper bound on non-unique paths of 43%.

In the future, we intend to assess how specific

slots are affected by recall and search space trade-off, and perform evaluation over all slot types: names, values and strings. In addition, we intend to expand the bootstrapping experiments with variations over the training data.

This work highlights NER, coreference and typing as the areas that have the most impact on slot filling recall, enabling researchers to focus on problems that will most improve performance.

Acknowledgements

We would like to thank the anonymous reviewers for their useful feedback. This work was supported by an Australian Postgraduate Award, the Capital Markets CRC Computable News project and Australian Research Council Discovery grant DP1097291.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting Relations from Large Plain-text Collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 85–94, San Antonio, Texas, USA.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open Information Extraction from the Web. In *Proceedings of IJCAI*, pages 2670–2676, Hyderabad, India.

Sergey Brin. 1998. Extracting patterns and relations from the World Wide Web. In *Proceedings of the 1998 International Workshop on the Web and Databases*, Valencia, Spain.

Lorna Byrne and John Dunnion. 2011. UCD IIRG at TAC 2011. In *Proceedings of TAC*, Gaithersburg, Maryland, USA.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of AAAI*, pages 1306–1313, Atlanta, Georgia, USA.

Linguistic Data Consortium. 2010. TAC KBP Source Data. LDC2010E12.

Linguistic Data Consortium. 2011. TAC KBP 2011 English Slot Filling Assessment Results. LDC2011E88.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford Typed Dependencies Representation. In *Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, Manchester, United Kingdom.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying Relations for Open Information Extraction. In *Proceedings of EMNLP*, pages 1535–1545, Edinburgh, United Kingdom.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of ACL*, pages 363–370, Ann Arbor, Michigan, USA.

Ryan Gabbard, Marjorie Freedman, and Ralph Weischedel. 2011. Coreference for Learning to Extract Relations: Yes Virginia, Coreference Matters. In *Proceedings of ACL*, pages 288–293, Portland, Oregon, USA.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL-HLT*, pages 541–550, Portland, Oregon, USA.

Heng Ji and Ralph Grishman. 2011. Knowledge Base Population: Successful Approaches and Challenges. In *Proceedings of ACL-HLT*, pages 1148–1158, Portland, Oregon.

Heng Ji, Ralph Grishman, and Hoa Dang. 2011. Overview of the TAC2011 Knowledge Base Population Track. In *Proceedings of TAC*, Gaithersburg, Maryland, USA.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of ACL*, pages 423–430, Sapporo, Japan.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of CONLL: Shared Task*, pages 28–34, Portland, Oregan, USA.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of EMNLP-CONLL*, pages 523–534, Jeju Island, Korea.

Paul McNamee and Hoa Dang. 2009. Overview of the TAC 2009 Knowledge Base Population Track. In *Proceedings of TAC*, Gaithersburg, Maryland, USA.

Bonan Min and Ralph Grishman. 2012. Challenges in the Knowledge Base Population Slot Filling Task. In *Proceedings of LREC*, pages 1148–1158, Istanbul, Turkey.

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

Glen Pink, Will Radford, Will Cannings, Andrew Naoum, Joel Nothman, Daniel Tse, and James R. Curran. 2013. SYDNEY_CMCRC at TAC 2013. In *Proceedings of TAC*, Gaithersburg, Maryland, USA.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. 2013. Relation Extraction with Matrix Factorization and Universal Schemas. In *Proceedings of HLT-NAACL*, Atlanta, Georgia, USA.

Benjamin Roth, Tassilo Barth, Michael Wiegand, Mittul Singh, and Dietrich Klakow. 2013. Effective Slot Filling Based on Shallow Distant Supervision Methods. In *Proceedings of TAC*, Gaithersburg, Maryland, USA.

Benjamin Roth, Tassilo Barth, Grzegorz Chrupała, Martin Gropp, and Dietrich Klakow. 2014. RelationFactory: A Fast, Modular and Effective System for Knowledge Base Population. *Proceedings of EACL*, pages 89–92.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP-CONLL*, pages 455–465, Jeju Island, Korea.

Mihai Surdeanu. 2013. Overview of the TAC2013 Knowledge Base Population Evaluation: English Slot Filling and Temporal Slot Filling. In *Proceedings of TAC*, Gaithersburg, Maryland, USA.

Kumutha Swampillai and Mark Stevenson. 2011. Extracting Relations Within and Across Sentences. In *Proceedings of RANLP*, pages 25–32, Hissar, Bulgaria.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proceedings of HLT-NAACL*, pages 173–180, Edmonton, Canada.

Yafang Wang, Bin Yang, Lizhen Qu, Marc Spaniol, and Gerhard Weikum. 2011. Harvesting Facts from Textual Web Sources by Constrained Label Propagation. In *Proceedings of CIKM*, pages 837–846, Glasgow, Scotland, UK.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised Relation Discovery with Sense Disambiguation. In *Proceedings of ACL*, pages 712–720, Jeju Island, Korea.

Xingxing Zhang, Jianwen Zhang, Junyu Zeng, Jun Yan, Zheng Chen, and Zhifang Sui. 2013. Towards Accurate Distant Supervision for Relational Facts Extraction. In *Proceedings of ACL-HLT*, pages 810–815, Jeju Island, Korea.

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring Various Knowledge in Relation Extraction. In *Proceedings of ACL*, pages 427–434, Ann Arbor, Michigan, USA.

Relieving the Computational Bottleneck: Joint Inference for Event Extraction with High-Dimensional Features

Deepak Venugopal and **Chen Chen** and **Vibhav Gogate** and **Vincent Ng**
Department of Computer Science and Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
dxv021000@utdallas.edu, {yzcchen,vgogate,vince}@hlt.utdallas.edu

Abstract

Several state-of-the-art event extraction systems employ models based on Support Vector Machines (SVMs) in a pipeline architecture, which fails to exploit the joint dependencies that typically exist among events and arguments. While there have been attempts to overcome this limitation using Markov Logic Networks (MLNs), it remains challenging to perform joint inference in MLNs when the model encodes many high-dimensional sophisticated features such as those essential for event extraction. In this paper, we propose a new model for event extraction that combines the power of MLNs and SVMs, dwarfing their limitations. The key idea is to reliably learn and process high-dimensional features using SVMs; encode the output of SVMs as low-dimensional, soft formulas in MLNs; and use the superior joint inferencing power of MLNs to enforce joint consistency constraints over the soft formulas. We evaluate our approach for the task of extracting biomedical events on the BioNLP 2013, 2011 and 2009 Genia shared task datasets. Our approach yields the best F1 score to date on the BioNLP'13 (53.61) and BioNLP'11 (58.07) datasets and the second-best F1 score to date on the BioNLP'09 dataset (58.16).

1 Introduction

Event extraction is the task of extracting and labeling all instances in a text document that correspond to a pre-defined event type. This task is quite challenging for a multitude of reasons: events are often nested, recursive and have several arguments; there is no clear distinction between arguments and events; etc. For instance, consider the BioNLP Genia event extraction shared task (Nédellec et al., 2013). In this task, participants are asked to extract instances of a pre-defined set of biomedical events from text. An event is identified by a keyword called the *trigger* and can have an arbitrary number of arguments that correspond to pre-defined argument types. The task is complicated by the fact that an event may serve as an argument of another event (nested events). An example of the task is shown in Figure 1. As we can see, event $E13$ takes as arguments two events, $E14$ and $E12$, which in turn has $E11$ as one of its arguments.

A standard method that has been frequently employed to perform this shared task uses a *pipeline* architecture with three steps: (1) detect if a token is a trigger and assign a trigger type label to it; (2) for every detected trigger, determine all its arguments and assign types to each detected argument; and (3) combine the extracted triggers and arguments to obtain events. Though adopted by the top-performing systems such as the highest scoring system on the BioNLP'13 Genia shared task (Kim et al., 2013), this approach is problematic for at least two reasons. First, as is typical in pipeline architectures, errors may propagate from one stage to the next. Second, since each event/argument is identified and assigned a type independently of the others, it fails to capture the relationship between a trigger and its neighboring triggers, an argument and its neighboring arguments, etc.

More recently, researchers have investigated joint inference techniques for event extraction using Markov Logic Networks (MLNs) (e.g., Poon and Domingos (2007), Poon and Vanderwende (2010), Riedel and McCallum (2011a)), a statistical relational model that enables us to model the dependencies between different instances of a data sample. However, it is extremely challenging to make joint inference using MLNs work well in practice (Poon and Domingos, 2007). One reason is that it is generally difficult to model sophisticated linguistic features using MLNs. The diffi-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 831–843,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

. . . demonstrated that HOIL-1L interacting protein (HOIP), a ubiquitin ligase that can catalyze the assembly of linear polyubiquitin chains, is recruited to DC40 in a TRAF2-dependent manner following engagement of CD40 . . .

(a) Sentence fragment

ID	Event Type	Trigger	Arguments
$E11$	*Binding*	recruited	*Theme*={HOIL-1L interacting protein,CD40}
$E12$	*Regulation*	dependent	*Theme*=$E11$, *Cause*=TRAF2
$E13$	*+ve Regulation*	following	*Theme*=$E12$, *Cause*=$E14$
$E14$	*Binding*	engagement	*Theme*=CD40

(b) Events

Figure 1: Example of event extraction in the BioNLP Genia task. The table in (b) shows all the events extracted from sentence (a). Note that successful extraction of $E13$ depends on $E12$ and $E14$.

culty stems from the fact that some of these features are extremely high dimensional (e.g., Chen and Ng (2012), Huang and Riloff (2012b), Li et al. (2012), Li et al. (2013b), Li et al. (2013c)), and to reliably learn weights of formulas that encode such features, one would require an enormous number of data samples. Moreover, even the complexity of approximate inference on such models is quite high, often prohibitively so. For example, a trigram can be encoded as an MLN formula, $\texttt{Word}(w_1, p-1) \wedge \texttt{Word}(w_2, p) \wedge \texttt{Word}(w_3, p+1) \Rightarrow \texttt{Type}(p, T)$. For any given position (p), this formula has W^3 groundings, where W is the number of possible words, making it too large for learning/inference. Therefore, current MLN-based systems tend to include a highly simplified model ignoring powerful linguistic features. This is problematic because such features are essential for event extraction.

Our contributions in this paper are two-fold. First, we propose a novel model for biomedical event extraction based on MLNs that addresses the aforementioned limitations by leveraging the power of Support Vector Machines (SVMs) (Vapnik, 1995; Joachims, 1999) to handle high-dimensional features. Specifically, we (1) learn SVM models using rich linguistic features for trigger and argument detection and type labeling; (2) design an MLN composed of *soft formulas* (each of which encodes a soft constraint whose associated weight indicates how important it is to satisfy the constraint) and *hard formulas* (constraints that always need to be satisfied, thus having a weight of ∞) to capture the relational dependencies between triggers and arguments; and (3) encode the SVM output as *prior knowledge* in the MLN in the form of soft formulas, whose weights are computed using the confidence values generated by the SVMs. This formulation naturally allows SVMs and MLNs to complement each other's strengths and weaknesses: learning in a large and sparse feature space is much easier with SVMs than with MLNs, whereas modeling relational dependencies is much easier with MLNs than with SVMs.

Our second contribution concerns making inference with this MLN feasible. Recall that inference involves detecting and assigning the type label to all the triggers and arguments. We show that existing Maximum-a-posteriori (MAP) inference methods, even the most advanced approximate ones (e.g., Selman et al. (1996), Marinescu and Dechter (2009), Sontag and Globerson (2011)), are infeasible on our proposed MLN because of their high memory cost. Consequently, we identify decompositions of the MLN into disconnected components and solve each independently, thereby drastically reducing the memory requirements.

We evaluate our approach on the BioNLP 2009, 2011 and 2013 Genia shared task datasets. On the BioNLP'13 dataset, our model significantly outperforms state-of-the-art pipeline approaches and achieves the best F1 score to date. On the BioNLP'11 and BioNLP'09 datasets, our scores are slightly better and slightly worse respectively than the best reported results. However, they are significantly better than state-of-the-art MLN-based systems.

2 Background

2.1 Related Work

As a core task in information extraction, event extraction has received significant attention in the natural language processing (NLP) community. The development and evaluation of large-scale learning-based event extraction systems was propelled in part by the availability of annotated corpora produced as part of the Message Understanding Conferences (MUCs), the Automatic Content Extraction (ACE) evaluations, and the BioNLP shared

tasks on event extraction. Previous work on event extraction can be broadly divided into two categories, one focusing on the development of features (henceforth *feature-based* approaches) and the other focusing on the development of models (henceforth *model-based* approaches).

Feature-based approaches. Early work on feature-based approaches has primarily focused on designing local sentence-level features such as token and syntactic features (Grishman et al., 2005; Ahn, 2006). Later, it was realized that local features were insufficient to reliably and accurately perform event extraction in complex domains and therefore several researchers proposed using *high-level features*. For instance, Ji and Grishman (2008) used global information from related documents; Gupta and Ji (2009) extracted implicit time information; Patwardhan and Riloff (2009) used broader sentential context; Liao and Grishman (2010; 2011) leveraged document-level cross-event information and topic-based features; and Huang and Riloff (2012b) explored discourse properties.

Model-based approaches. The model-based approaches developed to date have focused on modeling global properties and seldom use rich, high-dimensional features. To capture global event structure properties, McClosky et al. (2011a) proposed a dependency parsing model. To extract event arguments, Li et al. (2013b) proposed an Integer Linear Programming (ILP) model to encode the relationship between event mentions. To overcome the error propagation problem associated with the pipeline architecture, several joint models have been proposed, including those that are based on MLNs (e.g., Poon and Domingos (2007), Riedel et al. (2009), Poon and Vanderwende (2010)), structured perceptrons (e.g., Li et al. (2013c)), and dual decomposition with minimal domain adaptation (e.g., Riedel and McCallum (2011a; 2011b)).

In light of the high annotation cost required by supervised learning-based event extraction systems, several semi-supervised, unsupervised, and rule-based systems have been proposed. For instance, Huang and Riloff (2012a) proposed a bootstrapping method to extract event arguments using only a small amount of annotated data; Lu and Roth (2012) developed a novel unsupervised sequence labeling model; Bui et al. (2013) implemented a rule-based approach to extract biomedical events; and Ritter et al. (2012) used unsupervised learning to extract events from Twitter data.

Our work extends prior work by developing a rich framework that leverages sophisticated feature-based approaches as well as joint inference using MLNs. This combination gives us the best of both worlds because on one hand, it is challenging to model sophisticated linguistic features using MLNs while on the other hand, feature-based approaches employing sophisticated high-dimensional features suffer from error propagation as the model is generally not rich enough for joint inference.

2.2 The Genia Event Extraction Task

The BioNLP Shared Task (BioNLP-ST) series (Kim et al. (2009), Kim et al. (2011a) and Nédellec et al. (2013)) is designed to tackle the problem of extracting structured information from the biomedical literature. The Genia Event Extraction task is arguably the most important of all the tasks proposed in BioNLP-ST and is also the only task organized in all three events in the series.

The 2009 edition of the Genia task (Kim et al., 2009) was conducted on the Genia event corpus (Kim et al., 2008), which only contains abstracts of the articles that represent domain knowledge around NFκB proteins. The 2011 edition (Kim et al., 2011b) augmented the dataset to include full text articles, resulting in two collections, the abstract collection and the full text collection. The 2013 edition (Kim et al., 2013) further augmented the dataset with recent full text articles but removed the abstract collection entirely.

The targeted event types have also changed slightly over the years. Both the 2009 and 2011 editions are concerned with nine fine-grained event sub-types that can be categorized into three main types, namely simple, binding and regulation events. These three main event types can be distinguished by the kinds of arguments they take. A simple event can take exactly one protein as its $Theme$ argument. A binding event can take one or more proteins as its $Theme$ arguments, and is therefore slightly more difficult to extract than a simple event. A regulation event takes exactly one protein or event as its $Theme$ argument and optionally one protein or event as its $Cause$ argument. If a regulation event takes another event as its $Theme$ or $Cause$ argument, it will lead to a nested event. Regulation events are considered the most difficult-to-extract among the three event types owing in part to the presence of an optional $Cause$ argument and their recursive structure. The 2013 edition intro-

duced a new event type, protein-mod, and its three sub-types. Theoretically, a protein-mod event takes exactly one protein as its $Theme$ argument and optionally one protein or event as its $Cause$ argument. In practice, however, it rarely occurs: there are only six protein-mod events having $Cause$ arguments in the training data for the 2013 edition. Consequently, our model makes the simplifying assumption that a protein-mod event can only take one $Theme$ argument, meaning that we are effectively processing protein-mod events in the same way as simple events.

2.3 Markov Logic Networks

Statistical relational learning (SRL) (Getoor and Taskar, 2007) is an emerging field that seeks to unify logic and probability, and since most NLP techniques are grounded either in logic or probability or both, NLP serves as an ideal application domain for SRL. In this paper, we will employ a popular SRL approach called Markov logic networks (MLNs) (Domingos and Lowd, 2009). At a high level, an MLN is a set of weighted first-order logic formulas (f_i, w_i), where w_i is the weight associated with formula f_i. Given a set of constants that model objects in the domain, it defines a Markov network or a log-linear model (Koller and Friedman, 2009) in which we have one node per ground first-order atom and a propositional feature corresponding to each grounding of each first-order formula. The weight of the feature is the weight of the corresponding first-order formula.

Formally, the probability of a world ω, which represents an assignment of values to all ground atoms in the Markov network, is given by:

$$\Pr(\omega) = \frac{1}{Z} \exp\left(\sum_i w_i N(f_i, \omega)\right)$$

where $N(f_i, \omega)$ is the number of groundings of f_i that evaluate to True in ω and Z is a normalization constant called the partition function.

The key inference tasks over MLNs are computing the partition function (Z) and the most-probable explanation given evidence (the MAP task). Most queries, including those required by event extraction, can be reduced to these inference tasks. Formally, the partition function and the MAP tasks are given by:

$$Z = \sum_\omega \exp\left(\sum_i w_i N(f_i, \omega)\right) \quad (1)$$

$$\arg\max_\omega P(\omega) = \arg\max_\omega \sum_i w_i N(f_i, \omega) \quad (2)$$

3 Pipeline Model

We implement a pipeline event extraction system using SVMs. This pipeline model serves two important functions: (1) providing a baseline for evaluation and (2) producing prior knowledge for the joint model.

Our pipeline model consists of two steps: trigger labeling and argument labeling. In the trigger labeling step, we determine whether a candidate trigger is a true trigger and label each true trigger with its trigger type. Then, in the argument labeling step, we identify the arguments for each true trigger discovered in the trigger labeling step and assign a role to each argument.

We recast each of the two steps as a classification task and employ $\mathbf{SVM}^{multiclass}$ (Tsochantaridis et al., 2004) to train the two classifiers. We describe each step in detail below.

3.1 Trigger Labeling

A preliminary study of the BioNLP'13 training data suggests that 98.7% of the true triggers' head words[1] are either verbs, nouns or adjectives. Therefore, we consider only those words whose part-of-speech tags belong to the above three categories as candidate triggers. To train the trigger classifier, we create one training instance for each candidate trigger in the training data. If the candidate trigger is not a trigger, the class label of the corresponding instance is $None$; otherwise, the label is the type of the trigger. Thus, the number of class labels equals the number of trigger types plus one. Each training instance is represented by the features described in Table 1(a). These features closely mirror those used in state-of-the-art trigger labeling systems such as Miwa et al. (2010b) and Björne and Salakoski (2013).

After training, we apply the resulting trigger classifier to classify the test instances, which are created in the same way as the training instances. If a test instance is predicted as $None$ by the classifier, the corresponding candidate trigger is labeled as a non-trigger; otherwise, the corresponding candidate trigger is posited as a true trigger whose type is the class value assigned by the classifier.

[1] Head words are found using Collins' (1999) rules.

(a) Features for trigger labeling

Token features	The basic token features (see Table 1(c)) computed from (1) the candidate trigger word and (2) the surrounding tokens in a window of two; character bigrams and trigrams of the candidate trigger word; word n-grams (n=1,2,3) of the candidate trigger word and its context words in a window of three; whether the candidate trigger word contains a digit; whether the candidate trigger word contains an upper case letter; whether the candidate trigger word contains a symbol.
Dependency features	The basic dependency path features (see Table 1(c)) computed using the shortest paths from the candidate trigger to (1) the nearest protein word, (2) the nearest protein word to its left, and (3) the nearest protein word to its right.
Other features	The distances from the candidate trigger word to (1) the nearest protein word, (2) the nearest protein word to its left, and (3) the nearest protein word to its right; the number of protein words in the sentence.

(b) Features for argument labeling

Token features	Word n-grams (n=1,2,3) of (1) the candidate trigger word and its context in a window of three and (2) the candidate argument word and its context in a window of three; the basic token features (see Table 1(c)) computed from (1) the candidate trigger word and (2) the candidate argument word; the trigger type of the candidate trigger word.
Dependency features	The basic dependency features (see Table 1(c)) computed using the shortest path from the candidate trigger word to the candidate argument word.
Other features	The distance between the candidate trigger word and the candidate argument word; the number of proteins between the candidate trigger word and the candidate argument word; the concatenation of the candidate trigger word and the candidate argument word; the concatenation of the candidate trigger type and the candidate argument word.

(c) Basic token and dependency features

Basic token features	Six features are computed given a token t, including: (a) the lexical string of t, (b) the lemma of t, (c) the stem of t obtained using the Porter stemmer (Porter, 1980), (d) the part-of-speech tag of t, (e) whether t appears as a true trigger in the training data, and (f) whether t is a protein name.
Basic dependency features	Six features are computed given a dependency path p, including: (a) the vertex walk in p, (b) the edge walk in p, (c) the n-grams (n=2,3,4) of the (stemmed) words associated with the vertices in p, (d) the n-grams (n=2,3,4) of the part-of-speech tags of the words associated with the vertices in p, (e) the n-grams (n=2,3,4) of the dependency types associated with the edges in p, and (f) the length of p.

Table 1: Features for trigger labeling and argument labeling.

3.2 Argument Labeling

The argument classifier is trained as follows. Each training instance corresponds to a candidate trigger and one of its candidate arguments.[2] A candidate argument for a candidate trigger ct is either a protein or a candidate trigger that appears in the same sentence as ct. If ct is not a true trigger, the label of the associated instance is set to $None$. On the other hand, if ct is a true trigger, we check whether the candidate argument in the associated instance is indeed one of ct's arguments. If so, the class label of the instance is the argument's role; otherwise, the class label is $None$. The features used for representing each training instance, which are modeled after those used in Miwa et al. (2010b) and Björne and Salakoski (2013), are shown in Table 1(b).

After training, we can apply the resulting classifier to classify the test instances, which are created in the same way as the training instances. If a test instance is assigned the class $None$ by the classifier, the corresponding candidate argument is classified as not an argument of the trigger. Otherwise, the candidate argument is a true argument of the trigger whose role is the class value assigned by the classifier.

[2]Following the definition of the GENIA event extraction task, the protein names are provided as part of the input.

4 Joint Model

In this section, we describe our Markov logic model that encodes the relational dependencies in the shared task and uses the output of the pipeline model as prior knowledge (soft evidence). We begin by describing the structure of our Markov logic model, and then describe the parameter learning and inference algorithms for it.

4.1 MLN Structure

Figure 2 shows our proposed MLN for BioNLP event extraction, which we refer to as `BioMLN`. The MLN contains six predicates.

The *query predicates* in Figure 2(a) are those whose assignments are not given during inference and thus need to be predicted. Predicate `TriggerType`(sid,tid,$ttype$!) is true when the token located in sentence sid at position tid has type $ttype$. Δ_{ttype}, which denotes the set of constants (or objects) that the logical variable $ttype$

TriggerType(sid,tid,$ttype$!) ArgumentRole(sid,aid,tid,$arole$!) (a) Query	Simple(sid,tid) Regulation(sid,tid) (b) Hidden	Word(sid,tid,$word$) DepType(sid,aid,tid,$dtype$) (c) Evidence

1. $\exists t$ TriggerType(i,j,t).
2. $\exists a$ ArgumentRole(i,k,j,a).
3. $\neg$TriggerType(i,j,$None$) $\Rightarrow \exists k$ ArgumentRole(i,k,j,$Theme$).
4. Simple(i,j) $\Rightarrow \neg \exists k$ ArgumentRole(i,k,j,$Cause$).
5. TriggerType(i,j,$None$) $\Leftrightarrow$ ArgumentRole(i,k,j,$None$).
6. $\neg$ArgumentRole(i,k,j,$None$) $\wedge \neg$TriggerType(i,k,$None$) $\Rightarrow$ Regulation(i,j).
7. Simple(i,j) $\Leftrightarrow$ TriggerType(i,j,$Simple1$) $\vee \ldots \vee$ TriggerType(i,j,$Binding$).
8. Regulation(i,j) $\Leftrightarrow$ TriggerType(i,j,Reg) $\vee$ TriggerType(i,j,$PosReg$) $\vee$ TriggerType(i,j,$NegReg$).
9. Word(i,j,+w) $\wedge$ TriggerType(i,j,+t) $\wedge$ DepType(i,k,j,+d) $\wedge$ ArgumentRole(i,k,j,+a)

(d) Joint Formulas

Figure 2: The BioMLN structure.

can be instantiated to, includes all possible trigger types in the dataset plus $None$ (which indicates that the token is not a trigger). The "!" symbol models commonsense knowledge that only one of the types in the domain Δ_{ttype} of $ttype$ is true for every unique combination of sid and tid. Similarly, predicate ArgumentRole(sid,aid,tid,$arole$!) asserts that a token in sentence sid at position aid plays *exactly one* argument role, denoted by $arole$, with respect to the token at position tid. Δ_{arole} includes the two argument types, namely, $Theme$ and $Cause$ plus the additional $None$ that indicates that the token is not an argument.

The hidden predicates in Figure 2(b) are "clusters" of trigger types. Predicate Simple(sid,tid) is true when the token in sentence sid at position tid corresponds to one of the *Simple* event trigger types (BioNLP'13 has 9 simple events, BioNLP'09/'11 have 5) or a binding event trigger type. Similarly, Regulation(sid,tid) asserts that the token in sentence sid at position tid corresponds to any of the three regulation event trigger types.

The *evidence* predicates in Figure 2(c) are those that are always assumed to be known during inference. We define two evidence predicates based on dependency structures. Word(sid,tid,$word$) is true when the word in sentence sid at position tid is equal to $word$. DepType(sid,aid,tid,$dtype$) asserts that $dtype$ is the dependency type in the dependency parse tree that connects the token at position tid to the token at position aid in sentence sid. If the word at tid and the word at aid are directly connected in the dependency tree, then $dtype$ is the label of dependency edge with direction; otherwise $dtype$ is $None$.

The MLN formulas, expressing commonsense, prior knowledge in the domain (Poon and Vanderwende, 2010; Riedel and McCallum, 2011a), are shown in Fig. 2(d). All formulas, except Formula (9), are hard formulas, meaning that they have infinite weights. Note that during weight learning, we only learn the weights of soft formulas.

Formulas (1) and (2) along with the "!" constraint in the predicate definition ensure that the token types are mutually exclusive and exhaustive. Formula (3) asserts that every trigger should have an argument of type $Theme$, since a $Theme$ argument is mandatory for any event. Formula (4) models the constraint that a $Simple$ or $Binding$ trigger has no arguments of type $Cause$ since only regulation events have a $Cause$. Formula (5) asserts that non-triggers have no arguments and vice-versa. Formula (6) models the constraint that if a token is both an argument of t and a trigger by itself, then t must belong to one of the three regulation trigger types. This formula captures the recursive relationship between triggers. Formulas (7) and (8) connect the hidden predicates with the query predicates. Formula (9) is a soft formula encoding

the relationship between triggers and arguments in a dependency parse tree. It joins a word and the dependency type label that connects the word token to the argument token in the dependency parse tree with the trigger types and argument types of the two tokens. The "+" symbol indicates that each grounding of Formula (9) may have a different weight.

4.2 Weight Learning

We can learn BioMLN from data either discriminatively or generatively. Since discriminative learning is much faster than generative learning, we use the former. In discriminative training, we maximize the conditional log-likelihood (CLL) of the query and the hidden variables given an assignment to the evidence variables. In principle, we can use the standard gradient descent algorithm for maximizing the CLL. In each iteration of gradient descent, we update the weights using the following equation (cf. Singla and Domingos (2005) and Domingos and Lowd (2009)):

$$w_j^{t+1} = w_j^t - \alpha(\mathbb{E}_{\mathbf{w}}(n_j) - n_j) \qquad (3)$$

where w_j^t represents the weight of the j^{th} formula in the t^{th} iteration, n_j is the number of groundings in which the j^{th} formula is satisfied in the training data, $\mathbb{E}_{\mathbf{w}}(n_j)$ is the expected number of groundings in which the j^{th} formula is satisfied given the current weight vector $\mathbf{w}$, and α is the learning rate.

As such, the update rule given in Equation (3) is likely to yield poor accuracy because the number of training examples of some types (e.g., $None$) far outnumber other types. To rectify this ill-conditioning problem (Singla and Domingos, 2005; Lowd and Domingos, 2007), we divide the gradient with the number of true groundings in the data, namely, we compute the gradient using $\frac{(\mathbb{E}_{\mathbf{w}}(n_j)-n_j)}{n_j}$.

Another key issue with using Equation (3) is that computing $\mathbb{E}_{\mathbf{w}}(n_j)$ requires performing inference over the MLN. This step is intractable, #P-complete in the worst case. To circumvent this problem and for fast, scalable training, we instead propose to use the voted perceptron algorithm (Collins, 2002; Singla and Domingos, 2005). This algorithm approximates $\mathbb{E}_{\mathbf{w}}(n_j)$ by counting the number of satisfied groundings of each formula in the MAP assignment. Computing the MAP assignment is much easier (although still NP-hard in the worst case) than computing $\mathbb{E}_{\mathbf{w}}(n_j)$, and as a result the voted perceptron algorithm is more scalable than the standard gradient descent algorithm. In addition, it converges much faster.

4.3 Testing

In the testing phase, we combine BioMLN with the output of the pipeline model (see Section 3) to obtain a new MLN, which we refer to as BioMLN$^+$. For every candidate trigger, the SVM trigger classifier outputs a vector of signed confidence values (which is proportional to the distance from the separating hyperplane) of dimension Δ_{ttype} with one entry for each trigger type. Similarly, for every candidate argument, the SVM argument classifier outputs a vector of signed confidence values of dimension Δ_{arole} with one entry for each argument role. In BioMLN$^+$, we model the SVM output as soft evidence, using two soft unit clauses, TriggerType(i,+j,+t) and ArgumentRole(i,+k,+j,+a). We use the confidence values to determine the weights of these clauses. Intuitively, higher (smaller) the confidence, higher (smaller) the weight.

Specifically, the weights of the soft unit clauses are set as follows. If the SVM trigger classifier determines that the trigger in sentence i at position j belongs to type t with confidence $C_{i,j}$, then we attach a weight of $\frac{C_{i,j}}{\alpha n_i}$ to the clause TriggerType(i,j,t). Here, n_i denotes the number of trigger candidates in sentence i. Similarly, if the SVM argument classifier determines that the token at position k in sentence i belongs to the argument role a with respect to the token at position j, with confidence $C'_{i,k,j}$, then we attach a weight of $\frac{C'_{i,k,j}}{\beta \sum_{j=1}^{n_i} m_{ij}}$ to the clause ArgumentRole(i, k, j,a). Here, m_{ij} denotes the number of argument candidates for the j^{th} trigger candidate in sentence i. α and β act as *scale parameters* for the confidence values ensuring that the weights don't get too large (or too small).

4.4 Inference

As we need to perform MAP inference, both at training time and at test time, in this subsection we will describe how to do it efficiently by exploiting unique properties of our proposed BioMLN.

Naively, we can perform MAP inference by grounding BioMLN to a Markov network and then reducing the Markov network by removing from it all (grounded propositional) formulas that are inconsistent with the evidence. On the re-

duced Markov network, we can then compute the MAP solution using standard MAP solvers such as MaxWalkSAT (a state-of-the-art local search based MAP solver) (Selman et al., 1996) and Gurobi[3] (a state-of-the-art, parallelized ILP solver).

The problem with the above approach is that grounding the MLN is infeasible in practice; even the reduced Markov network is just too large. For example, assuming a total of $|\Delta_{sid}|$ sentences and a maximum of N tokens in a sentence, Formula (3) alone has $O(|\Delta_{sid}|N^3)$ groundings. Concretely, at training time, assuming 1000 sentences with 10 tokens per sentence, Formula (3) itself yields one million groundings. Clearly, this approach is not scalable. It turns out, however, that the (ground) Markov network can be decomposed into several disconnected components, each of which can be solved independently. This greatly reduces the memory requirement of the inference step. Specifically, for every grounding of sid, we get a set of nodes in the Markov network that are disconnected from the rest of the Markov network and therefore independent of the rest of the network. Formally,

Proposition 1. *For any world ω of the* `BioMLN`*,*

$$P_{\mathcal{M}}(\omega) = P_{\mathcal{M}_i}(\omega_i)P_{\mathcal{M}\setminus\mathcal{M}_i}(\omega \setminus \omega_i) \quad (4)$$

where ω_i is the world ω projected on the groundings of sentence i and $\mathcal{M}_i$ is `BioMLN` *grounded only using sentence i.*

Using Equation (4), it is easy to see that the MLN $\mathcal{M}$ can be decomposed into $|\Delta_{sid}|$ disjoint MLNs, $\{\mathcal{M}_k\}_{k=1}^{|\Delta_{sid}|}$. The MAP assignment to $\mathcal{M}$ can be computed using, $\bigcup_{i=1}^{|\Delta_{sid}|}\left(\arg\max_{\omega_i} P_{\mathcal{M}_i}(\omega_i)\right)$. This result ensures that to approximate the expected counts $\mathbb{E}_{\mathbf{w}}(n_j)$, it is sufficient to keep exactly *one sentence's groundings in memory*. Specifically, $\mathbb{E}_{\mathbf{w}}(n_j)$ can be written as $\sum_{k=1}^{|\Delta_{sid}|}\mathbb{E}_{\mathbf{w}}(n_j^k)$, where $\mathbb{E}_{\mathbf{w}}(n_j^k)$ indicates the expected number of satisfied groundings of the j^{th} formula in the k^{th} sentence. Since the MAP computation is decomposable, we can estimate $\mathbb{E}_{\mathbf{w}}(n_j^k)$ using MAP inference on just the k^{th} sentence.

5 Evaluation

5.1 Experimental Setup

We evaluate our system on the BioNLP'13 (Kim et al., 2013), '11 (Kim et al., 2011a) and '09 (Kim et al., 2009) Genia datasets for the main event extraction shared task. Note that this task is the most important one for Genia and therefore has the most active participation. Statistics on the datasets are shown in Table 2. All our evaluations use the online tool provided by the shared task organizers. We report scores obtained using the *approximate span, recursive* evaluation.

Dataset	#Papers	#Abstracts	#TT	#Events
BioNLP'13	(10,10,14)	(0,0,0)	13	(2817,3199,3348)
BioNLP'11	(5,5,4)	(800,150,260)	9	(10310,4690,5301)
BioNLP'09	(0,0,0)	(800,150,260)	9	(8597,1809,3182)

Table 2: Statistics on the BioNLP datasets, which consist of annotated papers/abstracts from PubMed. (x, y, z): x in training, y in development and z in test. #TT indicates the total number of trigger types. The total number of argument types is 2.

To generate features, we employ the supporting resources provided by the organizers. Specifically, sentence split and tokenization are done using the GENIA tools, while part-of-speech information is provided by the BLLIP parser that uses the self-trained biomedical model (McClosky, 2010). Also, we create dependency features from the parse trees provided by two dependency parsers, the Enju parser (Miyao and Tsujii, 2008) and the aforementioned BLLIP parser that uses the self-trained biomedical model, which results in two sets of dependency features.

For MAP inference, we use Gurobi, a parallelized ILP solver. After inference, a post-processing step is required to generate biomedical events from the extracted triggers and arguments. Specifically, for binding events, we employ a learning-based method similar to Björne and Salakoski (2011), while for the other events, we employ a rule-based approach similar to Björne et al. (2009). Both the SVM baseline system and the combined MLN+SVM system employ the same post-processing strategy.

During weight learning, in order to combat the problem of different initializations yielding radically different parameter estimates, we start at several different initialization points and average the weights obtained after 100 iterations of gradient descent. However, we noticed that if we simply choose random initialization points, the variance of the weights was quite high and some initialization points were much worse than others. To counter this, we use the following method to systematically

[3] http://www.gurobi.com/

System	Rec.	Prec.	F1
Our System	48.95	59.24	**53.61**
EVEX (Hakala et al., 2013)	45.44	58.03	50.97
TEES-2.1 (Björne and Salakoski, 2013)	46.17	56.32	50.74
BIOSEM (Bui et al., 2013)	42.47	62.83	50.68
NCBI (Liu et al., 2013)	40.53	61.72	48.93
DLUTNLP (Li et al., 2013a)	40.81	57.00	47.56

Table 3: Recall (Rec.), Precision (Prec.) and F1 score on the BioNLP'13 test data.

initialize the weights. Let n_i be the number of satisfied groundings of formula f_i in the training data and m_i be the total number of possible groundings of f_i. We use a threshold γ to determine whether we wish to make the initial weight positive or negative. If $\frac{n_i}{m_i} \leq \gamma$, then we choose the initial weight uniformly at random from the range $[-0.1, 0]$. Otherwise, we chose it from the range $[0, 0.1]$. These steps ensure that the weights generated from different initialization points have smaller variance. Also, in the testing phase, we set the scale parameters for the soft evidence as $\alpha = \beta = \max_{c \in \mathbf{C}} |c|$, where $\mathbf{C}$ is the set of SVM confidence values.

5.2 Results on the BioNLP'13 Dataset

Among the three datasets, the BioNLP'13 dataset is most "realistic" one because it is the only one that contains full papers and no abstracts. As a result, it is also the most challenging dataset among the three. Table 3 shows the results of our system along with the results of other top systems published in the official evaluation of BioNLP'13. Our system achieves the best F1-score (an improvement of 2.64 points over the top-performing system) and has a much higher recall (mainly because our system detects more regulation events which outnumber other event types in the dataset) and a slightly higher precision than the winning system. Of the top five teams, NCBI is the only other joint inference system, which adopts joint pattern matching to predict triggers and arguments at the same time. These results illustrate the challenge in using joint inference effectively. NCBI performed much worse than the SVM-based pipeline systems, EVEX and TEES2.1. It was also worse than BIOSEM, a rule-based system that uses considerable domain expertise. Nevertheless, it was better than DLUTNLP, another SVM-based system.

Figure 3 compares our baseline pipeline model with our combined model. We can clearly see that the combined model has a significantly better F1 score than the pipeline model on most event types.

System	Rec.	Prec.	F1
Our System	53.42	63.61	**58.07**
Miwa12 (Miwa et al., 2012)	53.35	63.48	57.98
Riedel11 (Riedel et al., 2011)	–	–	56
UTurku (Björne and Salakoski, 2011)	49.56	57.65	53.30
MSR-NLP (Quirk et al., 2011)	48.64	54.71	51.50

Table 4: Results on the BioNLP'11 test data.

The regulation events are considered the most complex events to detect because they have a recursive structure. At the same time, this structure yields a large number of joint dependencies. The advantage of using a rich model such as MLNs can be clearly seen in this case; the combined model yields a 10 point and 6 point increase in F1-score on the test data and development data respectively compared to the pipeline model.

5.3 Results on the BioNLP'11 Dataset

Table 4 shows the results on the BioNLP'11 dataset. We can see that our system is marginally better than Miwa12, which is a pipeline-based system. It is also more than two points better than Riedel11, a state-of-the-art structured prediction-based joint inference system. Reidel11 incorporates the Stanford predictions (McClosky et al., 2011b) as features in the model. On the two hardest, most complex tasks, detecting regulation events (which have recursive structures and more joint dependencies than other event types) and detecting binding events (which may have multiple arguments), our system performs better than both Miwa12 and Riedel11.[4] Specifically, our system's F1 score for regulation events is 46.84, while those of Miwa12 and Riedel11 are 45.46 and 44.94 respectively. Our system's F1 score for the binding event is 58.79, while those of Miwa12 and Riedel11 are 56.64 and 48.49 respectively. These results clearly demonstrate the effectiveness of enforcing joint dependencies along with high-dimensional features.

5.4 Results on the BioNLP'09 Dataset

Table 5 shows the results on the BioNLP'09 dataset. Our system has a marginally lower score (by 0.11 points) than Miwa12, which is the best performing system on this dataset. Specifically, our system achieves a higher recall but a lower precision than Miwa12. However, note that Miwa12 used co-reference features while we are able to achieve

[4]Detailed results are not shown for any of these three datasets due to space limitations.

	SVM			MLN+SVM		
Type	**Rec.**	**Prec.**	**F1**	**Rec.**	**Prec.**	**F1**
Simple	64.47	87.89	74.38	73.11	78.99	75.94
Protein-Mod	66.49	79.87	72.57	72.25	69.70	70.95
Binding	39.04	50.00	43.84	48.05	43.84	45.85
Regulation	23.51	56.21	33.15	36.47	50.86	42.48
Overall	37.90	67.88	**48.64**	48.95	59.24	**53.61**

(a) Test

	SVM			MLN+SVM		
Type	**Rec.**	**Prec.**	**F1**	**Rec.**	**Prec.**	**F1**
Simple	55.79	81.63	66.28	63.21	75.10	68.64
Protein-Mod	64.47	87.89	74.38	71.14	85.63	77.72
Binding	31.90	48.77	38.57	47.99	50.00	48.97
Regulation	20.13	52.46	29.10	28.57	43.41	34.46
Overall	34.42	66.14	**45.28**	43.50	57.45	**49.51**

(b) Development

Figure 3: Comparison of the combined model (MLN+SVM) with the pipeline model on the BioNLP'13 test and development data.

System	**Rec.**	**Prec.**	**F1**
Miwa12 (Miwa et al., 2012)	52.67	65.19	58.27
Our System	53.96	63.08	**58.16**
Riedel11 (Riedel et al., 2011)	–	–	57.4
Miwa10 (Miwa et al., 2010a)	50.13	64.16	56.28
Bjorne (Björne et al., 2009)	46.73	58.48	51.95
PoonMLN (Poon&Vanderwende,2010)	43.7	58.6	50.0
RiedelMLN (Riedel et al., 2009)	36.9	55.6	44.4

Table 5: Results on the BioNLP'09 test data. "–" indicates that the corresponding values are not known.

similar accuracy without the use of co-reference data. The F1 score of Miwa10, which does not use co-reference features, is nearly 2 points lower than that of our system. Our system also has a higher F1 score than Reidel11, which is the best joint inference-based system for this task.

On the regulation events, our system (47.55) outperforms both Miwa12 (45.99) and Riedel11 (46.9), while on the binding event, our system (59.88) is marginally worse than Miwa12 (59.91) and significantly better than Riedel11 (52.6). As mentioned earlier, these are the hardest events to extract. Also, existing MLN-based joint inference systems such as RiedelMLN and PoonMLN do not achieve state-of-the-art results because they do not leverage complex, high-dimensional features.

6 Summary and Future Work

Markov logic networks (MLNs) are a powerful representation that can compactly encode rich relational structures and ambiguities (uncertainty). As a result, they are an ideal representation for complex NLP tasks that require joint inference, such as event extraction. Unfortunately, the superior representational power greatly complicates inference and learning over MLN models. Even the most advanced methods for inference and learning in MLNs (Gogate and Domingos, 2011) are unable to handle complex, high-dimensional features, and therefore existing MLN systems primarily use low-dimensional features. This limitation severely affects the accuracy of MLN-based NLP systems, and as a result, in some cases their performance is inferior to pipeline methods that do not employ joint inference.

In this paper, we presented a general approach for exploiting the power of high-dimensional linguistic features in MLNs. Our approach involves reliably processing and learning high-dimensional features using SVMs and encoding their output as low-dimensional features in MLNs. We showed that we could achieve scalable learning and inference in our proposed MLN model by exploiting decomposition. Our results on the BioNLP shared tasks from '13, '11, and '09 clearly show that our proposed combination is extremely effective, achieving the best or second best score on all three datasets.

In future work, we plan to (1) improve our joint model by incorporating co-reference information and developing model ensembles; (2) transfer the results of this investigation to other complex NLP tasks that can potentially benefit from joint inference; and (3) develop scalable inference and learning algorithms (Ahmadi et al., 2013).

Acknowledgments

This work was supported in part by the AFRL under contract number FA8750-14-C-0021, by the ARO MURI grant W911NF-08-1-0242, and by the DARPA Probabilistic Programming for Advanced-Machine Learning Program under AFRL prime contract number FA8750-14-C-0005. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of DARPA, AFRL, ARO or the US government.

References

Babak Ahmadi, Kristian Kersting, Martin Mladenov, and Sriraam Natarajan. 2013. Exploiting symmetries for scaling loopy belief propagation and relational training. *Machine Learning*, 92(1):91–132.

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning About Time and Events*, pages 1–8.

Jari Björne and Tapio Salakoski. 2011. Generalizing biomedical event extraction. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 183–191.

Jari Björne and Tapio Salakoski. 2013. TEES 2.1: Automated annotation scheme learning in the bionlp 2013 shared task. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 16–25.

Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2009. Extracting complex biological events with rich graph-based feature sets. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 10–18.

Quoc-Chinh Bui, David Campos, Erik van Mulligen, and Jan Kors. 2013. A fast rule-based approach for biomedical event extraction. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 104–108.

Chen Chen and Vincent Ng. 2012. Joint modeling for Chinese event extraction with rich linguistic features. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 529–544.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8.

Pedro Domingos and Daniel Lowd. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool, San Rafael, CA.

Lise Getoor and Ben Taskar, editors. 2007. *Introduction to Statistical Relational Learning*. MIT Press.

Vibhav Gogate and Pedro Domingos. 2011. Probabilistic theorem proving. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 256–265.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. NYU's English ACE 2005 system description. In *Proceedings of the ACE 2005 Evaluation Workshop*. Washington.

Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 369–372.

Kai Hakala, Sofie Van Landeghem, Tapio Salakoski, Yves Van de Peer, and Filip Ginter. 2013. EVEX in ST'13: Application of a large-scale text mining resource to event extraction and network construction. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 26–34.

Ruihong Huang and Ellen Riloff. 2012a. Bootstrapped training of event extraction classifiers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 286–295.

Ruihong Huang and Ellen Riloff. 2012b. Modeling textual cohesion for event extraction. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 254–262.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, USA.

Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC bioinformatics*, 9(1):10.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 shared task on event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 1–9.

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, Ngan Nguyen, and Jun'ichi Tsujii. 2011a. Overview of BioNLP shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 1–6.

Jin-Dong Kim, Yue Wang, Toshihisa Takagi, and Akinori Yonezawa. 2011b. Overview of Genia event task in BioNLP shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 7–15.

Jin-Dong Kim, Yue Wang, and Yamamoto Yasunori. 2013. The Genia event extraction shared task, 2013 edition - overview. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 8–15.

Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Peifeng Li, Guodong Zhou, Qiaoming Zhu, and Libin Hou. 2012. Employing compositional semantics and discourse consistency in Chinese event extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1006–1016.

Lishuang Li, Yiwen Wang, and Degen Huang. 2013a. Improving feature-based biomedical event extraction system by integrating argument information. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 109–115.

Peifeng Li, Qiaoming Zhu, and Guodong Zhou. 2013b. Argument inference from relevant event mentions in Chinese argument extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1477–1487.

Qi Li, Heng Ji, and Liang Huang. 2013c. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 73–82.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797.

Shasha Liao and Ralph Grishman. 2011. Acquiring topic features to improve event extraction: in preselected and balanced collections. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 9–16.

Haibin Liu, Karin Verspoor, Donald C. Comeau, Andrew MacKinlay, and W John Wilbur. 2013. Generalizing an approximate subgraph matching-based system to extract events in molecular biology and cancer genetics. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 76–85.

Daniel Lowd and Pedro Domingos. 2007. Efficient weight learning for markov logic networks. In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 200–211.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 835–844.

Radu Marinescu and Rina Dechter. 2009. AND/OR branch-and-bound search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17):1457–1491.

David McClosky, Mihai Surdeanu, and Chris Manning. 2011a. Event extraction as dependency parsing. In *Proceedings of the Association for Computational Linguistics: Human Language Technologies*, pages 1626–1635.

David McClosky, Mihai Surdeanu, and Christopher Manning. 2011b. Event extraction as dependency parsing for BioNLP 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 41–45.

David McClosky. 2010. *Any domain parsing: Automatic domain adaptation for natural language parsing*. Ph.D. thesis, Ph.D. thesis, Brown University, Providence, RI.

Makoto Miwa, Sampo Pyysalo, Tadayoshi Hara, and Jun'ichi Tsujii. 2010a. Evaluating dependency representation for event extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 779–787.

Makoto Miwa, Rune Sætre, Jin-Dong Kim, and Jun'ichi Tsujii. 2010b. Event extraction with complex event classification using rich features. *Journal of Bioinformatics and Computational Biology*, 8(01):131–146.

Makoto Miwa, Paul Thompson, and Sophia Ananiadou. 2012. Boosting automatic event extraction from the literature using domain adaptation and coreference resolution. *Bioinformatics*, 28(13):1759–1765.

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

Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. 2013. Overview of BioNLP shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 1–7.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 151–160.

Hoifung Poon and Pedro Domingos. 2007. Joint inference in information extraction. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, pages 913–918.

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 813–821.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Chris Quirk, Pallavi Choudhury, Michael Gamon, and Lucy Vanderwende. 2011. MSR-NLP entry in BioNLP shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 155–163.

Sebastian Riedel and Andrew McCallum. 2011a. Fast and robust joint models for biomedical event extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1–12.

Sebastian Riedel and Andrew McCallum. 2011b. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 46–50.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A Markov logic approach to bio-molecular event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 41–49.

Sebastian Riedel, David McClosky, Mihai Surdeanu, Andrew McCallum, and Christopher D. Manning. 2011. Model combination for event extraction in bionlp 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 51–55.

Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. 2012. Open domain event extraction from Twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1104–1112.

Bart Selman, Henry Kautz, and Bram Cohen. 1996. Local Search Strategies for Satisfiability Testing. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, pages 521–532. American Mathematical Society, Washington, DC.

Parag Singla and Pedro Domingos. 2005. Discriminative training of Markov logic networks. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 868–873.

David Sontag and Amir Globerson. 2011. Introduction to Dual Decomposition for Inference. *Optimization for Machine Learning*.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21st International Conference on Machine Learning*, pages 104–112.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer, New York, NY.

Syllable weight encodes mostly the same information for English word segmentation as dictionary stress

John K Pate **Mark Johnson**
Centre for Language Technology
Macquarie University
Sydney, NSW, Australia
{john.pate,mark.johnson}@mq.edu.au

Abstract

Stress is a useful cue for English word segmentation. A wide range of computational models have found that stress cues enable a 2-10% improvement in segmentation accuracy, depending on the kind of model, by using input that has been annotated with stress using a pronouncing dictionary. However, stress is neither invariably produced nor unambiguously identifiable in real speech. Heavy syllables, i.e. those with long vowels or syllable codas, attract stress in English. We devise Adaptor Grammar word segmentation models that exploit either stress, or syllable weight, or both, and evaluate the utility of syllable weight as a cue to word boundaries. Our results suggest that syllable weight encodes largely the same information for word segmentation in English that annotated dictionary stress does.

1 Introduction

One of the first skills a child must develop in the course of language acquisition is the ability to segment speech into words. Stress has long been recognized as a useful cue for English word segmentation, following the observation that words in English are predominantly stress-initial (Cutler and Carter, 1987), together with the result that 9-month-old English-learning infants prefer stress-initial stimuli (Jusczyk et al., 1993). A range of statistical (Doyle and Levy, 2013; Christiansen et al., 1998; Börschinger and Johnson, 2014) and rule-based (Yang, 2004; Lignos and Yang, 2010) models have used stress information to improve word segmentation. However, that work uses stress-marked input prepared by marking vowels that are listed as stressed in a pronouncing dictionary. This pre-processing step glosses over the fact that stress identification itself involves a non-trival learning problem, since stress has many possible phonetic reflexes and no known invariants (Campbell and Beckman, 1997; Fry, 1955; Fry, 1958). One known strong correlate of stress in English is syllable weight: heavy syllables, which end in a consonant or have a long vowel, attract stress in English. We present experiments with Bayesian Adaptor Grammars (Johnson et al., 2007) that suggest syllable weight encodes largely the same information for word segmentation that dictionary stress information does.

Specifically, we modify the Adaptor Grammar word segmentation model of Börschinger and Johnson (2014) to compare the utility of syllable weight and stress cues for finding word boundaries, both individually and in combination. We describe how a shortcoming of Adaptor Grammars prevents us from comparing stress and weight cues in combination with the full range of phonotactic cues for word segmentation, and design two experiments to work around this limitation. The first experiment uses grammars that provide parallel analyses for syllable weight and stress, and learns initial/non-initial phonotactic distinctions. In this first experiment, syllable weight cues are actually more useful than stress cues at larger input sizes. The second experiment focuses on incorporating phonotactic cues for typical word-final consonant clusters (such as inflectional morphemes), at the expense of parallel structures. In this second experiment, weight cues merely match stress cues at larger input sizes, and the learning curve for the combined weight-and-stress grammar follows almost perfectly with the stress-only grammar. This second experiment suggests that the advantage of weight over stress in the first experiment was purely due to poor modeling of word-final consonant clusters by the stress-only grammar, not weight *per se*. All together, these results indicate that syllable weight

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 844–853,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

is highly redundant with dictionary-based stress for the purposes of English word segmentation; in fact, in our experiments, there is no detectable difference between relying on syllable weight and relying on dictionary stress.

2 Background

Stress is the perception that some syllables are more prominent than others, and reflects a complex, language-specific interaction between acoustic cues (such as loudness and duration), and phonological patterns (such as syllable shapes). The details on how stress is assigned, produced, and perceived vary greatly across languages. Three aspects of the English stress system are relevant for this paper. First, although English stress can shift in different contexts (Liberman and Prince, 1977), such as from the first syllable of 'fourteen' in isolation to the second syllable when followed by a stressed syllable, it is largely stable across different tokens of a given word. Second, most words in English end up being stress-initial on a type and token basis. Third, heavy syllables (those with a long vowel or a consonant coda) attract stress in English.

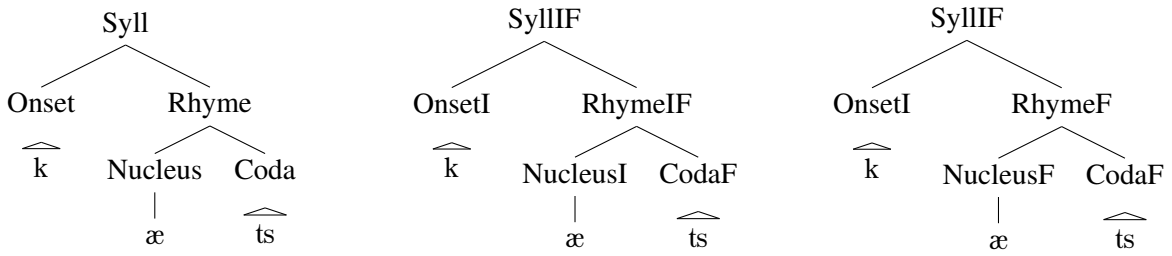
There is experimental evidence that English-learning infants prefer stress-initial words from around the age of seven months (Jusczyk et al., 1993; Juszcyk et al., 1999; Jusczyk et al., 1993; Thiessen and Saffran, 2003). A variety of computational models have subsequently been developed that take stress-annotated input and use this regularity to improve segmentation accuracy. The earliest Simple Recurrent Network (SRN) modeling experiments of Christiansen et al. (1998) and Christiansen and Curtin (1999) found that stress improved word segmentation from about 39% to 43% token f-score (see Evaluation). Rytting et al. (2010) applied the SRN model to probability distributions over phones obtained from a speech recognition system, and found that the entropy of the probability distribution over phones, as a proxy to local hyperarticulation and hence a stress cue, improved token f-score from about 16% to 23%. In a deterministic approach using pre-syllabified input, Yang (2004), with follow-ups in Lignos and Yang (2010) and Lignos (2011; 2012), showed that a 'Unique Stress Constraint' (USC), or assuming each word has at most one stressed syllable, leads to an improvement of about 2.5% boundary f-score.

Among explicitly probabilistic models, Doyle and Levy (2013) incorporated stress into Goldwater et al.'s (2009) Bigram model. They did this by modifying the base distribution over lexical forms to generate not simply phone strings but a sequence of syllables that may or may not be stressed. The resulting model can learn that some sequences of syllables (in particular, sequences that start with a stressed syllable) are more likely than others. However, observed stress improved token f-score by only 1%. Börschinger and Johnson (2014) used Adaptor Grammars (Johnson et al., 2007), a generalization of Goldwater et al.'s (2009) Bigram model that will be described shortly, and found a clearer 4-10% advantage in token f-score, depending on the amount of training data.

Together, the experimental and computational results suggest that infants in fact pay attention to stress, and that stress carries useful information for segmenting words in running speech. However, stress identification is itself a non-trivial task, as stress has many highly variable, context-sensitive, and optional phonetic reflexes. However, one strong phonological cue in English is syllable weight: heavy syllables attract stress. Heavy syllables, in turn, are syllables with a coda and/or a long vowel, which, in English, are tense vowels. Turk et al. (1995) replicated the Jusczyk et al. (1993) finding that English-learning infants prefer stress-initial stimuli (using non-words), and then examined how stress interacted with syllable weight. They found that syllable weight was not a necessary condition to trigger the preference: infants preferred stress-initial stimuli even if the initial syllable was light. However, they also found that infants most strongly preferred stimuli whose first syllable was both stressed and heavy: infants preferred stress-initial and heavy-initial stimuli to stress-initial and light-initial stimuli. This result suggests that infants are sensitive to syllable weight in determining typical stress and rythmic patterns in their language.

2.1 Models

We will adopt the Adaptor Grammar framework used by Börschinger and Johnson (2014) to explore the utility of syllable weight as a cue to word segmentation by way of its covariance with stress. Adaptor Grammars are Probabilistic Context Free Grammars (PCFGs) with a spe-



(a) Basic syllable. (b) Mono-syllable with initial Rhyme. (c) Mono-syllable with final Rhyme.

Figure 1: Different ways to incorporate phonotactics. It is not possible to capture word-final codas and word initial rhymes in monosyllabic words with factors the size of a PCFG rule.

cial set of *adapted* non-terminal nodes. We underline adapted non-terminals (X) to distinguish them from non-adapted non-terminals (Y). While a vanilla PCFG can only directly model regularities that are expressed by a single re-write rule, an Adaptor Grammar model caches entire subtrees that are rooted at adapted non-terminals. Adaptor Grammars can thus learn the internal structure of words, such as syllables, syllable onsets, and syllable rhymes, while still learning entire words as well.

In Adaptor Grammars, parameters are associated with PCFG rules. While this has been a useful factorization in previous work, it makes it difficult to integrate syllable weight and syllable stress in a linguistically natural way. A syllable is typically analyzed as having an optional onset followed by a rhyme, with the rhyme rewriting to a nucleus (the vowel) followed by an optional coda, as in Figure 1a. We expect stress and syllable weight to be useful primarily because initial syllables tend to be different from non-initial syllables. However, distinguishing final from non-final codas should be useful as well, due to the frequency of suffixes in English, and the importance of edge phenomena in phonology more generally (Brent and Cartwright, 1996). These principles come into conflict when modeling monosyllabic words. If we say that a monosyllable is an Initial and Final SyllIF, and has an initial Onset and an initial Rhyme, as in Figure 1b, then we can learn the initial/non-initial generalization about stressed or heavy rhymes at the expense of the generalization about final and non-final codas. If we say that a monosyllable is an initial onset with a final rhyme, the reverse occurs: we can learn the final/non-final coda generalization at the expense of the initial/non-initial regularities. If we split the symbols further, we'd generalize even less: we'd essentially have to learn the initial/non-initial patterns separately for monosyllables and polysyllables.

The most direct solution would introduce factors that are 'smaller' than a single PCFG rule. Essentially, we would compute the score of a PCFG rule in terms of multiple features of its right-hand side, rather than a single 'one-hot' feature identifying the expansion. We left this direction for future work and instead carried out two experiments using Adaptor Grammars that were designed to work around this limitation.

Our first experiment focuses on modeling the initial/non-initial distinction, leaving the final/non-final coda distinction unmodeled. The models in this experiment assume parallel structures for syllable weight and stress, and focus on providing the *most direct* comparison between syllable weight and stress with a strictly initial/non-initial distinction. This first experiment shows that observing dictionary stress is better early in learning, but that modeling syllable weight is better later in learning. However, it is possible that syllable weight was more useful because modeling syllable weight involves modeling the characteristics of codas; the advantage may not have been due to weight *per se* but due to having learned something about the effects of suffixes on final codas.

Our second experiment focuses on modeling some aspects of final codas at the expense of maintaining a rigid parallelism in the structures for syllable weight and stress. The models in this experiment split only those symbols that are necessary to bring stress or weight patterns into the expressive power of the model, and focus on comparing *richer* models of syllable weight and stress that account for inital/internal/final distinctions. This second experiment shows that observing dictionary stress is better early in learning, and that modeling syllable weight merely catches up to

$$\text{Sentence} \rightarrow \underline{\text{Collocations3}}^{+} \quad (1)$$
$$\underline{\text{Collocations3}} \rightarrow \underline{\text{Collocations2}}^{+} \quad (2)$$
$$\underline{\text{Collocations2}} \rightarrow \underline{\text{Collocation}}^{+} \quad (3)$$
$$\underline{\text{Collocation}} \rightarrow \underline{\text{Word}}^{+} \quad (4)$$

Figure 2: Three levels of collocation; symbols followed by $^{+}$ may occur one or more times.

stress without surpassing it. Moreover, a combined stress-and-weight model does no better than a stress model, suggesting that the weight grammar's contribution is fully redundant, for the purposes of word segmentation, with the stress observations.

Together, these experiments suggest that syllable weight eventually encodes everything about word segmentation that dictionary stress does, and that any advantage that syllable weight has over observing dictionary stress is entirely redundant with knowledge of word-final codas.

3 Experiments

3.1 Adaptor Grammars

We follow Börschinger and Johnson (2014) in using a 3-level collocation Adaptor Grammar, as introduced by Johnson and Goldwater (2009) and presented in Figure 2, as the backbone for all models, including the baseline. A 3-level collocation grammar assumes that words are grouped into collocations of words that tend to appear with each other, and that the collocations themselves are grouped into larger collocations, up to three levels of collocations. This collocational structure allows the model to capture strong word-to-word dependencies without having to group frequently-occuring word sequences into a single, incorrect, undersegmented 'word' as the unigram model tends to do (Johnson and Goldwater, 2009)

Word rewrites in different ways in Experiment I and Experiment II, which will be explained in the relevant experiment section.

3.2 Experimental Set-up

We applied the same experimental set-up used by Börschinger and Johnson (2014), to their dataset, as described below. To understand how different modeling assumptions interact with corpus size, we train on prefixes of each corpus with increasing input size: 100, 200, 500, 1,000, 2,000, 5,000, and 10,000 utterances. Inference closely followed Börschinger and Johnson (2014) and Johnson and Goldwater (2009). We set our hyperparameters to encourage onset maximization. The hyperparameter for syllable nodes to rewrite to an onset followed by a rhyme was 10, and the hyperparameter for syllable nodes to rewrite to a rhyme only was 1. Similarly, the hyperparameter for rhyme nodes to include a coda was 1, and the hyperparameter for rhyme nodes to exclude the coda was 10. All other hyperparameters specified vague priors. We ran eight chains of each model for 1,000 iterations, collecting 20 samples with a lag of 10 iterations between samples and a burn-in of 800 iterations. We used the same batch-initialization and table-label resampling to encourage the model to mix.

After gathering the samples, we used them to perform a single minimum Bayes risk decoding of a separate, held-out test set. This test set was constructed by taking the last 1,000 utterances of each corpus. We use a common test-set instead of just evaluating on the training data to ensure that performance figures are comparable across input sizes; when we see learning curves slope upward, we can be confident that the increase is due to learning rather than easier evaluation sets.

We measured our models' performance with the usual token f-score metric (Brent, 1999), the harmonic mean of how many proposed word tokens are correct (token precision) and how many of the actual word tokens are recovered (token recall). For example, a model may propose "the in side" when the true segmentation is "the inside." This segmentation would have a token precision of $\frac{1}{3}$, since one of three predicted words matches the true word token (even though the other predicted words are valid word types), and a token recall of $\frac{1}{2}$, since it correctly recovered one of two words, yield a token f-score of 0.4.

3.3 Dataset

We evaluated on a dataset drawn from the Alex portion of the Providence corpus (Demuth et al., 2006). This dataset contains $17,948$ utterances with $72,859$ word tokens directed to one child from the age of 16 months to 41 months. We used a version of this dataset that contained annotations of primary stress that Börschinger and Johnson (2014) added to this input using an extended

RhymeI → HeavyRhyme
RhymeI → LightRhyme
Rhyme → HeavyRhyme
Rhyme → LightRhyme
HeavyRhyme → LongVowel
HeavyRhyme → Vowel Coda
LightRhyme → ShortVowel

(a) Weight-sensitive grammar

RhymeI → RhymeS
RhymeI → RhymeU
Rhyme → RhymeS
Rhyme → RhymeU
RhymeS → Vowel Stress (Coda)
RhymeU → Vowel (Coda)

(b) Stress-sensitive grammar

RhymeI → Vowel (Coda)
Rhyme → Vowel (Coda)

(c) Baseline grammar

RhymeI → HeavyRhymeS
RhymeI → HeavyRhymeU
RhymeI → LightRhymeS
RhymeI → LightRhymeU
Rhyme → HeavyRhymeS
Rhyme → HeavyRhymeU
Rhyme → LightRhymeS
Rhyme → LightRhymeU
HeavyRhymeS → LongVowel Stress
HeavyRhymeS → LongVowel Stress Coda
HeavyRhymeU → LongVowel
HeavyRhymeU → LongVowel Coda
LightRhymeS → ShortVowel Stress
LightRhymeU → ShortVowel

(d) Combined grammar

Figure 3: Experiment I Grammars

version of CMUDict (cmu, 2008).[1] The mean number of syllables per word token was 1.2, and only three word tokens had more than five syllables. Of the $40,323$ word tokens with a stressed syllable, $27,258$ were monosyllabic. Of the $13,065$ polysyllabic word tokens with a stressed syllable, $9,931$ were stress-initial. Turning to the $32,536$ word tokens with no stress (i.e., the function words), all but 23 were monosyllabic (the 23 were primarily contractions, such as "couldn't").

3.4 Experiment I: Parallel Structures

The goal of this first experiment is to provide the most direct comparison possible between grammars that attend to stress cues and grammars that attend to syllable weight cues. As these are both hypothesized to be useful by way of an initial/non-initial distinction, we defined a word to be an initial syllable SyllI followed by zero to three syllables, and syllables to consist of an optional onset and a rhyme:

$$\underline{\text{Word}} \rightarrow \text{SyllI} \quad (\text{Syll})^{\{0,3\}} \tag{5}$$

$$\text{SyllI} \rightarrow (\underline{\text{OnsetI}}) \quad \text{RhymeI} \tag{6}$$

$$\text{Syll} \rightarrow (\underline{\text{Onset}}) \quad \text{Rhyme} \tag{7}$$

In the baseline grammar, presented in Figure 3c, rhymes rewrite to a vowel followed by an optional consonant coda. Rhymes then rewrite to be heavy or light in the weight grammar, as in Figure 3a, to be stressed or unstressed in the stress grammar, as in Figure 3b. In the combination grammar, rhymes rewrite to be heavy or light and stressed or unstressed, as in Figure 3d. LongVowel and ShortVowel both re-write to all vowels. An additional grammar that restricted them to rewrite to long and short vowels, respectively, led to virtually identical performance, suggesting that vowel quantity can be learned for the purposes of word segmentation from distributional cues. We will also present evidence that the model did manage to learn most of the contrast.

Figure 4 presents learning curves for the grammars in this parallel structured comparison. We see that observing stress without modeling weight

[1] This dataset and these Adaptor Grammar models are available at: `http://web.science.mq.edu.au/~jpate/stress/`

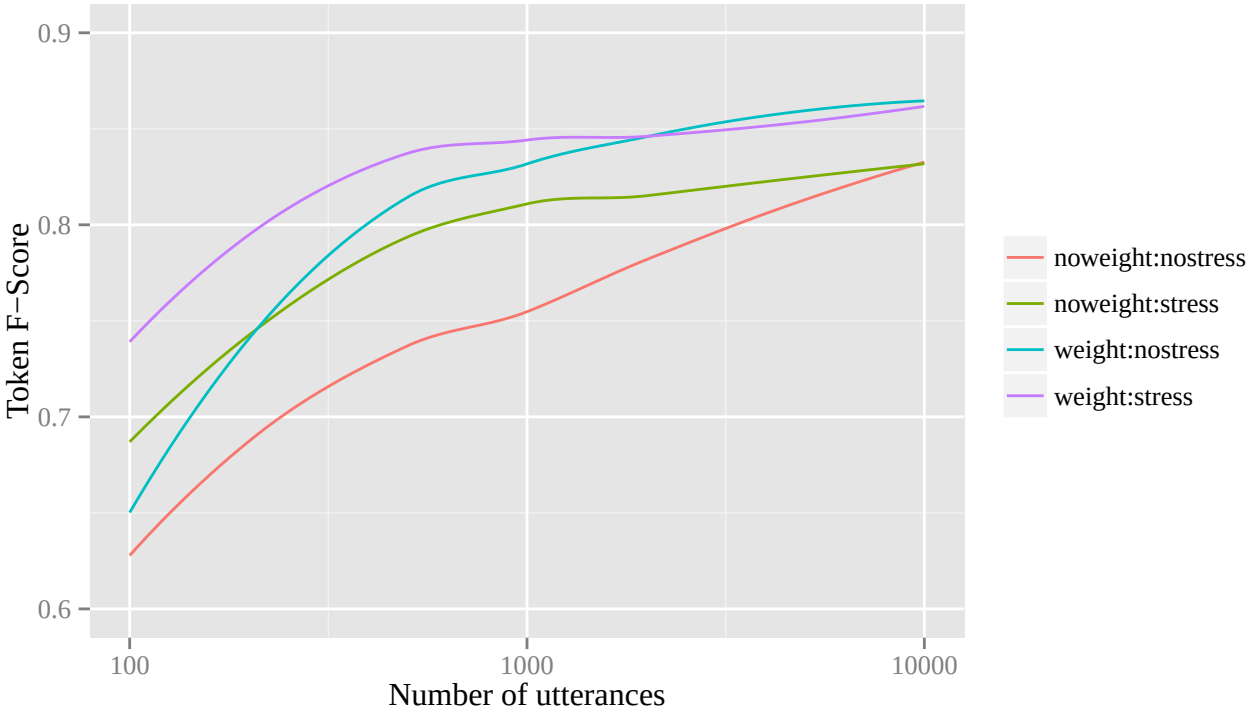


Figure 4: Learning curves on the Alex corpus for Experiment I grammars with parallel distinctions between Stressed/Unstressed and Heavy/Light syllable rhymes.

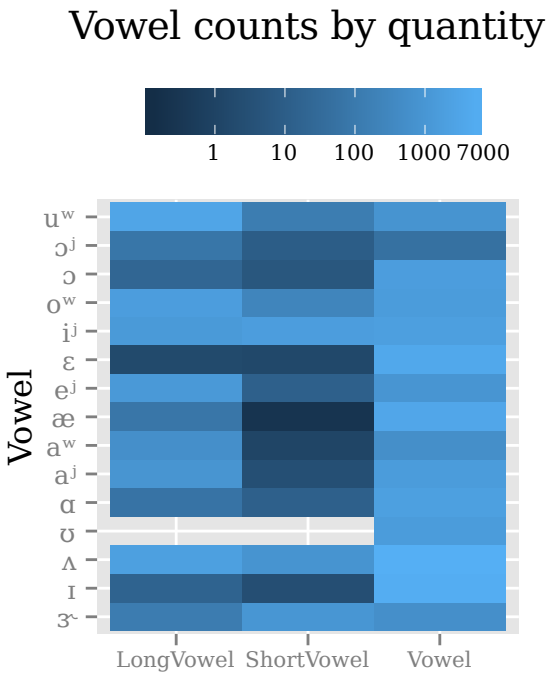


Figure 5: Heatmap of learned vowels in the Experiment I weight-only grammar. Each cell corresponds to the count of a particular vowel being analyzed as one of the three vowel types. Diphthongs are rarely ShortVowel.

outperforms both the baseline and the weight-only grammar early in learning. The weight-only grammar rapidly improves in performance at larger training data sizes, increasing its advantage over the baseline, while the advantage of the stress-only grammar slows and appears to disappear at the largest training data size. At 10,000 utterances, the improvement of the weight-only grammar over the stress-only grammar is significant according to an independent samples t-test ($t = 7.2, p < 0.001$, 14 degrees of freedom). This pattern suggests that annotated dictionary stress is easy to take advantage of at low data sizes, but that, with sufficient data, syllable weight can provide even more information about word boundaries. The best overall performance early in learning is obtained by the combined grammar, suggesting that syllable weight and dictionary stress provide information about word segmentation that is not redundant.

An examination of the final segmentation suggests that the weight grammar has learned that initial syllables tend to be heavy. Specifically, across eight runs, 98.1% of RhymeI symbols rewrote to HeavyRhyme, whereas only 54.5% of Rhyme symbols (i.e. non-initial rhymes) rewrote to HeavyRhyme.

Model	Mean TF	Std. Dev.
noweight:nostress	0.830	0.005
noweight:stress	0.831	0.008
weight:nostress	0.861	0.008
weight:stress	0.861	0.008

Table 1: Segmentation Token F-score for Experiment I at 10,000 utterances across eight runs.

We also examined the final segmentation to see well the model learned the distinction between long vowels and short vowels. Figure 5 presents a heatmap, with colors on a log-scale, showing how many times each vowel label rewrote to each possible vowel in the (translated to IPA). Although the quantity generalisations are not perfect, we do see a general trend where ShortVowel rarely rewrites to diphthongs.

3.5 Experiment II: Word-final Codas

Experiment I suggested that, under a basic initial/non-initial distinction, syllable weight eventually encodes more information about word boundaries than does dictionary stress. This is a surprising result, since we initially investigated syllable weight as a noisy proxy for dictionary stress. One possible source of the 'extra' advantage that the syllable weight grammar exhibited has to do with the importance of word-final codas, which can encode word-final morphemes in English (Brent and Cartwright, 1996). Even though the grammars did not explicitly model them, the weight grammar could implicitly capture a bias for or against having a coda in non-initial position, while the stress grammar could not. This is because most word tokens are one or two syllables, and only one of the two rhyme types of the weight grammar included a coda. Thus, the HeavyRhyme symbol could simultaneously capture the most important aspects of both stress and coda constraints.

To see if the extra advantage of the syllable weight grammar can be attributed to the influence of word-final codas, we formulated a set of grammars that model word-final codas and also can learn stress and/or syllable weight patterns. These grammars are more similar in structure to the ones that Börschinger and Johnson (2014) used. For the baseline and weight grammar, we again defined words to consist of up to four syllables with an initial SyllI syllable, but this time distinguished final syllables SyllF in polysyllabic words. The non-stress grammars use the following rules for producing syllables:

$$\underline{\text{Word}} \rightarrow \text{SyllIF} \tag{8}$$

$$\underline{\text{Word}} \rightarrow \text{SyllI} \quad (\text{Syll})^{\{0,2\}} \quad \text{SyllF} \tag{9}$$

$$\text{SyllIF} \rightarrow (\text{OnsetI}) \quad \text{RhymeI} \tag{10}$$

$$\text{SyllI} \rightarrow (\text{OnsetI}) \quad \text{RhymeI} \tag{11}$$

$$\text{Syll} \rightarrow (\text{Onset}) \quad \text{Rhyme} \tag{12}$$

$$\text{SyllF} \rightarrow (\text{Onset}) \quad \text{RhymeF} \tag{13}$$

For the stress grammar, we followed Börschinger and Johnson (2014) in distinguishing stressed and unstressed syllables, rather than simply stressed rhymes as in Experiment I, to allow the model to learn likely stress patterns at the word level. A word can consist of up to four syllables, and any syllable and any number of syllables may be stressed, as in Figure 6a.

The baseline grammar is similar to the previous one, except it distinguishes word-final codas, as in Figure 6b. The weight grammar, presented in Figure 6c, rewrites rhymes to a nucleus followed by an optional coda and distinguishes nuclei in open syllables according to their position in the word. The stress grammar, presented in Figure 6d, is the all-stress-patterns model (without the unique stress constraint) Börschinger and Johnson (2014). This grammar introduces additional distinctions at the syllable level to learn likely stress patterns, and distinguishes final from non-final codas. The combined model is identical to the stress model, except Vowel non-terminals in closed and word-internal syllables are replaced with Nucleus non-terminals, and Vowel non-terminals in word-inital (-final) open syllables are replaced with NucleusI (NucleusF) non-terminals.

To summarize, the stress models distinguish stressed and unstressed syllables in initial, final, and internal position. The weight models distinguish the vowels of initial open syllables, the vowels of final open syllables, and other vowels, allowing them to take advantage of an important cue from syllable weight for word segmentation: if an initial vowel is open, it should usually be long.

Figure 7 shows segmentation performance on the Alex corpus with these more complete models. While the performance of the weight grammars is virtually unchanged compared to Figure 4, the two grammars that do not model syllable weight improve dramatically. This result supports our proposal that much of the advantage of the weight

$$\underline{\text{Word}} \rightarrow \{\text{SyllUIF}|\text{SyllSIF}\}$$
$$\underline{\text{Word}} \rightarrow \{\text{SyllUI}|\text{SyllSI}\} \quad \{\text{SyllU}|\text{SyllS}\}^{\{0,2\}} \quad \{\text{SyllUF}|\text{SyllSF}\}$$

(a) The all-patterns stress model

$$\text{Rhyme} \rightarrow \text{Vowel} \quad (\underline{\text{Coda}})$$
$$\text{RhymeF} \rightarrow \text{Vowel} \quad (\underline{\text{CodaF}})$$

(b) Baseline grammar

$$\text{RhymeI} \rightarrow \underline{\text{NucleusI}}$$
$$\text{RhymeI} \rightarrow \underline{\text{Nucleus}} \quad \underline{\text{Coda}}$$
$$\text{Rhyme} \rightarrow \underline{\text{Nucleus}} \quad (\underline{\text{Coda}})$$
$$\text{RhymeF} \rightarrow \underline{\text{NucleusF}}$$
$$\text{RhymeF} \rightarrow \underline{\text{Nucleus}} \quad \underline{\text{CodaF}}$$

(c) Weight-sensitive grammar

$$\text{SyllSIF} \rightarrow \underline{\text{OnsetI}} \quad \text{RhymeSF}$$
$$\text{SyllUIF} \rightarrow \underline{\text{OnsetI}} \quad \text{RhymeUF}$$
$$\text{SyllSI} \rightarrow \underline{\text{Onset}} \quad \text{RhymeS}$$
$$\text{SyllUI} \rightarrow \underline{\text{Onset}} \quad \text{RhymeU}$$
$$\text{SyllSF} \rightarrow \underline{\text{Onset}} \quad \text{RhymeSF}$$
$$\text{SyllUF} \rightarrow \underline{\text{Onset}} \quad \text{RhymeUF}$$
$$\text{RhymeSI} \rightarrow \text{Vowel} \quad \text{Stress} \quad (\underline{\text{Coda}})$$
$$\text{RhymeUI} \rightarrow \text{Vowel} \quad (\underline{\text{Coda}})$$
$$\text{RhymeS} \rightarrow \text{Vowel} \quad \text{Stress} \quad (\underline{\text{Coda}})$$
$$\text{RhymeU} \rightarrow \text{Vowel} \quad (\underline{\text{Coda}})$$
$$\text{RhymeSF} \rightarrow \text{Vowel} \quad \text{Stress} \quad (\underline{\text{CodaF}})$$
$$\text{RhymeUF} \rightarrow \text{Vowel} \quad (\underline{\text{CodaF}})$$

(d) Stress-sensitive grammar

Figure 6: Experiment II Grammars.

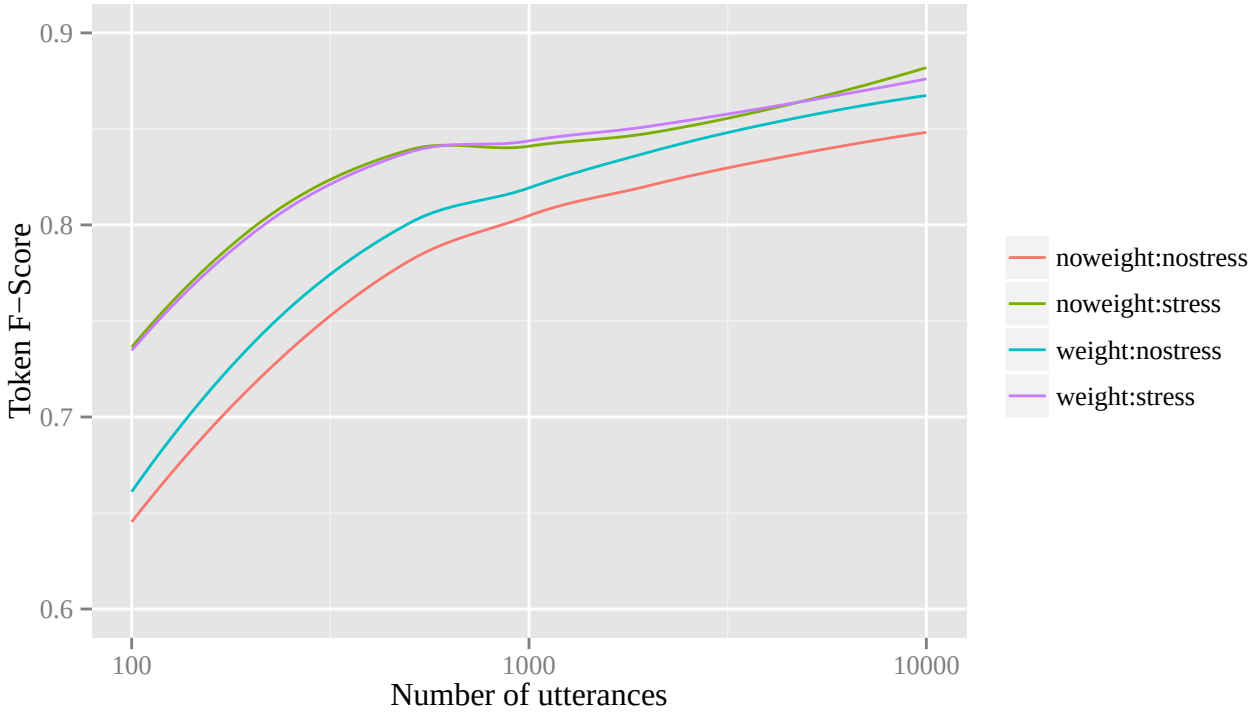


Figure 7: Learning curves on the Alex corpus for Experiment II grammars with word-final phonotactics that exploit Stress and Weight.

Model	Mean TF	Std. Dev.
noweight:nostress	0.846	0.007
noweight:stress	0.880	0.005
weight:nostress	0.865	0.011
weight:stress	0.875	0.005

Table 2: Segmentation Token F-score for Experiment II at 10,000 utterances across eight runs.

grammars over stress in Experiment I was due to modeling of word-final coda phonotactics.

Table 2 presents token f-score at 10,000 training utterances averaged across eight runs, along with the standard deviation in f-score. We see that the noweight:nostress grammar is several standard deviations than the grammars that model syllable weight and/or stress, while the syllable weight and/or stress grammars exhibit a high degree of overlap.

4 Conclusion

We have presented computational modeling experiments that suggest that syllable weight (eventually) encodes nearly everything about word segmentation that dictionary stress does. Indeed, our experiments did not find a persistent advantage to observing stress over modeling syllable weight. While it is possible that a different modeling approach might find such a persistent advantage, this advantage could not provide more than 13% absolute F-score. This result suggests that children may be able to learn and exploit important rhythm cues to word boundaries purely on the basis of segmental input. However, this result also suggests that annotating input with dictionary stress has missed important aspects of the role of stress in word segmentation. As mentioned, Turk et al. (1995) found that infants preferred initial light syllables to be stressed. Such a preference obviously cannot be learned by attending to syllable weight alone, so infants who have learned weight distinctions must also be sensitive to non-segmental acoustic correlates to stress. There was no long-term advantage to observing stress in addition to attending to syllable weight in our models, however, suggesting that annotated dictionary stress does not capture the relevant non-segmental phonetic detail. More modeling is necessary to assess the non-segmental phonetic features that distinguish stressed light syllables from unstressed light syllables.

This investigation also highlighted a weakness of current Adaptor Grammar models: the 'smallest' factors are the size of one PCFG rule. Allowing further factorizations, perhaps using feature functions of a rule's right-hand side, would allow models to capture finer-grained distinctions without fully splitting the symbols that are involved.

References

Benjamin Börschinger and Mark Johnson. 2014. Exploring the role of stress in Bayesian word segmentation using Adaptor Grammars. *Transactions of the ACL*, 2:93–104.

Michael R Brent and Timothy A Cartwright. 1996. Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition*, 61:93–125.

Michael Brent. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105.

Nick Campbell and Mary Beckman. 1997. Stress, prominence, and spectral tilt. In *Proceedings of an ESCA workshop*, pages 67–70, Athens, Greece.

Morten H. Christiansen and Suzanne L Curtin. 1999. The power of statistical learning: No need for algebraic rules. In *Proceedings of the 21st annual conference of the Cognitive Science Society*.

Morten H. Christiansen, Joseph Allen, and Mark S. Seidenberg. 1998. Learning to segment speech using multiple cues: A connectionist model. *Language and Cognitive Processes*, 13:221—268.

2008. The CMU pronouncing dictionary. `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`.

Anne Cutler and David M Carter. 1987. The predominance of strong initial syllables in the English vocabulary. *Computer Speech & Language*, 2(3):133–142.

Katherine Demuth, Jennifer Culbertson, and Jennifer Alter. 2006. Word-minimality, epenthesis, and coda licensing in the acquisition of English. *Language and Speech*, 49:137–174.

Gabriel Doyle and Roger Levy. 2013. Combining multiple information types in Bayesian word segmentation. In *Proceedings of NAACL 2013*, pages 117–126. Association for Computational Linguistics.

D B Fry. 1955. Duration and intensity as physical correlates of linguistic stress. *J. Acoust. Soc. of Am.*, 27:765–768.

D B Fry. 1958. Experiments in the perception of stress. *Language and Speech*, 1:126–152.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.

Mark Johnson and Sharon Goldwater. 2009. Improving nonparametric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325. Association for Computational Linguistics.

Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. 2007. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In B Schoelkopf, J Platt, and T Hoffmann, editors, *Advances in Neural Information Processing Systems*, volume 19. The MIT Press.

Peter W Jusczyk, Anne Cutler, and Nancy J Redanz. 1993. Infants' preference for the predominant stress patterns of English words. *Child Development*, 64(3):675–687.

Peter W Juszcyk, Derek M Houston, and Mary Newsome. 1999. The beginnings of word segmentation in English-learning infants. *Cognitive Psychology*, 39(3–4):159–207.

Mark Liberman and Alan Prince. 1977. On stress and linguistic rhythm. *Linguistic Inquiry*, 8(2):249–336, Spring.

Constantine Lignos and Charles Yang. 2010. Recession segmentation: simpler online word segmentation using limited resources. In *Proceedings of ACL 2010*, pages 88–97. Association for Computational Linguistics.

Constantine Lignos. 2011. Modeling infant word segmentation. In *Proceedings of the fifteenth conference on computational natural language learning*, pages 29–38. Association for Computational Linguistics.

Constantine Lignos. 2012. Infant word segmentation: An incremental, integrated model. In *Proceedings of the West Coast Conference on Formal Linguistics 30*.

C Anton Rytting, Chris Brew, and Eric Fosler-Lussier. 2010. Segmenting words from natural speech: subsegmental variation in segmental cues. *Journal of Child Language*, 37(3):513–543.

Erik D Thiessen and Jenny R Saffran. 2003. When cues collide: use of stress and statistical cues to word boundaries by 7-to-9-month-old infants. *Developmental Psychology*, 39(4):706–716.

Alice Turk, Peter W Jusczyk, and Louann Gerken. 1995. Do English-learning infants use syllable weight to determine stress? *Language and Speech*, 38(2):143–158.

Charles Yang. 2004. Universal grammar, statistics or both? *Trends in Cognitive Science*, 8(10):451–456.

A Joint Model for Unsupervised Chinese Word Segmentation

Miaohong Chen Baobao Chang Wenzhe Pei
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
Beijing, P.R.China, 100871
miaohong-chen@foxmail.com,{chbb,peiwenzhe}@pku.edu.cn

Abstract

In this paper, we propose a joint model for unsupervised Chinese word segmentation (CWS). Inspired by the "products of experts" idea, our joint model firstly combines two generative models, which are word-based hierarchical Dirichlet process model and character-based hidden Markov model, by simply multiplying their probabilities together. Gibbs sampling is used for model inference. In order to further combine the strength of goodness-based model, we then integrated nVBE into our joint model by using it to initializing the Gibbs sampler. We conduct our experiments on PKU and MSRA datasets provided by the second SIGHAN bakeoff. Test results on these two datasets show that the joint model achieves much better results than all of its component models. Statistical significance tests also show that it is significantly better than state-of-the-art systems, achieving the highest F-scores. Finally, analysis indicates that compared with nVBE and HDP, the joint model has a stronger ability to solve both combinational and overlapping ambiguities in Chinese word segmentation.

1 Introduction

Unlike English and many other western languages, there are no explicit word boundaries in Chinese sentences. Therefore, word segmentation is a crucial first step for many Chinese language processing tasks such as syntactic parsing, information retrieval and machine translation. A great deal of supervised methods have been proposed for Chinese word segmentation. While successful, they require manually labeled resources and often suffer from issues like poor domain adaptability. Thus, unsupervised word segmentation methods are still attractive to researchers due to its independence on domain and manually labeled corpora.

Previous unsupervised approaches to word segmentation can be roughly classified into two types. The first type uses carefully designed goodness measure to identify word candidates. Popular goodness measures include description length gain (DLG) (Kit and Wilks, 1999), accessor variety (AV) (Feng et al., 2004), boundary entropy (BE) (Jin and Tanaka-Ishii, 2006) and normalized variation of branching entropy (nVBE) (Magistry and Sagot, 2012) etc. Goodness measure based model is not segmentation model in a very strict meaning and is actually strong in generating word list without supervision. It inherently lacks capability to deal with ambiguous string, which is one of main sources of segmentation errors and has been extensively explored in supervised Chinese word segmentation.

The second type focuses on designing sophisticated statistical model, usually nonparametric Bayesian models, to find the segmentation with highest posterior probability, given the observed character sequences. Typical statistical models includes Hierarchical Dirichlet process (HDP) model (Goldwater et al., 2009), Nested Pitman-Yor process (NPY) model (Mochihashi et al., 2009) etc, which are actually nonparametric language models and therefor can be categorized as word-based model. Word-based model makes decision on wordhood of a candidate character sequence mainly based on information outside the sequence, namely, the wordhood of character sequences being adjacent to the concerned sequence.

Inspired by the success of character-based model in supervised word segmentation, we propose a Bayesian HMM model for unsupervised Chinese word segmentation. With the Bayesian HMM model, we formulate the unsupervised segmentation tasks as procedure of tagging positional

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 854–863,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

tags to characters. Different from word-based model, character-based model like HMM-based model as we propose make decisions on wordhood of a candidate character sequence based on information inside the sequence, namely, ability of characters to form words. Although the Bayesian HMM model alone does not produce competitive results, it contributes substantially to the joint model as proposed in this paper.

Our joint model takes advantage from three different models: namely, a character-based model (HMM-based), a word-based model (HDP-based) and a goodness measure based model (nVBE model). The combination of HDP-based model and HMM-based model enables to utilize information of both word-level and character-level. We also show that using nVBE model as initialization model could further improve the performance to outperform the state-of-the-art systems and leads to improvement in both wordhood judgment and disambiguation ability.

Word segmentation systems are usually evaluated with metrics like precision, recall and F-Score, regardless of supervised or unsupervised. Following normal practice, we evaluate our model and compare it with state-of-the-art systems using F-Score. However, we argue that the ability to solve segmentation ambiguities is also important when evaluating different types of unsupervised word segmentation systems.

This paper is organized as follows. In Section 2, we will introduce several related systems for unsupervised word segmentation. Then our joint model is presented in Section 3. Section 4 shows our experiment results on the benchmark datasets and Section 5 concludes the paper.

2 Related Work

Unsupervised Chinese word segmentation has been explored in a number of previous works and by various methods. Most of these methods can be divided into two categories: goodness measure based methods and nonparametric Bayesian methods.

There have been a plenty of work that is based on a specific goodness measure. Zhao and Kit (2008) compared several popular unsupervised models within a unified framework. They tried various types of goodness measures, such as Description Length Gain (DLG) proposed by Kit and Wilks (1999), Accessor Variety (AV) proposed by Feng et al. (2004) and Boundary Entropy (Jin and Tanaka-Ishii, 2006). A notable goodness-based method is ESA: "Evaluation, Selection, Adjustment", which is proposed by Wang et al. (2011) for unsupervised Mandarin Chinese word segmentation. ESA is an iterative model based on a new goodness algorithm that adopts a local maximum strategy and avoids threshold setting. One disadvantage of ESA is that it needs to iterate the process several times on the corpus to get good performance. Another disadvantage is the requirement for a manually segmented training corpus to find best value for parameters (they called it *proper exponent*). Another notable work is nVBE: Magistry and Sagot (2012) proposed a model based on the Variation of Branching Entropy. By adding normalization and viterbi decoding, they improve performance over Jin and Tanaka-Ishii (2006) and remove most of the parameters and thresholds from the model.

Nonparametric Bayesian models also achieved state-of-the-art performance in unsupervised word segmentation. Goldwater et al. (2009) introduced a unigram and a bigram model for unsupervised word segmentation, which are based on Dirichlet process and hierarchical Dirichlet process (Teh et al., 2006) respectively. The main drawback is that it needs almost 20,000 iterations before the Gibbs sampler converges. Mochihashi et al. (2009) extended this method by introducing a nested character model and an efficient blocked Gibbs sampler. Their method is based on what they called nested Pitman-Yor language model.

One disadvantage of goodness measure based methods is that they do not have any disambiguation ability in theory in spite of their competitive performances. This is because once the goodness measure is given, the decoding algorithm will segment any ambiguous strings into the same word sequences, no matter what their context is. In contrast, nonparametric Bayesian language models aim to segment character string into a "reasonable" sentence according to the posterior probability. Thus, theoretically, this method should have better ability to solve ambiguities over goodness measure based methods.

3 Joint Model

In this section, we will discuss our joint model in detail.

3.1 Combining HDP and HMM

In supervised Chinese word segmentation literature, word-based approaches and character-based approaches often have complementary advantages (Wang et al., 2010).Since the two types of model try to solve the problem from different perspectives and by utilizing different levels of information (word level and character level). In unsupervised Chinese word segmentation literature, the HDP-base model can be viewed as a typical word-based method. And we can also build a character-based unsupervised model by using a hidden Markov model. We believe that the HDP-based model and the HMM-based model are also complementary with each other, and a combination of them will take advantage of both and thus capture different levels of information.

Now the problem we are facing is how to combine these two models. To keep the joint model simple and involve as little extra parameters as possible, we combine the two baseline models by just multiplying their probabilities together and then renormalizing it. Let $C = c_1 c_2 \cdots c_{|C|}$ be a string of characters and $W = w_1 w_2 \cdots w_{|W|}$ is the corresponding segmented words sequence. Then the conditional probability of the segmentation W given the character string C in our joint model is defined as:

$$P_J(W|C) = \frac{1}{Z(C)} P_D(W|C) P_M(W|C) \quad (1)$$

where $P_D(W|C)$ is the probability from the HDP model as given in Equation 6 and $P_M(W|C)$ is the probability given by the Bayesian HMM model as given in Equation 2. $Z(C)$ is a normalization term to make sure that $P_J(W|C)$ is a probability distribution. The combining method is inspired by Hinton (1999), which proved that it is possible to combine many individual expert models by multiplying the probabilities and then renormalizing it. They called it "product of experts". We can see that combining models in this way does not involve any extra parameters and Gibbs sampling can be easily used for model inference.

3.2 Bayesian HMM

The dominant method for supervised Chinese word segmentation is character-based model which was first proposed by Xue (2003). This method treats word segmentation as a tagging problem, each tag indicates the position of a character within a word. The most commonly used tag set is {**S**ingle, **B**egin, **M**iddle, **E**nd}. Specifically, **S** means the character forms a single word, **B/E** means the character is the begining/ending character of the word, and **M** means the character is in the middle of the word. Existing models are trained on manually annotated data in a supervised way based on discriminative models such as Conditional Random Fields (Peng et al., 2004; Tseng et al., 2005). Supervised character-based methods make full use of character level information and thus have been very successful in the last decade. However, no unsupervised model has utilized character level information in the way as supervised method does.

We can also build a character-based model for Chinese word segmentation using hidden Markov model(HMM) as formulated in the following equation:

$$P_M(W|C) = \prod_{i=1}^{|C|} P_t(t_i|t_{i-1}) P_e(c_i|t_i) \quad (2)$$

where C and W have the same meaning as before. $P_t(t_i|t_{i-1})$ is the transition probability of tag t_i given its former tag t_{i-1} and $P_e(c_i|t_i)$ is the emission probability of character c_i given its tag t_i. This model can be easily trained with Maximum Likelihood Estimation (MLE) on annotated data or with Expectation Maximization (EM) on raw texts. But using any of this methods will make it difficult to combine it with the HDP-based model. Instead, we propose a Bayesian HMM for unsupervised word segmentation. The Bayesian HMM model is defined as follows:

$$\begin{aligned} t_i|t_{i-1} = t, p^t &\sim Mult(p^t) \\ c_i|t_i = t, e^t &\sim Mult(e^t) \\ p^t|\theta &\sim Dirichlet(\theta) \\ e^t|\sigma &\sim Dirichlet(\sigma) \end{aligned}$$

where p^t and e^t are transition and emission distributions, θ and σ are the symmetric parameters of Dirichlet distributions. Now suppose we have observed tagged text h, then the conditional probability $P_M(w_i|w_{i-1} = l, h)$ can be obtained:

$$\begin{aligned} &P_M(w_i|w_{i-1} = l, h) \\ &\quad = \prod_{j=1}^{|w_i|} P_t(t_j|t_{j-1}, h) P_e(c_j|t_j, h) \quad (3) \end{aligned}$$

where $< w_{i-1}, w_i >$ is a word bigram, l is the index of word w_{i-1}, c_j is the jth character in word

w_i and t_j is the corresponding tag.$P_t(t_j|t_{j-1}, h)$ and $P_e(c_j|t_j, h)$ are the posterior probabilities, they are given as:

$$P_t(t_j|t_{j-1}, h) = \frac{n_{<t_{j-1},t_j>} + \theta}{n_{<t_{j-1},*>} + T\theta} \quad (4)$$

$$P_e(c_j|t_j, h) = \frac{n_{<t_j,c_j>} + \sigma}{n_{<t_j,*>} + V\sigma} \quad (5)$$

where $n_{<t_{j-1},t_j>}$ is the tag bigram count of $< t_{j-1}, t_j >$ in h, $n_{<t_j,c_j>}$ denotes the number of occurrences of tag t_j and character c_j, and $*$ means a sum operation. T and V are the size of character tag set (we follow the commonly used {**SBME**} tag set and thus $T = 4$ in this case) and character vocabulary.

3.3 HDP Model

Goldwater et al. (2009) proposed a nonparametric Bayesian model for unsupervised word segmentation which is based on HDP (Teh et al., 2006). In this model, the conditional probability of the segmentation W given the character string C is defined as:

$$P_D(W|C) = \prod_{i=0}^{|W|} P_D(w_i|w_{i-1}) \quad (6)$$

where w_i is the ith word in W. This is actually a nonparametric bigram language model. This bigram model assumes that each different word has a different distribution over words following it, but all these different distributions are linked through a HDP model:

$$\begin{aligned} w_i|w_{i-1} = l &\sim G_l \\ G_l &\sim DP(\alpha_1, G_0) \\ G_0 &\sim DP(\alpha, H) \end{aligned}$$

where DP denotes a Dirichlet process.

Suppose we have observed segmentation result h, then we can get the posterior probability $P_D(w_i|w_{i-1} = l, h)$ by integrating out G_l:

$$\begin{aligned} &P_D(w_i|w_{i-1} = l, h) \\ &\quad = \frac{n_{<w_{i-1},w_i>} + \alpha_1 P_D(w_i|h)}{n_{<w_{i-1},*>} + \alpha_1} \end{aligned} \quad (7)$$

where $n_{<w_{i-1},w_i>}$ denotes the total number of occurrences of the bigram $< w_{i-1}, w_i >$ in the observation h. And $P_D(w_i|h)$ can be got by integrating out G_0:

$$P_D(w_i|h) = \frac{t_{w_i} + \alpha H(w_i)}{t + \alpha} \quad (8)$$

where t_{w_i} denotes the number of tables associated with w_i in the Chinese Restaurant Franchise metaphor (Teh et al., 2006), t is the total number of tables and $H(w_i)$ is the base measure of G_0. In fact, $H(w_i)$ is the prior distribution over words, so prior knowledge can be injected in this distribution to enhance the performance.

In Goldwater et al. (2009)'s work, the base measure $H(w_i)$ are defined as a character unigram model:

$$H(w_i) = (1 - p_s)^{|w_i|-1} p_s \prod_j P(c_{ij})$$

where, p_s is the probability of generating a word boundary. $P(c_{ij})$ is the probability of the jth character c_{ij} in word w_i, this probability can be estimated from the training data using maximum likelihood estimation.

3.4 Initializing with nVBE

Among various goodness measure based models, we choose nVBE (Magistry and Sagot, 2012) to initialize our Gibbs sampler with its segmentation results. nVBE achieved a relatively high perfomance over other goodness measure based methods. And it's very simple as well as efficient.

Theoretically, the Gibbs sampler may be initialized at random or using any other methods. Initialization does not make a difference since the Gibbs sampler will eventually converge to the posterior distribution if it iterates as much as possible. This is an essential attribute of Gibbs sampling. However, we believe that initializing the Gibbs sampler with the result of nVBE will benefit us in two ways. On one hand, in consideration of its combination of nonparametric Bayesian method and goodness-based method, it will improve the overall performance as well as solve more segmentation ambiguities with the help of HDP-based model. On the other hand, it makes the convergence of Gibbs sampling faster. In practice, random initialization often leads to extremely slow convergence.

3.5 Inference with Gibbs Sampling

In our proposed joint model, Gibbs sampling (Casella and George, 1992) can be easily used to identify the highest probability segmentation from among all possibilities. Following Goldwater et al. (2009), we can repeatedly sample from potential word boundaries. Each boundary

variable can only take on two possible values, corresponding to a word boundary or not word boundary.

For instance, suppose we have obtained a segmentation result $\beta|c_{i-2}c_{i-1}c_ic_{i+1}c_{i+2}|\gamma$, where β and γ are the words sequences to the left and right and $c_{i-2}c_{i-1}c_ic_{i+1}c_{i+2}$ are characters between them. Now we are sampling at location i to decide whether there is a word boundary between c_i and c_{i+1}. Denote h_1 as the hypothesis that it forms a word boundary (the corresponding result is $\beta w_1 w_2 \gamma$ where $w_1 = c_{i-2}c_{i-1}c_i$ and $w_2 = c_{i+1}c_{i+2}$), and h_2 as the opposite hypothesis (then the corresponding result is $\beta w \gamma$ where $w = c_{i-2}c_{i-1}c_ic_{i+1}c_{i+2}$). The posterior probability for these two hypotheses would be:

$$P(h_1|h^-) \propto P_D(h_1|h^-)P_M(h_1|h^-) \quad (9)$$
$$P(h_2|h^-) \propto P_D(h_2|h^-)P_M(h_2|h^-) \quad (10)$$

where $P_D(h|h^-)$ and $P_M(h|h^-)$ are the posterior probabilities in HDP-based model and in HMM-based model, and h^- denotes the current segmentation results for all observed data except $c_{i-2}c_{i-1}c_ic_{i+1}c_{i+2}$. Note that the normalization term $Z(C)$ can be ignored during inference. The posterior probabilities for these two hypotheses in the HDP-based model is given as:

$$\begin{aligned} P_D(h_1|h^-) &= P_D(w_1|w_l, h^-) \\ &\times P_D(w_2|w_1, h^-)P_D(w_r|w_2, h^-) \quad (11) \\ P_D(h_2|h^-) &= P_D(w|w_l, h^-) \\ &\times P_D(w_r|w, h^-) \quad (12) \end{aligned}$$

where $w_l(w_r)$ is the first word to the left (right) of w. And the posterior probabilities for the Bayesian HMM model is given as:

$$\begin{aligned} &P_M(h_1|h^-) \\ &\propto \prod_{j=i-2}^{i+2} P_t(t_j|t_{j-1}, h^-)P_e(c_j|t_j, h^-) \quad (13) \\ &P_M(h_2|h^-) \\ &\propto \prod_{j=i-2}^{i+2} P_t(t_j|t_{j-1}, h^-)P_e(c_j|t_j, h^-) \quad (14) \end{aligned}$$

where $P_t(t_j|t_{j-1}, h^-)$ and $P_e(c_j|t_j, h^-)$ are given in Equation 4 and 5. The difference is that under hypothesis h_1, $c_{i-2}c_{i-1}c_ic_{i+1}c_{i+2}$ are tagged as "BMEBE" and under hypothesis h_2 as "BMMME".

Once the Gibbs sampler is converged, a natural way to is to treat the result of last iteration as the final segmentation result, since each set of assignments to the boundary variables uniquely determines a segmentation.

4 Experiments

In this section, we test our joint model on PKU and MSRA datesets provided by the Second Segmentation Bake-off (SIGHAN 2005) (Emerson, 2005). Most previous works reported their results on these two datasets, this will make it convenient to directly compare our joint model with theirs.

4.1 Setting

The second SIGHAN Bakeoff provides several large-scale labeled data for evaluating the performance of Chinese word segmentation systems. Two of the four datasets are used in our experiments. Both of the dataset contains only simplified Chinese. Table 1 shows the statistics of the two selected corpus. For development set, we randomly select a small subset (about 10%) of the training data. Specifically, 2000 sentences are selected for PKU corpus and 8000 sentences for MSRA corpus. The rest training data plus the test set is then combined for segmentation but only test data is used for evaluation. The development set is used to tune parameters of the HDP-based model and HMM-based model separately. Since our joint model does not involve any additional parameters, we reuse the parameters of the HDP-based model and HMM-based model in the joint model. Specifically, we set $\alpha_1 = 1000.0, \alpha = 10.0, p_s = 0.5$ for the HDP-based model and set $\theta = 1.0, \sigma = 0.01$ for the HMM-based model.

For evaluation, we use standard F-Score on words for all following experiments. F-Score is the harmonic mean of the word precision and recall. Precision is given as:

$$P = \frac{\#correct\ words\ in\ result}{\#total\ words\ in\ result}$$

and recall is given as:

$$R = \frac{\#correct\ words\ in\ result}{\#total\ words\ in\ gold\ corpus}$$

then F-Score is calculated as:

$$F = \frac{2 \times R \times F}{R + F}$$

Corpus	TrainingSize (words)	TestSize (words)
PKU	1.1M	104K
MSRA	2.37M	107K

Table 1: Statistics of training and testing data

Huang and Zhao (2007) provided an empirical method to estimate the consistency between the four different segmentation standards involved in the Bakeoff-3. A lowest consistency rate 84.8% is found among the four standards. Zhao and Kit (2008) considered this figure as the upper bound for any unsupervised Chinese word segmentation systems. We also use it as the **topline** in our comparison.

4.2 Prior Knowledge Used

When it comes to the evaluation and comparison for unsupervised word segmentation systems, an important issue is what kind of pre-processing steps and prior knowledge are needed. To be fully unsupervised, any prior knowledge such as punctuation information, encoding scheme and word length could not be used in principle. Nevertheless, information like punctuation can be easily injected to most existing systems and significantly enhance the performance. The problem we are faced with is that we don't know for sure what kind of prior information are used in other systems. One may use a small punctuation set to segment a long sentence into shorter ones, while another may write simple regular expressions to identify dates and numbers. Lot of work we compare to don't even mention this subject.

Fortunately, we notice that Wang et al. (2011) provided four kinds of preprocessings (they call *settings*). In their settings 1 and 2, punctuation and other encoding information are not used. In setting 3, punctuation is used to segment character sequences into sentences, and both punctuation and other encoding information are used in setting 4. Then the results reported in Magistry and Sagot (2012) relied on setting 3 and setting 4. In order to make the comparison as fair as possible, we use setting 3 in our experiment, i.e., only a punctuation set for simplified Chinese is used in all our experiments. We will compare our experiment results to previous work on the same setting if they are provided .

4.3 Experiment Results

Table 2 summarizes the F-Scores obtained by different models on PKU and MSRA corpus, as well as several state-of-the-art systems. Detailed information about the presented models are listed as follows:

- **nVBE:** the model based on Variation of Branching Entropy in Magistry and Sagot (2012). We re-implement their model on setting 3[1].
- **HDP:** the HDP-based model proposed by Goldwater et al. (2009), initialized randomly.
- **HDP+HMM:** the model combining HDP-based model and HMM-based model as proposed in Section 3, initialized randomly.
- **HDP+nVBE:** the HDP-based model, initialized with the results of nVBE model.
- **Joint:** the "HDP+HMM" model initialized with nVBE model.
- **ESA:** the model proposed in Wang et al. (2011), as mentioned above, the conducted experiments on four different settings, we report their results on setting 3.
- **NPY(2):** the 2-gram language model presented by Mochihashi et al. (2009).
- **NPY(3):** the 3-gram language model presented by Mochihashi et al. (2009).

For all of our Gibbs samplers, we run 5 times to get the averaged F-Scores. We also give the variance of the F-Scores in Table 2. For each run, we find that random initialization takes around 1,000 iterations to converge, while initialing with nVBE only takes as few as 10 iterations. This makes

[1]The results we got with our implementation is slightly lower than what was reported in Magistry and Sagot (2012). According to Pei et al. (2013), they had contacted the authors and confirmed that the higher results was due to a bug in code. So we report the results with our bug free implementation as Pei et al. (2013) did. Our reported results are identical to those of Pei et al. (2013)

System	PKU			MSRA		
	R	**P**	**F**	**R**	**P**	**F**
nVBE	78.3	77.5	77.9	79.1	77.3	78.2
HDP	69.0	68.4	68.7(0.012)	70.4	69.4	69.9(0.020)
HDP+HMM	77.5	73.2	75.3(0.005)	79.9	73.0	76.3(0.013)
HDP+nVBE	80.7	77.9	79.3(0.012)	81.8	77.3	79.5(0.005)
Joint	**83.1**	**79.2**	**81.1**(0.002)	**84.2**	**79.3**	**81.7**(0.005)
ESA	N/A	N/A	77.4	N/A	N/A	78.4
NPY(2)	N/A	N/A	N/A	N/A	N/A	80.2
NPY(3)	N/A	N/A	N/A	N/A	N/A	80.7
Topline	N/A	N/A	84.8	N/A	N/A	84.8

Table 2: Experiment results and comparison to state-of-the-art systems. The figures in parentheses denote the variance the of F-Scores.

our joint model very efficient and possible to work in practical applications as well. At last, a single sample (the last one) is used for evaluation.

From Table 2, we can see that the joint model (Joint) outperforms all the presented systems in F-Score on all testing corpora. Specifically, comparing "HDP+HMM" with "HDP", the former model increases the overall F-Score from 68.7% to 75.3% (+6.6%) in PKU corpora and from 69.9% to 76.3% (+6.4%) in MSRA corpora, which proves that the character information in the HMM-based model can actually enhance the performance of the HDP-based model. Comparing "HDP+nVBE" with "HDP", the former model also increases the overall F-Score by 10.6%/9.6% in PKU/MSRA corpora, which demonstrates that initializing the HDP-based model with nVBE will improve the performance by a large margin. Finally, the joint model "Joint" take advantage from both from the character-based HMM model and the nVBE model, it achieves a F-Score of 81.1% on PKU and 81.7% on MSRA. This result outperforms all its component baselines such as "HDP", "HDP+HMM" and "HDP+nVBE".

Our joint model also shows competitive advantages over several state-of-the-art systems. Compared with nVBE,the F-Score increases by 3.2% on PKU corpora and by 3.5% on MSRA corpora. Compared with ESA, the F-Score increases by 3.7%/3.3% in PKU/MSRA corpora. Lastly, compared to the nonparametric Bayesian models (NPY(n)), our joint model still increases the F-Score by 1.5% (NPY(2)) and 1.0% (NPY(3)) on MSRA corpora. Moreover, compared with the empirical topline figure 84.8%, our joint model achieves a pretty close F-Score. The differences are 3.7% on PKU corpora and 3.1% on MSRA corpora.

An phenomenon we should pay attention to is the poor performance of the HMM-based model. With our implementation of the Bayesian HMM, we achieves a 34.3% F-Score on PKU corpora and a 34.9% F-Score on MSRA corpora, just slightly better than random segmentation. The result show that the hidden Markov Model alone is not suitable for character-based Chinese word segmentation problem. However, it still substantially contributes to the joint model.

We find that the variance of the results are rather small, this shows the stability of our Gibbs samplers. From the segmentation results generated by the joint model, we also found that quite a large amount of errors it made are related to dates, numbers (both Chinese and English) and English words. This problem can be easily addressed during preprocessing by considering encoding information as previous work, and we believe this will bring us much better performance.

4.4 Disambiguation Ability

Previous unsupervised work usually evaluated their models using F-score, regardless of goodness measure based model or nonparametric Bayesian model. However, segmentation ambiguity is a very important factor influencing accuracy of Chinese word segmentation systems (Huang and Zhao, 2007). We believe that the disambiguation ability of the models should also be considered when evaluating different types of unsupervised segmentation systems, since different type of models shows different disambiguation ability. We will compare the disambiguation ability of dif-

ferent systems in this section.

In general, there are mainly two kinds of ambiguity in Chinese word segmentation problem:

- **Combinational Ambiguity**: Given character strings "A" and "B", if "A", "B", "AB" are all in the vocabulary, and "AB" or "A-B" (here "-" denotes a space) occurred in the real text,then "AB" can be called a combinational ambiguous string.
- **Overlapping Ambiguity**: Given character strings "A", "J" and "B", if "A", "B", "AJ" and "JB" are all in the vocabulary, and "A-JB" or "AJ-B" occurred in the real text, then "AJB" can be called an overlapping ambiguous string.

We count the total number of mistakes different systems made at ambiguous strings (the vocabulary is obtained from the gold standard answer of testing set). As we have mentioned in Section 2, goodness measure based methods such as nVBE do not have any disambiguation ability in theory. Our observation is identical to this argument. We find that nVBE always segments ambiguous strings into the same result. Take a combinational string "只有" as an example, "只 (just)", "有 (have)" and "只有 (only)" are all in the vocabulary. In the PKU test set, this string occurs 14 times as "只-有 (just have)" and 18 times as "只有 (only)", 32 times in total. nVBE segments all the 32 strings into "只有 (only)" (i.e. 18 of them are correct), while the joint model segments it 22 times as "只有 (only)" and 10 times as "只-有 (just have)" according to its context, and 24 of them are correct.

Table 3 and 4 show the statistics of combinational ambiguity and overlapping ambiguity respectively. The numbers in parentheses denote the total number of ambiguous strings. From these tables, we can see that HDP+nVBE makes less mistakes than nVBE in most circumstances, except that it solves less combinational ambiguities on MSRA corpora. But our proposed joint model solves the most combinational and overlapping ambiguities, on both PKU and MSRA corpora. Specifically, compared to nVBE, the joint model correctly solves 171/871 more combinational ambiguities on PKU/MSRA corpora, which is a $0.6\%/13.8\%$ relative error reduction. It also solves 28/45 more overlapping ambiguities on PKU/MSRA corpora, which is a $11.5\%/23.4\%$ relative error reduction. This indicates that the joint model has a stronger ability of disambiguation over the compared systems.

System	PKU(35371)	MSRA(38506)
nVBE	8087	7236
HDP+nVBE	7970	7500
Joint	**7916**	**6305**

Table 3: Statistics of combinational ambiguity. This table shows the total number of mistakes made by different systems at combinational ambiguous strings. The numbers in parentheses denote the total number of combinational ambiguous strings.

System	PKU(603)	MSRA(467)
nVBE	244	192
HDP+nVBE	239	164
Joint	**216**	**157**

Table 4: Statistics of overlapping ambiguity. This table shows the total number of mistakes made by different systems at overlapping ambiguous strings. The numbers in parentheses denote the total number of overlapping ambiguous strings.

4.5 Statistical Significance Test

The main results presented in Table 2 has shown that our proposed joint model outperforms the two baselines as well as state-of-the-art systems. But it is also important to know if the improvement is statistically significant over these systems. So we conduct statistical significance tests of F-scores among these various models. Following Wang et al. (2010), we use the bootstrapping method (Zhang et al., 2004).

Here is how it works: suppose we have a testing set T_0 to test several word segmentation systems, there are N testing examples (sentences or line of characters) in T_0. We create a new testing set T_1 with N examples by sampling with replacement from T_0, then repeat these process $M-1$ times. And we will have a total $M+1$ testing sets. In our test procedures, M is set to 2000.

Since we just implement our joint model and its component models, we can not generate paired samples for other models (i.e. ESA and NPY(n)). Instead, we follow Wang et al. (2010)'s method and first calculate the 95% confidence interval for

our proposed model. Then other systems can be compared with the joint model in this way: if the F-score of system **B** doesn't fall into the 95% confidence interval of system **A**, they are considered as statistically significantly different from each other.

For all significant tests, we measure the 95% confidence interval for the difference between two models. First, the test results show that "HDP+nVBE" and "HDP+HMM" are both significantly better than "HDP". Second, the "Joint" model significantly outperforms all its component models, including "HDP", "nVBE", "HDP+nVBE" and "HDP+HMM". Finally, the comparison also shows that the joint model significantly outperforms state-of-the-art systems like ESA and NPY(n).

5 Conclusion

In this paper, we proposed a joint model for unsupervised Chinese word segmentation. Our joint model is a combination of the HDP-based model, which is a word-based model, and HMM-based model, which is a character-based model. The way we combined these two component baselines makes it natural and simple to inference with Gibbs sampling. Then the joint model take advantage of a goodness-based method (nVBE) by using it to initialize the sampler. Experiment results conducted on PKU and MSRA datasets provided by the second SIGHAN Bakeoff show that the proposed joint model not only outperforms the baseline systems but also achieves better performance (F-Score) over several state-of-the-art systems. Significance tests showed that the improvement is statistically significant. Analysis also indicates that the joint model has a stronger ability to solve ambiguities in Chinese word segmentation. In summary, the joint model we proposed combines the strengths of character-based model, nonparametric Bayesian language model and goodness-based model.

Acknowledgments

The contact author of this paper, according to the meaning given to this role by Key Laboratory of Computational Linguistics, Ministry of Education, School of Electronics Engineering and Computer Science, Peking University, is Baobao Chang. And this work is supported by National Natural Science Foundation of China under Grant No. 61273318 and National Key Basic Research Program of China 2014CB340504.

References

George Casella, Edward I. George. 1992. Explaining the Gibbs sampler. *The American Statistician*, 46(3): 167-174.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, 133. MLA.

Haodi Feng, Kang Chen, Xiaotie Deng, et al. 2004. Accessor variety criteria for Chinese word extraction *Computational Linguistics*, 30(1): 75-93.

Sharon Goldwater, Thomas L. Griffiths, Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition* 112(1): 21-54.

Geoffrey E. Hinton. 1999. Products of experts. *Artificial Neural Networks*. Ninth International Conference on Vol. 1.

Changning Huang, Hai Zhao. 2007. Chinese word segmentation: A decade review. *Journal of Chinese Information Processing*, 21(3): 8-20.

Zhihui Jin, Kumiko Tanaka-Ishii. 2006. Unsupervised segmentation of Chinese text by use of branching entropy. *Proceedings of the COLING/ACL on Main conference poster sessions*, page 428-435.

Chunyu Kit, Yorick Wilks. 1999. Unsupervised learning of word boundary with description length gain. *Proceedings of the CoNLL99 ACL Workshop. Bergen, Norway: Association for Computational Linguis-tics*, page 1-6.

Pierre Magistry, Benoit Sagot. 2012. Unsupervized word segmentation: the case for mandarin chinese. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2. Association for Computational Linguistics*, page 383-387.

Daichi Mochihashi, Takeshi Yamada, Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1. Association for Computational Linguistics*, page 100-108.

Wenzhe Pei, Dongxu Han, Baobao Chang. 2013. A Refined HDP-Based Model for Unsupervised Chinese Word Segmentation. *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data. Springer Berlin Heidelberg*, page 44-51.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, et al. 2006. Sharing Clusters among Related Groups: Hierarchical Dirichlet Processes. *NIPS*.

Fuchun Peng, Fangfang Feng, Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. *Proceedings of COLING*, page 562-568.

Huihsin Tseng, Pichuan Chang, Galen Andrew, et al. 2005. A conditional random field word segmenter for sighan bakeoff 2005. *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, Vol. 171.

Kun Wang, Chengqing Zong, Keh-Yih Su. 2010. A character-based joint model for Chinese word segmentation. *Proceedings of the 23rd International Conference on Computational Linguistics. Association for Computational Linguistics*, page 1173-1181.

Hanshi Wang, Jian Zhu, Shiping Tang, et al. 2011. A new unsupervised approach to word segmentation. *Computational Linguistics*, 37(3): 421-454.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1): 29-48.

Hai Zhao, Chunyu Kit. 2008. An Empirical Comparison of Goodness Measures for Unsupervised Chinese Word Segmentation with a Unified Framework. *IJCNLP*, page 6-16.

Ying Zhang, Stephan Vogel, Alex Waibel. 2004. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system? *LREC*.

Domain Adaptation for CRF-based Chinese Word Segmentation using Free Annotations

Yijia Liu †‡, Yue Zhang †, Wanxiang Che ‡, Ting Liu ‡, Fan Wu †
†Singapore University of Technology and Design
‡Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{yjliu,car,tliu}@ir.hit.edu.cn {yue_zhang,fan_wu}@sutd.edu.sg

Abstract

Supervised methods have been the dominant approach for Chinese word segmentation. The performance can drop significantly when the test domain is different from the training domain. In this paper, we study the problem of obtaining partial annotation from freely available data to help Chinese word segmentation on different domains. Different sources of free annotations are transformed into a unified form of partial annotation and a variant CRF model is used to leverage both fully and partially annotated data consistently. Experimental results show that the Chinese word segmentation model benefits from free partially annotated data. On the SIGHAN Bakeoff 2010 data, we achieve results that are competitive to the best reported in the literature.

1 Introduction

Statistical Chinese word segmentation gains high accuracies on newswire (Xue and Shen, 2003; Zhang and Clark, 2007; Jiang et al., 2009; Zhao et al., 2010; Sun and Xu, 2011). However, manually annotated training data mostly come from the news domain, and the performance can drop severely when the test data shift from newswire to blogs, computer forums and Internet literature (Liu and Zhang, 2012).

Several methods have been proposed for solving the domain adaptation problem for segmentation, which include the traditional token- and type-supervised methods (Song et al., 2012; Zhang et al., 2014). While token-supervised methods rely on manually annotated target-domain sentences, type-supervised methods leverage manually assembled domain-specific lexicons to improve target-domain segmentation accuracies. Both methods are competitive given the same amount of annotation effects (Garrette and Baldridge, 2012; Zhang et al., 2014). However, obtaining manually annotated data can be expensive.

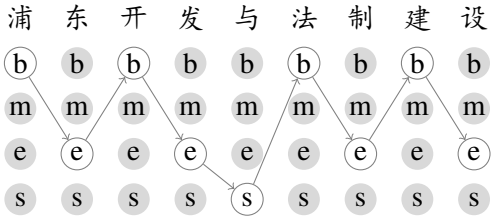


Figure 1: The segmentation problem, illustrated using the sentence "浦东 (Pudong) 开发 (development) 与 (and) 法制 (legal) 建设 (construction)". Possible segmentation labels are drawn under each character, where *b, m, e, s* stand for the beginning, middle, end of a multi-character word, and a single character word, respectively. The path shows the correct segmentation by choosing one label for each character.

On the other hand, there are *free* data which contain limited but useful segmentation information over the Internet, including large-scale unlabeled data, domain-specific lexicons and semi-annotated web pages such as Wikipedia. In the last case, word-boundary information is contained in hyperlinks and other markup annotations. Such free data offer a useful alternative for improving the segmentation performance, especially on domains that are not identical to newswire, and for which little annotation is available.

In this paper, we investigate techniques for adopting freely available data to help improve the performance on Chinese word segmentation. We propose a simple but robust method for constructing partial segmentation from different sources of free data, including unlabeled data and the Wikipedia. There has been work on making use of both unlabeled data (Sun and Xu, 2011; Wang et al., 2011) and Wikipedia (Jiang et al., 2013)

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 864–874,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

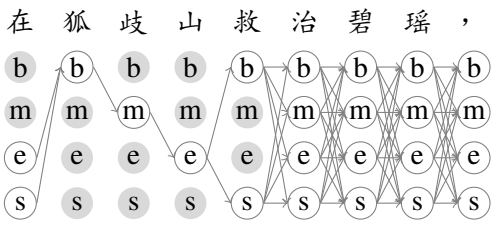
to improve segmentation. However, no empirical results have been reported on a unified approach to deal with different types of free data. We use a conditional random fields (Lafferty et al., 2001; Tsuboi et al., 2008) variant that can leverage the partial annotations obtained from different sources of free annotation. Training is achieved by a modification to the learning objective, incorporating partial annotation likelihood, so that a single model can be trained consistently with a mixture of full and partial annotation.

Experimental results show that our method of using partially annotated data can consistently improves cross-domain segmentation performance. We obtain results which are competitive to the best reported in the literature. Our segmentor is freely released at `https://github.com/ExpResults/partial-crfsuite`.

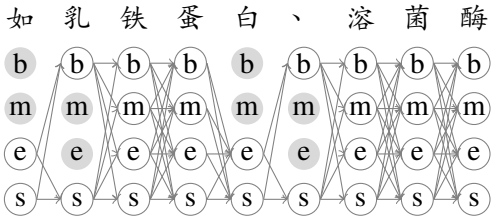
2 Obtaining Partially Annotated Data

We model the Chinese word segmentation task as a character sequence tagging problem, which is to give each character in a sentence a word-boundary tag (Xue and Shen, 2003). We adopt four tags, *b, m, e* and *s*, which represent the *b*eginning, *m*iddle, *e*nd of a multi-character word, and a *s*ingle character word, respectively. A manually segmented sentence can be represented as a tag sequence, as shown in Figure 1.

We investigate two major sources of freely-available annotations: lexicons and natural annotation, both with the help of unannotated data. To make use of the first source of information, we incorporate words from a lexicon into unannotated sentences by matching of character sequences, resulting in partially annotated sentences, as shown in Figure 2a. In this example, the word "狐岐山 (the Huqi Mountain)" in the unannotated sentence matches an item in the lexicon. As a result, we obtain a partially-annotated sentence, in which the segmentation ambiguity of the characters "狐 (fox)", "岐 (brandy road)" and "山 (mountain)" are resolved ("狐" being the beginning, "岐" being the middle and "山" being the end of the same word). At the same time, the segmentation ambiguity of the surrounding characters "在 (at)" and "救 (save)" are reduced ("在" being either a single-character word or the end of a multi-character word, and "救" being either a single-character word or the beginning of a multi-character word).



(a) "在 (at) 狐岐山 (Huqi Mountain) 救 治 (save) 碧 瑶 (Biyao)", where "狐岐山" matches a lexicon word.



(b) "如 (e.g.) 乳铁蛋白 (lysozyme) 、 溶菌酶 (lactoferrin)", where "乳铁蛋白" is a hyperlink.

Figure 2: Examples of partially annotated data. The paths show possible correct segmentations.

Natural annotation, which refers to word boundaries that can be inferred from URLs, fonts or colors on web pages, also result in partially-annotated sentences. Taking a web page shown in Figure 2b for example. It can be inferred from the URL tags on "乳铁蛋白" that "乳" should be either the beginning of a multi-character word or a single-character word, and "白" should be either the end a multi-character word or single-character word. Similarly, possible tags of the surrounding character "如" and "、" can also be inferred.

We turn both lexicons and natural annotation into the same form of *partial annotation* with same unresolved ambiguities, as shown in Figure 2, and use them together with available *full annotation* (Figure 1) as the training data for the segmentor. In this section, we describe in detail how to obtain partially annotated sentences from each resource, respectively.

2.1 Lexicons

In this scenario, we assume that there are unlabeled sentences along with a lexicon for the target domain. We obtain partially segmented sentences by extracting word boundaries from the unlabeled sentences with the help of the lexicon. Previous matching methods (Wu and Tseng, 1993; Wong and Chan, 1996) for Chinese word segmentation largely rely on the lexicons, and are generally considered being weak in ambiguity resolution (Gao

People's Daily	看到 (saw) 海南 (Hainan) 旅游业 (tourist industry) 充满 (full) 希望 (hope) saw tourist industry in Hainan is full of hope
Wikipedia	主要(mainly) 是(is) 旅游 (tourist) 业 (industry) 和(and) 软件 (software) 产业(industry) mainly is tourist industry and software industry

(a) Case of incompatible annotation on "旅游业(tourist industry)" between People's Daily and Wikipedia.

Literature	《说文解字 (Shuo Wen Jie Zi, a book) 段(segmented) 注(annotated) 》 the segmented and annotated version of Shuo Wen Jie Zi
Computer	每条(each) 记录(record) 被(is) 分隔(splitted) 为(into) 字段 (fields) each record is splitted into several fields

(b) Similar subsequence "字段(field)" is segmented differently under different domains in Wikipedia.

Table 1: Examples natural annotation from Wikipedia. Underline marks annotated words.

et al., 2005). But for obtaining the partial labeled data with lexicon, the matching method can still be a solution. Since we do not aim to recognize every word from sentence, we can select a lexicon with smaller coverage but less ambiguity to achieve relatively precise matching result.

In this paper, we apply two matching schemes to the same raw sentences to obtain partially annotated sentences. The first is a simple forward-maximum matching (FMM) scheme, which is very close to the forward maximum matching algorithm of Wu and Tseng (1993) for Chinese word segmentation. This scheme scans the input sentence from left to right. At each position, it attempts to find the longest subsequence of Chinese characters that matches a lexicon entry. If such an entry is found, the subsequence is tagged with the corresponding tags, and its surrounding characters are also constrained to a smaller set of tags. If no subsequence is found in the lexicon, the character is left with all the possible tags. Taking the sentence in Figure 2a for example. When the algorithm scans the second character, "狐", and finds the entry "狐岐山" in the lexicon, the subsequence of characters is recognized as a word, and tagged with *b, m* and *e*, respectively. At the same time, the previous character "在" can be inferred as only end of a multi-character word (*e*) or a single-character word (*s*). The second matching scheme is backward maximum matching, which can be treated as the application of FMM on the reverse of unlabeled sentences using a lexicon of reversed words.

To mitigate the errors resulting from one single matching scheme, we combine the two matching results by agreement. The basic idea is that if a subsequence of sentence is recognized as word by multiple matching results, it can be considered as a more precise annotation. Our algorithm reads partial segmentation by different methods and selects the subsequences that are identified as word by all methods as annotated words.

2.2 Natural Annotation

We use the Chinese Wikipedia for natural annotation. Partially annotated sentences are readily formed in Wikipedia by markup syntax, such as URLs. However, some subtle issues exist if the sentences are used directly. One problem is incompatibility of segmentation standards between the annotated training data and Wikipedia. Jiang et al. (2009) discuss this incompatibility problem between two corpora — the CTB and the People's Daily; the problem is even more severe on Wikipedia because it can be edited by any user. Table 1a shows a case of incompatible annotation between the People's Daily data and natural annotation in Wikipedia, where the three characters "旅游业" are segmented differently. Both can be treated as correct, although they have different segmentation granularities.

Another problem is the intrinsic ambiguity of segmentation. The same character sequence can be segmented into different words under different contexts. If the training and test data contain different contexts, the learned model can give incorrect results on the test data. This is particularly true across different domains. Table 1b gives such an example, where the character sequence "字段" is segmented differently in two of our test domains, but both cases exist in Wikipedia.

In summary, Wikipedia introduces both useful information for domain adaptation and harmful noise with negative effects on the model. To

achieve better performance of domain adaptation using Wikipedia, one intuitive approach is to select more domain-related data and less irrelevant data to minimize the risks that result from incompatible annotation and domain difference.

To this end, we assume that there are some raw sentences on the target domain, which can be used to evaluate the relevance between Wikipedia and target domain test data. We assume that URL-tagged entries reflect the segmentation standards of Wikipedia sentence, and use them to match Wikipedia sentences with the raw target domain data. If the character sequence of any URL-tagged entry in a Wikipedia sentence matches the target domain data, the Wikipedia sentence is selected for training. Another advantage of such data selection is that the training time consumption can be reduced by reducing the size of training data.

3 CRF for Word Segmentation

We follow the work of Zhao et al. (2010) and Sun and Xu (2011), and adopt the Conditional Random Fields (CRF) model (Lafferty et al., 2001) for the sequence labeling problem of word segmentation. Given an input characters sequence, the task is to assign one segmentation label from $\{b, m, e, s\}$ on each character. Let $\mathbf{x} = (x_1, x_2, ..., x_T)$ be the sequence of characters in sentence whose length is T, and $\mathbf{y} = (y_1, y_2, ..., y_T)$ be the corresponding label sequence, where $y_i \in Y$. The linear-chain conditional random field for Chinese word segmentation can be formalized as

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \exp \sum_{t=1}^{T} \sum_{k} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}) \quad (1)$$

where λ_k are the model parameters, f_k are the feature functions and Z is the probability normalizer.

$$Z = \sum_{\mathbf{y}} \exp \sum_{t=1}^{T} \sum_{k} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}) \quad (2)$$

We follow Sun and Xu (2011) and use the feature templates shown in Table 2 to model the segmented task. For ith character in the sentence, the n-gram features represent the surrounding characters of this character; $Type$ categorizes the character it into $digit$, $punctuation$, $english$ and $other$; $Identical$ indicates whether the input character is the same with its surrounding characters. This feature captures repetition patterns such as "试试 (try)" or "走走 (stroll)".

Type	Template
unigram	C_s $(i-3 < s < i+3)$
bigram	C_sC_{s+1} $(i-3 < s < i+2)$
	C_sC_{s+2} $(i-3 < s < i+1)$
type	$Type(C_i)$
	$Type(C_s)Type(C_{s+1})$ $(i-1 < s < i+2)$
identical	$Identical(C_s, C_{s+1})$ $(i-3 < s < i+1)$
	$Identical(C_s, C_{s+2})$ $(i-3 < s < i)$

Table 2: Feature templates for the ith character.

For fully-annotated training data, the learning problem of conditional random fields is to maximize the log likelihood over all the training data (Lafferty et al., 2001)

$$\mathcal{L} = \sum_{n=1}^{N} \log p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)})$$

Here N is the number of training sentences. Both the likelihood $p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)})$ and its gradient can be calculated by performing the forward-backward algorithm (Baum and Petrie, 1966) on the sequence, and several optimization algorithm can be adopted to learn parameters from data, including L-BFGS (Liu and Nocedal, 1989) and SGD (Bottou, 1991).

4 Training a CRF with partially annotated data

For word segmentation with partially annotated data, some characters in a sentence can have a definite segmentation label, while some can have multiple labels with ambiguities remaining. Taking the partially annotated sentence in Figure 2a for example, the corresponding potential label sequence for "在狐岐山救" is $\{(e, s), (b), (m), (e), (b, s)\}$, where the characters "狐", "岐" and "山" have fixed labels but for "在" and "救", some ambiguities exist. Note that the full annotation in Figure 1 can be regarded as a special case of partial annotation, where the number of potential labels for each character is one.

We follow Tsuboi et al. (2008) and model marginal probabilities over partially annotated data. Define the possible labels that correspond to the partial annotation as $\mathbf{L} = (L_1, L_2, ..., L_T)$, where each L_i is a non-empty subset of Y that corresponds to the set of possible labels for x_i. Let

$\mathbf{Y_L}$ be the set of all possible label sequences where $\forall \mathbf{y} \in \mathbf{Y_L}$, $y_i \in L_i$. The marginal probability of $\mathbf{Y_L}$ can be modeled as

$$p(\mathbf{Y_L}|\mathbf{x}) = \frac{1}{Z} \sum_{\mathbf{y} \in \mathbf{Y_L}} \exp \sum_{t=1}^{T} \sum_{k} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}) \quad (3)$$

Defining the unnormalized marginal probability as

$$Z_{\mathbf{Y_L}} = \sum_{\mathbf{y} \in \mathbf{Y_L}} \exp \sum_{t=1}^{T} \sum_{k} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}),$$

and the normalizer Z being the same as Equation 2, the log marginal probability of $\mathbf{Y_L}$ over N partially annotated training examples can be formalized as

$$\mathcal{L}_{\mathbf{Y_L}} = \sum_{n=1}^{N} \log p(\mathbf{Y_L}|\mathbf{x}) = \sum_{n=1}^{N} (\log Z_{\mathbf{Y_L}} - \log Z)$$

The gradient of the likelihood can be written as

$$\frac{\partial \mathcal{L}_{\mathbf{Y_L}}}{\partial \lambda_k} = \sum_{n=1}^{N} \sum_{t=1}^{T} \sum_{\substack{y_{\mathbf{Y_L}} \in L_t, \\ y'_{\mathbf{Y_L}} \in L_{t-1}}} f_k(y_{\mathbf{Y_L}}, y'_{\mathbf{Y_L}}, \mathbf{x}) p_{\mathbf{Y_L}}(y_{\mathbf{Y_L}}, y'_{\mathbf{Y_L}}|\mathbf{x}) - \sum_{n=1}^{N} \sum_{t=1}^{T} \sum_{y,y'} f_k(y, y', \mathbf{x}) p(y, y'|\mathbf{x})$$

Both $Z_{\mathbf{Y_L}}$ and its gradient are similar in form to Z. By introducing a modification to the forward-backward algorithm, $Z_{\mathbf{Y_L}}$ and $\mathcal{L}_{\mathbf{Y_L}}$ can be calculated. Define the *forward variable* for partially annotated data $\alpha_{\mathbf{Y_L},t}(j) = p_{\mathbf{Y_L}}(x_{\langle 1,\ldots,t \rangle}, y_t = j)$. A modification on the forward algorithm can be formalized as

$$\alpha_{\mathbf{Y_L},t}(j) = \begin{cases} 0 & j \notin L_t \\ \sum_{i \in L_{t-1}} \Psi_t(j, i, x_t) \alpha_{\mathbf{Y_L},t-1}(i) & j \in L_t \end{cases}$$

where $\Psi_t(j, i, x)$ is a potential function that equals $\sum_k \lambda_k f_k(y_t = j, y_{t-1} = i, x_t)$. Similarly, for the *backward variable* $\beta_{\mathbf{Y_L},t}$,

$$\beta_{\mathbf{Y_L},t}(i) = \begin{cases} 0 & i \notin L_t \\ \sum_{j \in L_{t+1}} \Psi_t(j, i, x_{t+1}) \beta_{\mathbf{Y_L},t+1}(j) & i \in L_t \end{cases}$$

$Z_{\mathbf{Y_L}}$ can be calculated by $\alpha_{\mathbf{Y_L}}(T)$, and $p_{\mathbf{Y_L}}(y, y'|x)$ can be calculated by $\alpha_{\mathbf{Y_L},t-1}(y') \Psi_t(y, y', x_t) \beta_{\mathbf{Y_L},t}(y)$.

Note that if each element in $\mathbf{Y_L}$ is constrained to one single label, the CRF model in Equation 3 degrades into Equation 1. So we can train a unified model with both fully and partially annotated data. We implement this CRF model based on a open source toolkit CRFSuite.[1] In our experiments, we use the L-BFGS (Liu and Nocedal, 1989) algorithm to learn parameters from both fully and partially annotated data.

5 Experiments

We perform our experiments on the domain adaptation test data from SIGHAN Bakeoff 2010 (Zhao et al., 2010), adapting annotated training sentences from People's Daily (PD) (Yu et al., 2001) to different test domains. The fully annotated data is selected from the People's Daily newspaper in January of 1998, and the four test domains from the SIGHAN Bakeoff 2010 include finance, medicine, literature and computer. Sample segmented data in the computer domain from this bakeoff is used as development set. Statistics of the data are shown in first half of Table 3. We use wikidump20140419[2] for the Wikipedia data. All the traditional Chinese pages in Wikipedia are converted to simplified Chinese. After filtering functional pages like *redirection* and removing duplication, 5.45 million sentences are reserved.

For comparison with related work on using a lexicon to improve segmentation, another set of test data is chosen for this setting. We use the Chinese Treebank (CTB) as the source domain data, and Zhuxian (a free Internet novel, also named as "Jade dynasty", referred to as ZX henceforth) as the target domain data.[3] The ZX data are written in a different style from newswire, and contains many out-of-vocabulary words. This setting has been used by Liu and Zhang (2012) and Zhang et al. (2014) for domain adaptation of segmentation and POS-tagging. We use the standard training, development and test split. Statistics of the test data annotated by Zhang et al. (2014) are shown in the second half of Table 3.

The data preparation method in Section 2 and the CRF method in Section 4 are used for all the experiments. Both recall of out-of-vocabulary words (R_{oov}) and F-score are used to evaluate the

[1] http://www.chokkan.org/software/crfsuite/

[2] http://dumps.wikimedia.org/zhwiki/20140419/

[3] Annotated target domain test data and lexicon are available from http://ir.hit.edu.cn/~mszhang/eacl14mszhang.zip.

	Data set	Train	Development	Test			
PD → SIGHAN		PD	Computer	Finance	Medicine	Literature	Computer
	# sent.	19,056	1,000	560	1,308	670	1,329
	# words	1,109,734	21,398	33,035	31,499	35,735	35,319
	OOV		0.1766	0.0874	0.1102	0.0619	0.1522

	Data set	Train	Development	Test	Unlabeled		Unlabeled
CTB5 → ZX		CTB5	ZX			Wikipedia	
	# sent.	18,086	788	1,394	32,023		5,456,151
	# words	493,934	20,393	34,355			
	OOV		0.1377	0.1550			

Table 3: Statistics of data used in this paper.

segmentation performance. There is a mixture of Chinese characters, English words and numeric expression in the test data from SIGHAN Bakeoff 2010. To test the influence of Wikipedia data on Chinese word segmentation alone, we apply regular expressions to detect English words and numeric expressions, so that they are marked as *not segmented*. After performing this preprocessing step, cleaned test input data are fed to the CRF model to give a relatively strong baseline.

5.1 Free Lexicons

5.1.1 Obtaining lexicons

For domain adaption from CTB to ZX, we use a lexicon released by Zhang et al. (2014). The lexicon is crawled from a online encyclopedia[4], and contains the names of 159 characters and artifacts in the Zhuxian novel. We follow Zhang et al. (2014) and name it **NR** for convenience of further discussion. The NR lexicon can be treated as a strongly domain-related, high quality but relatively small lexicon. It's a typical example of freely available lexicon over the Internet.

For domain adaptation from PD to medicine and computer, we collect a list of page titles under the corresponding categories in Wikipedia. For medicine, entries under *essential medicines*, *biological system* and *diseases* are collected. For computer, entries under *computer network*, *Microsoft Windows* and *software widgets* are selected. These lexicons are typical freely available lexicons that we can access to.

5.1.2 Obtaining Unlabeled Sentences

For ZX, partially annotated sentences are obtained using the *NR* lexicon and unlabeled ZX sentences by applying the matching scheme described in Section 2. The CTB5 training data and the partially annotated data are mixed as the final training data. Different amounts of unlabeled data are applied to the development test set, and results are shown in Figure 3. From this figure we can see that incorporating 16K sentences gives the highest accuracy, and adding more partial labeled data does not change the accuracy significantly. So for the ZX experiments, we choose the 16K sentences as the unlabeled data.

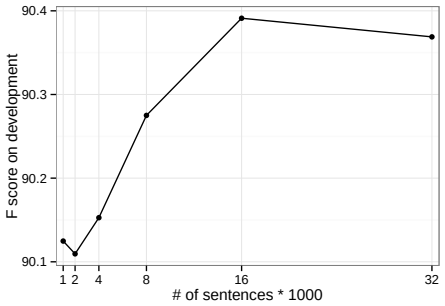


Figure 3: F-score on the development data when using different numbers of unlabeled data.

For the medicine and computer experiments, we selected domain-specific sentences by matching with the domain-specific lexicons. About 46K out of the 5.45 million wiki sentences contain subsequences in the medicine lexicon and 22K in the case of the computer domain. We randomly select 16K sentences as the unlabeled data for each domain, respectively.

5.1.3 Final results

We incorporate the partially annotated data obtained with the help of lexicon for each of the test domain. For adaptation from CTB to ZX, we trained our baseline model on the CTB5 training data with the feature templates in Table 2. For adaptation from PD to medicine and computer, we

[4] http://baike.baidu.com/view/18277.htm

Domain	ZX		Medicine		Computer	
	F	Roov	F	Roov	F	Roov
Baseline	87.50	73.65	91.36	72.95	93.16	84.02
Baseline+Lexicon Feature	90.36	80.69	91.60	74.39	93.14	84.27
Baseline+PA (Lex)	**90.63**	**84.88**	**91.68**	**74.99**	**93.47**	**85.63**
Zhang et al. (2014)	88.34	-	-	-	-	-

Table 4: Final result for adapting CTB to Zhuxian and adapting PD to the medicine and computer domains, using partially annotated data (referred to as *PA*) obtained from unlabeled data and lexicons.

trained our baseline model on the PD training data with the same feature template setting.

Previous research makes use of a lexicon by adding lexicon features directly into a model (Sun and Xu, 2011; Zhang et al., 2014), rather than transforming them into partially annotated sentences. To make a comparison, we follow Sun and Xu (2011) and add three lexicon features to represent whether c_i is located at the beginning, middle or the end of a word in the lexicon, respectively. For each test domain, the lexicon for the lexicon feature model consists of the most frequent words in the source domain training data (about 6.7K for CTB5 and 8K for PD, respectively) and the domain-specific lexicon we obtained in Section 5.1.1.

The results are shown in Table 4, where the first row shows the performance of the baseline models and the second row shows the performance of the model incorporating lexicon feature. The third row shows our method using partial annotation. On the ZX test set, our method outperforms the baseline by more than 3 absolute percentage. The model with partially annotated data performs better than the one with additional lexicon features. Similar conclusion is obtained when adapting from PD to medicine and computer. By incorporating the partially annotated data, the segmentation of lexicon words, along with the context, is learned.

We also compare our method with the work of Zhang et al. (2014), who reported results only on the ZX test data. We use the same lexicon settings. Our method gives better result than Zhang et al. (2014), showing that the combination of a lexicon and unannotated sentence into partially annotated data can lead to better performance than using a dictionary alone in type-supervision. Given that we only explore the use of free resource, combining a lexicon with *unannotated* sentences is a better option than using the lexicon directly. Zhang et al.'s concern, on the other hand, is to compare type- and token-annotation. Our partial annotation can thus be treated as a compromise to obtain some *pseudo* partial token-annotations when *full* token annotations are unavailable. Another thing to note is that the model of Zhang et al. (2014) is a joint model for segmentation and POS-tagging, which is generally considered stronger than a single segmentation model.

Method	Com. Dev	
	F	Roov
Baseline	93.56	83.75
Baseline+PA (Random 160K)	94.29	86.58
Baseline+PA (Selected)	**95.00**	**88.28**

Table 5: The performance of data selection on the development set of the computer domain.

5.2 Free Natural Annotation

When extracting word boundaries from Wikipedia sentences, we ignore natural annotations on English words and digits because these words are recognized by the preprocessor. Following Jiang et al. (2013), we also recognize a naturally annotated two-character subsequence as a word.

5.2.1 Effect of data selection

To make better use of more domain-specific data, and to alleviate noise in partial annotation, we apply the selection method proposed in Section 2 to the Wikipedia data. On the computer domain development test data, this selection method results in 9.4K computer-related sentences with partial annotation. A model is trained with both the PD training data and the partially annotated computer domain Wikipedia data. For comparison, we also trained a model with 160K randomly selected Wikipedia sentences. The experimental result is shown in Table 5. The model incorporating selected data achieves better performance compared to the model with randomly sampled data, demonstrating that data selection is helpful to improving

Method	Finance		Medicine		Literature		Computer		
	F	Roov	F	Roov	F	Roov	F	Roov	Avg-F
Baseline	95.20	86.90	91.36	72.90	92.27	73.61	93.16	83.48	93.00
Baseline+PA (Random 160K)	95.16	87.60	92.41	78.13	92.17	75.30	93.91	83.48	93.41
Baseline+PA (Selected)	**95.54** +0.34	**88.53**	92.47 +1.11	**78.28**	92.49 +0.22	**76.84**	**93.93** +0.77	**87.53**	**93.61**
Jiang et al. (2013)	93.16		**93.34**		**93.53**		91.19		92.80

Table 6: Experimental results on the SIGHAN Bakeoff 2010 data.

the domain adaption accuracy.

5.2.2 Final Result

The final results on the four test domains are shown in Table 6. From this table, we can see that significant improvements are achieved with the help of the partially annotated Wikipedia data, when compared to the baseline. The models trained with selected partial annotation perform better than those trained with random partial annotation. Our F-scores are competitive to those reported by Jiang et al. (2013). However, since their model is trained on a different source domain, the results are not directly comparable.

5.2.3 Analysis

In this section, we study the effect of Wikipedia on domain adaptation when no data selection is performed, in order to analyze the effect of partially annotated data. We randomly sample 10K, 20K, 40K, 80K and 160K sentences from the 5.45 million Wikipedia sentences, and incorporate them into the training process, respectively. Five models are obtained adding the baseline, and we test their performances on the four test domains. Figure 4 shows the results.

From the figure we can see that for the medicine and computer domains, where the OOV rate is relatively high, the F-score generally increases when more data from Wikipedia are used. The trends of F-score and OOV recall against the volume of Wikipedia data are almost identical. However, for the finance and literature domains, which have low OOV rates, such a relation between data size and accuracy is not witnessed. For the literature domain, even an opposite trends is shown.

We can draw the following conclusions: (1) Natural annotation on Wikipedia data contributes to the recognition of OOV words on domain adaptation; (2) target domains with more OOV words benefit more from Wikipedia data. (3) along with the positive effect on OOV recognition, Wikipedia data can also introduce noise, and hence data selection can be useful.

Method	Med.	Com.
	F	F
Baseline	91.36	93.16
Baseline+PA (Lex)	91.68	93.47
Baseline+PA (Natural)	92.47	93.93
Baseline+PA (Lex+Natural)	**92.63**	**94.07**

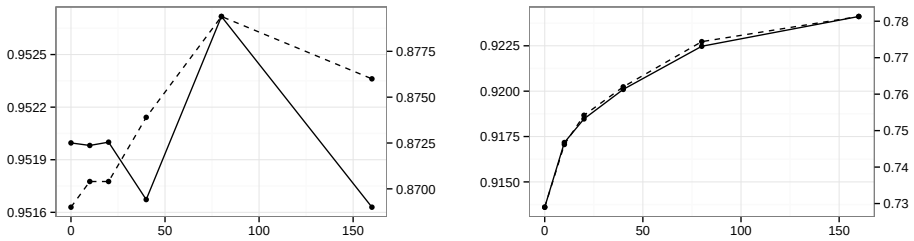
Table 7: Results by combining different sources of free annotation.

5.3 Combining Lexicon and Natural Annotation

To make the most use of free annotation, we combine available free lexicon and natural annotation resources by joining the partially annotated sentences derived using each resource, training our CRF model with these partially annotated sentences and the fully annotated PD sentences. The tests are performed on medicine and computer domains. Table 7 shows the results, where further improvements are made on both domains when the two types of resources are combined.

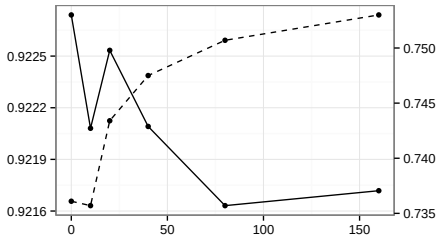
6 Related Work

There has been a line of research on making use of unlabeled data for word segmentation. Zhao and Kit (2008) improve segmentation performance by mutual information between characters, collected from large unlabeled data; Li and Sun (2009) use punctuation information in a large raw corpus to learn a segmentation model, and achieve better recognition of OOV words; Sun and Xu (2011) explore several statistical features derived from unlabeled data to help improve character-based word segmentation. These investigations mainly focus on in-domain accuracies. Liu and Zhang (2012)

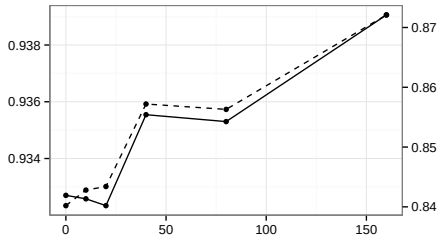


(a) Finance

(b) Medicine



(c) Literature



(d) Computer

Figure 4: Performance of the model incorporating difference sizes of Wikipedia data. The solid line represents the F-score and dashed line represents the recall of OOV words.

study domain adaptation using an unsupervised self-training method. In contrast to their work, we make use of not only unlabeled data, but also leverage any free annotation to achieve better results for domain adaptation.

There has also been work on making use of a dictionary and natural annotation for segmentation. Zhang et al. (2014) study type-supervised domain adaptation for Chinese segmentation. They categorize domain difference into two types: different vocabulary and different POS distributions. While the first type of difference can be effectively resolved by using lexicon for each domain, the second type of difference needs to be resolved by using annotated sentences. They found that given the same manual annotation time, a combination of the lexicon and sentence is the most effective. Jiang et al. (2013) use 160K Wikipedia sentences to improves segmentation accuracies on several domains. Both Zhang et al. (2014) and Jiang et al. (2013) work on discriminative models using the structure perceptron (Collins, 2002), although they study two different sources of information. In contrast to their work, we unify both types of information under the CRF framework.

CRF has been used for Chinese word segmentation (Tseng, 2005; Shi and Wang, 2007; Zhao and Kit, 2008; Wang et al., 2011). However, most previous work train a CRF by using full annotation only. In contrast, we study CRF based segmentation by using both full and partial annotation.

Several other variants of CRF model has been proposed in the machine learning literature, such as the generalized expectation method (Mann and McCallum, 2008), which introduce knowledge by incorporating a manually annotated feature distribution into the regularizer, and the JESS-CM (Suzuki and Isozaki, 2008), which use a EM-like method to iteratively optimize the parameter on both the annotated data and unlabeled data. In contrast, we directly incorporate the likelihood of partial annotation into the objective function. The work that is the most similar to ours is Tsuboi et al. (2008), who modify the CRF learning objective for partial data. They focus on Japanese lexical analysis using manually collected partial data, while we investigate the effect of partial annotation from freely available sources for Chinese segmentation.

7 Conclusion

In this paper, we investigated the problem of domain adaptation for word segmentation, by transferring various sources of free annotations into a consistent form of partially annotated data and applying a variant of CRF that can be trained using fully- and partially-annotated data simultaneously. We performed a large set of experiments to study the effectness of free data, finding that they are useful for improving segmentation accuracy. Experiments also show that proper data selection can further benefit the model's performance.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work was supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61133012 and 61370164, the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301 and SRG ISTD 2012 038 from Singapore University of Technology and Design.

References

Leonard E Baum and Ted Petrie. 1966. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, pages 1554–1563.

Léon Bottou. 1991. Stochastic gradient learning in neural networks. In *Proceedings of Neuro-Nîmes 91*, Nimes, France. EC2.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.

Jianfeng Gao, Mu Li, Andi Wu, and Chang-Ning Huang. 2005. Chinese word segmentation and named entity recognition: A pragmatic approach. *Comput. Linguist.*, 31(4):531–574, December.

Dan Garrette and Jason Baldridge. 2012. Type-supervised hidden markov models for part-of-speech tagging with incomplete tag dictionaries. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 821–831, Jeju Island, Korea, July. Association for Computational Linguistics.

Wenbin Jiang, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and pos tagging – a case study. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 522–530, Suntec, Singapore, August. Association for Computational Linguistics.

Wenbin Jiang, Meng Sun, Yajuan Lü, Yating Yang, and Qun Liu. 2013. Discriminative learning with natural annotations: Word segmentation as a case study. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 761–769, Sofia, Bulgaria, August. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Zhongguo Li and Maosong Sun. 2009. Punctuation as implicit annotations for chinese word segmentation. *Comput. Linguist.*, 35(4):505–512, December.

D. C. Liu and J. Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(3):503–528, December.

Yang Liu and Yue Zhang. 2012. Unsupervised domain adaptation for joint segmentation and POS-tagging. In *Proceedings of COLING 2012: Posters*, pages 745–754, Mumbai, India, December. The COLING 2012 Organizing Committee.

Gideon S. Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proceedings of ACL-08: HLT*, pages 870–878, Columbus, Ohio, June. Association for Computational Linguistics.

Yanxin Shi and Mengqiu Wang. 2007. A dual-layer crfs based joint decoding method for cascaded segmentation and labeling tasks. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1707–1712, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Yan Song, Prescott Klassen, Fei Xia, and Chunyu Kit. 2012. Entropy-based training data selection for domain adaptation. In *Proceedings of COLING 2012: Posters*, pages 1191–1200, Mumbai, India, December. The COLING 2012 Organizing Committee.

Weiwei Sun and Jia Xu. 2011. Enhancing chinese word segmentation using unlabeled data. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 970–979, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using gigaword scale unlabeled data. In *Proceedings of ACL-08: HLT*, pages 665–673, Columbus, Ohio, June. Association for Computational Linguistics.

Huihsin Tseng. 2005. A conditional random field word segmenter. In *In Fourth SIGHAN Workshop on Chinese Language Processing*.

Yuta Tsuboi, Hisashi Kashima, Shinsuke Mori, Hiroki Oda, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 897–904, Manchester, UK, August. Coling 2008 Organizing Committee.

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving chinese word segmentation and pos tagging with semi-supervised methods using large auto-analyzed data. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 309–317, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Pak-kwong Wong and Chorkin Chan. 1996. Chinese word segmentation based on maximum matching and word binding force. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 200–203, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zimin Wu and Gwyneth Tseng. 1993. Chinese text segmentation for text retrieval: Achievements and problems. *J. Am. Soc. Inf. Sci.*, 44(9):532–542, October.

Nianwen Xue and Libin Shen. 2003. Chinese word segmentation as lmr tagging. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing - Volume 17*, SIGHAN '03, pages 176–179, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shiwen Yu, Jianming Lu, Xuefeng Zhu, Huiming Duan, Shiyong Kang, Honglin Sun, Hui Wang, Qiang Zhao, and Weidong Zhan. 2001. Processing norms of modern chinese corpus. Technical report.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 840–847, Prague, Czech Republic, June. Association for Computational Linguistics.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Type-supervised domain adaptation for joint segmentation and pos-tagging. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 588–597, Gothenburg, Sweden, April. Association for Computational Linguistics.

Hai Zhao and Chunyu Kit. 2008. An empirical comparison of goodness measures for unsupervised chinese word segmentation with a unified framework. In *In: The Third International Joint Conference on Natural Language Processing (IJCNLP-2008*.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2010. A unified character-based tagging framework for chinese word segmentation. 9(2):5:1–5:32, June.

Balanced Korean Word Spacing with Structural SVM

Changki Lee* **Edward Choi** **Hyunki Kim**
*Kangwon National University, Chuncheon-si, Gangwondo, 200-701, Korea
Electronics and Telecommunications Research Institute, Daejeon, 305-350, Korea
leeck@kangwon.ac.kr mp2893@gmail.com hkk@etri.re.kr

Abstract

Most studies on statistical Korean word spacing do not utilize the information provided by the input sentence and assume that it was completely concatenated. This makes the word spacer ignore the correct spaced parts of the input sentence and erroneously alter them. To overcome such limit, this paper proposes a structural SVM-based Korean word spacing method that can utilize the space information of the input sentence. The experiment on sentences with 10% spacing errors showed that our method achieved 96.81% F-score, while the basic structural SVM method only achieved 92.53% F-score. The more the input sentence was correctly spaced, the more accurately our method performed.

1 Introduction

Automatic word spacing is a task to decide boundaries between words, which is frequently used for correcting spacing errors of text messages, Tweets, or Internet comments before using them in information retrieval applications (Lee and Kim, 2012). It is also often used in post-processing optical character recognition (OCR) or voice recognition (Lee et al., 2007). Except for some Asian languages such as Chinese, Japanese and Thai, most languages have explicit word spacing that improves the readability of the text and helps readers better understand the meaning of it. Korean especially has a tricky word spacing system and users often make mistakes, which makes automatic word spacing an interesting and essential task.

In order to easily acquire the training data, most studies on statistical Korean word spacing assume that well-spaced raw text (e.g. newspaper articles) is perfectly spaced and use it for training (Lee and Kim, 2012; Lee and Kim, 2013; Lee et al., 2007; Shim, 2011). This approach, however, cannot observe incorrect spacing since the assumption makes the training data devoid of negative example. Consequently, word spacers cannot use the spacing information given by the user, and erroneously alter the correctly spaced parts of the sentence. To utilize the user-given spacing information, a corpus of input sentences and their correctly spaced version is necessary. Constructing such corpus, however, requires much time and resource.

In this paper, to resolve such issue, we propose a structural SVM-based Korean word spacing model that can utilize the word spacing information given by the user. We name the proposed model "Balanced Word Spacing Model (BWSM)". Our approach trains a basic structural SVM-based Korean word spacing model as in (Lee and Kim, 2013), and tries to obtain the sentence which achieves the maximum score for the basic model while minimally altering the input sentence.

In the following section, we discuss related studies. In Section 3, the proposed method and its relation to Karush-Kuhn-Tucker (KKT) condition are explained. The experiment and discussion is presented in Section 4. Finally, in Section 5, the conclusion and future work for this study is given.

2 Related Work

There are two common approaches to Korean word spacing: rule-based approach and statistical approach. In rule-based approach, it is not easy to construct rules and maintain them. Furthermore, it requires morphological analysis to apply rule-based approach, which slows down the process. Recent studies, therefore, mostly focus on the statistical approach.

Most statistical approaches use well-spaced raw corpus as training data (e.g. newspaper articles) assuming that they are perfectly spaced. This is to avoid the expensive job of constructing new training data. Lee et al. (2007) treated the word spacing task as a sequence labeling problem on the input sentence which is a sequence of syllables. They proposed a method based on Hidden Markov Model (HMM). Shim (2011) also considered the word spacing task as a sequence labeling problem and proposed a method using Conditional Random Field (CRF) (Lafferty et al., 2001), which is a well-known powerful model for sequence labeling tasks. Lee and Kim

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 875–879,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

(2013) tried to solve the sequence labeling problem using structural SVM (Tsochantaridis et al., 2004; Joachims et al., 2009; Lee and Jang 2010; Shalev-Shwartz et al., 2011).

The studies above (Lee and Kim, 2013; Lee et al., 2007; Shim, 2011), however, do not take advantage of the spacing information provided by the user, and often erroneously alter the correctly spaced part of the sentence. Lee et al. (2007) tries to resolve this issue by combining an HMM model with an additional confidence model constructed from another corpus. Given an input sentence, they first apply the basic HMM model to obtain a candidate sentence. For every different word spacing between the input sentence and the candidate sentence, they calculate and compare the confidence using the confidence model, and whichever gets the higher confidence is used. The spacing accuracy was improved from 97.52% to 97.64%[1].

This study is similar to (Lee et al., 2007) in that it utilizes the spacing information given by the user. But unlike (Lee et al., 2007), BWSM uses structural SVM as the basic model and do not require an additional confidence model. Furthermore, while Lee et al. (2007) compares the spacing confidence for each syllable to obtain the final outcome, BWSM considers the whole sentence when altering its spacing, enabling it to achieve higher improvement on performance (from 92.53% F-score to 96.81% F-score).

3 Balanced Word Spacing Model

Like previous studies, the proposed model treats the Korean word spacing task as a sequence labeling problem. The label consists of *B* and *I*, which are assigned to each syllable of the sentence. Assuming that $\mathbf{x} = \langle x_1, x_2, \ldots, x_T \rangle$ is a sequence of total *T* syllables of the input sentence and $\mathbf{y} = \langle y_1, y_2, \ldots, y_T \rangle$ is a sequence of labels for each syllable, an example could be given as follows[2]:

Input: ah/beo/ji/ga bang/eh deul/eo/ga/sin/da (Father entered the room)
x = <ah, beo, ji, ga, bang, eh, deul, eo, ga, sin, da>
y = <B, I, I, I, B, I, B, I, I, I, I>

Figure 1: An example of word spacing.

In order to utilize the spacing information provided by the user, we propose a new model, the Balanced Word Spacing Model that adheres to the following principles:

1. The model must obtain the most likely sequence of labels($\mathbf{y}^*$), while minimally altering the user-given sequence of labels ($\mathbf{y}_{input}$).
2. We assume that it costs α per syllable to change the spacing of the original sentence, in order to keep the original spacing information as much as possible.

Mathematically formulating the above principles would give us the following equation:

$$\mathbf{y}^* = \text{argmax}\{score(\mathbf{x}, \mathbf{y}) - \alpha \cdot L(\mathbf{y}_{input}, \mathbf{y})\} \quad (1)$$

In Equation 1, $score(\mathbf{x}, \mathbf{y})$ calculates how compatible the sequence of label **y** is with the input sentence **x**. It is calculated by a basic word spacing model as in (Lee and Kim, 2013). $L(\mathbf{y}_{input}, \mathbf{y})$ counts the number of different labels between the user-given sequence of labels $\mathbf{y}_{input}$ and an arbitrary sequence of labels **y**. $\mathbf{y}^*$ of Equation 1 can be obtained by setting the gradient of $score(\mathbf{x}, \mathbf{y}) - \alpha \cdot L(\mathbf{y}_{input}, \mathbf{y})$ to 0, which is equivalent to the following equation:

$$\nabla score(\mathbf{x}, \mathbf{y}^*) = \alpha \cdot \nabla L(\mathbf{y}_{input}, \mathbf{y}^*) \quad (2)$$

In order to view the proposed model in a different perspective, we consider BWSM in terms of Karush-Kuhn-Tucker (KKT) condition. KKT condition is a technique for solving optimization problems with inequality constraints. It is a generalized version of Lagrange multipliers, which is a technique for solving optimization problems with equality constraints. Converting the aforementioned principles to a constrained optimization problem gives:

$$\text{Maximize: } score(\mathbf{x}, \mathbf{y})$$
$$\text{subject to } L(\mathbf{y}_{input}, \mathbf{y}) \leq b \quad (3)$$

Equation 3 tries to obtain **y** that maximizes $score(\mathbf{x}, \mathbf{y})$, namely the score of the basic model, while maintaining $L(\mathbf{y}_{input}, \mathbf{y})$ below *b*, which is equivalent to altering the word spacing of the input sentence less than or equal to *b* times. To solve this constrained optimization problem, we apply KKT condition and define a new Lagrangian function as follows:

$$\Lambda(\mathbf{x}, \mathbf{y}, \alpha) = score(\mathbf{x}, \mathbf{y}) - \alpha\{L(\mathbf{y}_{input}, \mathbf{y}) - b\} \quad (4)$$

[1] Accuracy was calculated based on syllables.
[2] Slashes are used for distinguishing between syllables.

Setting the gradient of the Equation 4 to zero, namely $\nabla\Lambda(\mathbf{x}, \mathbf{y}, \alpha) = 0$, we get the following necessary conditions:

$$\begin{aligned}&\text{Stationarity: } \nabla score(\mathbf{x}, \mathbf{y}^*) = \alpha^*\{\nabla L(\mathbf{y}_{input}, \mathbf{y}^*)\}\\&\text{Primal feasibility: } L(\mathbf{y}_{input}, \mathbf{y}^*) \leq b\\&\text{Dual feasibility: } \alpha^* \geq 0\\&\text{Complementary slackness: } \alpha^*\{L(\mathbf{y}_{input}, \mathbf{y}^*) - b\} = 0 \quad (5)\end{aligned}$$

Comparing Equation 1 with Equation 4 reveals that they are the same except the constant b. And $\mathbf{y}^*$ which satisfies the conditions of Equation 5, and hence the solution to Equation 4, is also the same as $\mathbf{y}^*$ which satisfies Equation 2, and hence the solution to Equation 1.

For the basic word spacing model, we use margin rescaled version of structural SVM as Lee and Kim (2013). The objective function of structural SVM is as follows:

$$\begin{aligned}&\min_{w,\xi} \frac{1}{2} \| \mathbf{w} \|^2 + \frac{C}{n}\sum_i^n \xi_i, \ \ s.t. \ \forall i, \ \xi_i \geq 0\\&\forall i, \forall \mathbf{y} \in Y \backslash \mathbf{y}_i : \mathbf{w}^{\mathrm{T}} \delta\Psi(\mathbf{x}_i, \mathbf{y}) \geq L(\mathbf{y}_i, \mathbf{y}) - \xi_i\\&\text{where } \delta\Psi(\mathbf{x}_i, \mathbf{y}) = \Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \mathbf{y}) \quad (6)\end{aligned}$$

In Equation 6, $(\mathbf{x}_i, \mathbf{y}_i)$ represents the i-th sequence of syllables and its correct spacing labels. $L(\mathbf{y}_i, \mathbf{y})$ is a loss function that counts the number of different labels between the correct labels $\mathbf{y}_i$ and the predicted sequence of labels $\mathbf{y}$. $\Psi(\mathbf{x}, \mathbf{y})$ is a typical feature vector function. The features used for the basic word spacing model are the same features used in (Lee and Kim, 2013). Since structural SVM was used for the basic word spacing model, the score function of Equation 1 becomes $score(\mathbf{x}, \mathbf{y}) = \mathbf{w}^{\mathrm{T}}\Psi(\mathbf{x}, \mathbf{y})$.

We propose two approaches for implementing Equation 1.

1. N-best re-ranking: N-best sequences of spacing labels are obtained using the basic structural SVM model. For each of the sequence, $\alpha^* L(\mathbf{y}_{input}, \mathbf{y}^*)$ is calculated and subtracted from $score(\mathbf{x}, \mathbf{y})$. The result of the subtraction is used to re-rank the sequences, and the one with the highest rank is chosen.
2. Modified Viterbi search: Viterbi search algorithm, which is used in the basic word spacing model to solve $\mathbf{y}^* = \text{argmax}\{\mathbf{w}^{\mathrm{T}}\Psi(\mathbf{x}, \mathbf{y})\}$, is modified to solve $\mathbf{y}^* = \text{argmax}\{\mathbf{w}^{\mathrm{T}}\Psi(\mathbf{x}, \mathbf{y}) - \alpha \cdot L(\mathbf{y}_{input}, \mathbf{y})\}$. Both $\Psi(\mathbf{x}, \mathbf{y})$ and $L(\mathbf{y}_{input}, \mathbf{y})$ can be calculated syllable by syllable, which makes it easy to modify Viterbi search algorithm.

The first approach seems straightforward and easy, but it would take a long time to obtain N-best sequences of labels. Furthermore, the correct label sequence might not be in those N-best sequences, hence degrading the overall performance. The second approach is fast since it does not calculate N-best sequences, and unlike the first approach, will always consider the correct label sequence as a candidate.

4 Experiment

In order to compare the performance of BWSM with HMM-based Korean word spacing and structural SVM-based Korean word spacing, we use Sejong raw corpus (Kang and Kim, 2004) as train data and ETRI POS tagging corpus as test data[3]. Pegasos-struct algorithm from (Lee and Kim, 2013) was used to train the basic structural SVM-based model. The optimal value for the tradeoff variable C of structural SVM was found after conducting several experiments[4].

The rate of word spacing error varies depending on the corpus. Newspaper articles rarely have word spacing errors but text messages or Tweets frequently contain word spacing errors. To reflect such variety, we randomly insert spacing errors into the test set to produce various test sets with spacing error rate 0%, 10%, 20%, 35%, 50%, 60%, and 70%[5].

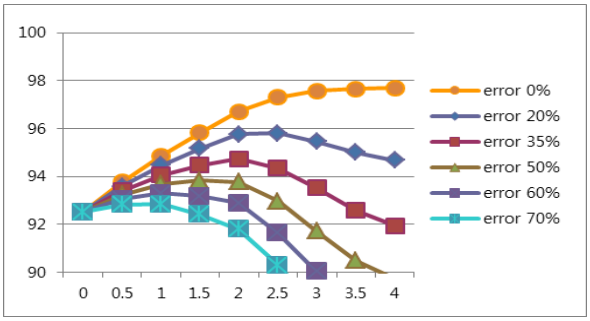


Figure 2: Word-based F-score of N-best re-ranking approach.

Figure 2 shows the relation between α(x-axis) and word-based F-score[6](y-axis) of N-best re-

[3] The number of words for the training set and test set are 26 million and 290,000 respectively.
[4] We experimented with 10, 100, 1000, 10000, 100000 and 1000000, the optimal value being 100000.
[5] We altered the input to the system and retained the original gold standard's space unit.
[6] Word-based F-score = 2*$\text{Prec}_{\text{word}}$*$\text{Recall}_{\text{word}}$ / ($\text{Prec}_{\text{word}}$ + $\text{Recall}_{\text{word}}$),
$\text{Prec}_{\text{word}}$ = (# of correctly spaced words) / (the total number of words produced by the system),

ranking approach using test sets with different spacing error rate. When $\alpha = 0$, BWSM becomes a normal structural SVM-based model. As α increases, F-score also increases for a while but decreases afterward. And F-score increases more when using test sets with low error rate. It is worth noticing that when using the test set with 0% error rate, as α increases, F-score converges to 98%. The reason it does not reach 100% is that the correct label sequence is sometimes not included in the N-best sequences.

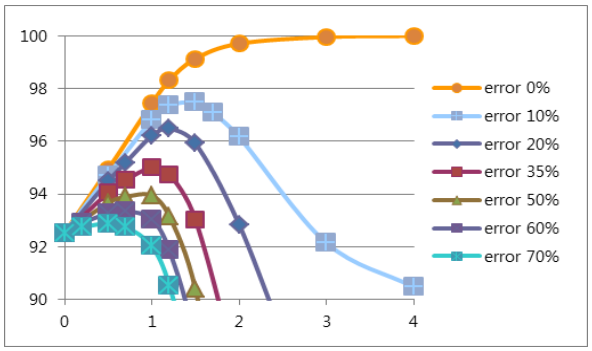


Figure 3: Word-based F-score of modified Viterbi search.

Figure 3 shows the relation between α(x-axis) and word-based F-score(y-axis) of modified Viterbi search approach using test sets with different spacing error rate. The graphs are similar to Figure 2, but F-score reaches higher values compared to N-best re-ranking approach. Notice that, when using the test set with 0% error rate, F-score becomes 100% as α surpasses 3. This is because, unlike N-best re-ranking approach, modified Viterbi search approach considers all possible sequences as candidates.

From Figure 2 and 3, it can be seen that BWSM, which takes into consideration the spacing information provided by the user, can improve performance significantly. It is also apparent that modified Viterbi search approach outperforms N-best re-ranking approach. The optimal value for α varies as test sets with different error rate are used. It is natural that, for test sets with low error rate, the optimal value of α increases, thus forcing the model to more utilize the user-given spacing information. It is difficult to automatically obtain the optimal α for an arbitrary input sentence. Therefore we set α to 1, which, according to Figure 3, is more or less the optimal value for most of the test sets.

$\text{Recall}_{\text{word}}$ = (# of correctly spaced words) / (the total number of words in the test data)

Model	Syllable based precision	Word based precision
HMM (Lee et al., 2007)	98.44	90.31
S-SVM (Lee and Kim, 2013)	99.01	92.53
Modified Viterbi (error rate 10%)	**99.64**	**96.81**
Modified Viterbi (error rate 20%)	**99.55**	**96.21**
Modified Viterbi (error rate 35%)	**99.35**	**95.01**

Table 1: Precision of BWSM and previous studies

With α set to 1, and using modified Viterbi search algorithm, the performance of BWSM is shown in Table 1 with other previous studies (Lee and Kim, 2013; Lee et al., 2007). Table 1 shows that BWSM gives superior performance than other studies that do not utilize user-given spacing information.

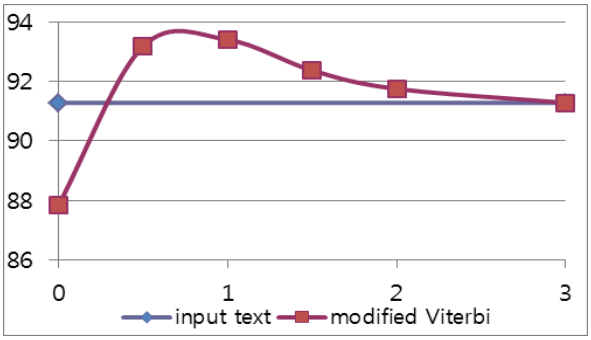


Figure 4: Word-based F-score of modified Viterbi search on Tweets.

We also collected Tweets from Twitter and tested modified Viterbi algorithm on them. Figure 4 shows the relation between α (x-axis) and word-based F-score (y-axis). The raw Tweets showed word-based F-score of approximate 91%, and the basic structural SVM model ($\alpha = 0$) showed somewhat inferior 88%. Modified Viterbi algorithm showed the similar behavior as Figure 3, showing 93.2~93.4% word-based F-score when α was set to 0.5~1. Figure 4 shows that BWSM is effective not only on text with randomly inserted spacing errors, but also on actual data, Tweets.

5 Conclusion

In this paper, we proposed BWSM, a new structural SVM-based Korean word spacing model that utilizes user-given spacing information. BWSM can obtain the most likely sequence of spacing labels while minimally altering the word spacing of the input sentence. Experiments on test sets with various error rate showed that BWSM significantly improved word-based F-

score, from 95.47% to 98.39% in case of the test set with 10% error rate.

For future work, there are two interesting directions. First is to improve BWSM so that it can automatically obtain the optimal value of α for an arbitrary sentence. This will require a training set consisting of text with actual human spacing errors and its corrected version. Second is to apply BWSM to other interesting problems such as named entity recognition (NER). Newspaper articles often use certain symbols such as quotation marks or brackets around the titles of movies, songs and books. Such symbols can be viewed as user-given input, which BWSM will try to respect as much as possible while trying to find the most likely named entities.

Acknowledgments

This work was supported by the IT R&D program of MSIP/KEIT (10044577, Development of Knowledge Evolutionary WiseQA Platform Technology for Human Knowledge Augmented Services). We would like to thank the anonymous reviewers for their comments.

References

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. 2009. Cutting-plane training of structural SVMs. *Machine Learning*, Vol. 77, No. 1.

Beom-mo Kang and Hunggyu Kim. 2004. Sejong Korean Corpora in the making. In *Proceedings of the LREC*, 1747-1750.

John Lafferty, Andrew McCallum and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the ICML*, 282-289.

Changki Lee and Myung-Gil Jang. 2010. A Modified Fixed-threshold SMO for 1-Slack Structural SVM. *ETRI Journal,* Vol.32, No.1, 120-128.

Changki Lee and Hyunki Kim. 2012. Automatic Korean word spacing using structural SVM. In *Proceedings of the KCC,* 270-272.

Changki Lee and Hyunki Kim. 2013. Automatic Korean word spacing using Pegasos algorithm. *Information Processing and Management*, Vol. 49, No. 1, 370-379.

Do-Gil Lee, Hae-Chang Rim and Dongsuk Yook. 2007. Automatic word spacing using probabilistic models based on character n-grams. *Intelligent Systems IEEE*, Vol. 22, No. 1, 28-35.

Seung-Wook Lee, Hae-Chang Rim and So-Young Park. 2007. A new approach for Korean word spacing incorporating confidence value of user's input. In *Proceedings of the ALPIT*, 92-97.

Kwang-Sup Shim. 2011. Automatic word spacing based on Conditional Random Fields. *Korean Journal of Cognitive Science*, Vol. 22, No. 2, 217-233.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. 2004. Pegasos: Primal estimated sub-gradient solver for SVM. *Mathematical Programming*, Vol. 127, No. 1.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the ICML*, 104-111.

Morphological Segmentation for Keyword Spotting

Karthik Narasimhan[1], **Damianos Karakos**[2], **Richard Schwartz**[2], **Stavros Tsakalidis**[2], **Regina Barzilay**[1]
[1]Computer Science and Artificial Intelligence Laboratory,
Massachusetts Institute of Technology
[2]Raytheon BBN Technologies
{karthikn, regina}@csail.mit.edu
{dkarakos, schwartz, stavros}@bbn.com

Abstract

We explore the impact of morphological segmentation on keyword spotting (KWS). Despite potential benefits, state-of-the-art KWS systems do not use morphological information. In this paper, we augment a state-of-the-art KWS system with sub-word units derived from supervised and unsupervised morphological segmentations, and compare with phonetic and syllabic segmentations. Our experiments demonstrate that morphemes improve overall performance of KWS systems. Syllabic units, however, rival the performance of morphological units when used in KWS. By combining morphological, phonetic and syllabic segmentations, we demonstrate substantial performance gains.

1 Introduction

Morphological analysis plays an increasingly important role in many language processing applications. Recent research has demonstrated that adding information about word structure increases the quality of translation systems and alleviates sparsity in language modeling (Chahuneau et al., 2013b; Habash, 2008; Kirchhoff et al., 2006; Stallard et al., 2012).

In this paper, we study the impact of morphological analysis on the keyword spotting (KWS) task. The aim of KWS is to find instances of a given keyword in a corpus of speech data. The task is particularly challenging for morphologically rich languages as many target keywords are unseen in the training data. For instance, in the Turkish dataset (Babel, 2013) we use, from the 2013 IARPA Babel evaluations, 36.06% of the test words are unseen in the training data. However, 81.44% of these unseen words have a morphological variant in the training data. Similar patterns are observed in other languages used in the Babel evaluations. This observation strongly supports the use of morphological analysis to handle out-of-vocabulary (OOV) words in KWS systems.

Despite this potential promise, state-of-the-art KWS systems do not commonly use morphological information. This surprising fact can be due to multiple reasons, ranging from the accuracy of existing morphological analyzers to the challenge of integrating morphological information into existing KWS architectures. While using morphemes is likely to increase coverage, it makes recognition harder due to the inherent ambiguity in the recognition of smaller units. Moreover, it is not clear a priori that morphemes, which are based on the semantics of written language, are appropriate segmentation units for a speech-based application.

We investigate the above hypotheses in the context of a state-of-the-art KWS architecture (Karakos et al., 2013). We augment word lattices with smaller units obtained via segmentation of words, and use these modified lattices for keyword spotting. We consider multiple segmentation algorithms, ranging from near-perfect supervised segmentations to random segmentations, along with unsupervised segmentations and purely phonetic and syllabic segmentations. Our experiments show how sub-word units can be used effectively to improve the performance of KWS systems. Further, we study the extent of impact of the subwords, and the manner in which they can be used in KWS systems.

2 Related Work

Prior research on applications of morphological analyzers has focused on machine translation, language modeling and speech recognition (Habash, 2008; Chahuneau et al., 2013a; Kirchhoff et al., 2006). Morphological analysis enables us to link together multiple inflections of the same root, thereby alleviating word sparsity common in mor-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 880–885,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

phologically rich languages. This results in improved language model perplexity, better word alignments and higher BLEU scores.

Recent work has demonstrated that even morphological analyzers that use little or no supervision can help improve performance in language modeling and machine translation (Chahuneau et al., 2013b; Stallard et al., 2012). It has also been shown that segmentation lattices improve the quality of machine translation systems (Dyer, 2009).

In this work, we leverage morphological segmentation to reduce OOV rates in KWS. We investigate segmentations produced by a range of models, including acoustic sub-word units. We incorporate these subword units into a lattice framework within the KWS system. We also demonstrate the value of using alternative segmentations instead of or in combination with morphemes. In addition to improving the performance of KWS systems, this finding may also benefit other applications that currently use morphological segmentation for OOV reduction.

3 Segmentation Methods

Supervised Morphological Segmentation Due to the unavailability of gold morphological segmentations for our corpus (Babel, 2013), we use a resource-rich supervised system as a proxy. As training data for this system, we use the MorphoChallenge 2010 corpus[1] which consists of 1760 gold segmentations for Turkish.

We consider two supervised frameworks, both made up of two stages. In the first stage, common to both systems, we use a FST-based morphological parser (Çöltekin, 2010) that generates a set of candidate segmentations, leveraging a large database of Turkish roots and affixes. This stage tends to overgenerate, segmenting each word in eight different ways on average. In the next stage, we filter the resulting segmentations using one of two supervised filters (described below) trained on the MorphoChallenge corpus.

In the first approach, we use a binary log-linear classifier to accept/reject each segmentation hypothesis. For each word, this classifier may accept multiple segmentations, or rule out all the alternatives. In the second approach, to control the number of segmentations per word, we train a log-linear ranker that orders the segmentations for a word in decreasing order of likelihood. In our training corpus, each word has on average 2.5 gold segmentations. Hence, we choose the top two segmentations per word from the output of the ranker to use in our KWS system. In both filters, we use several features like morpheme unigrams, bigrams, lengths, number of morphemes, and phone sequences corresponding to the morphemes.

Feature	Example
morpheme unigrams	tak, acak
morpheme bigram	⟨tak, acak⟩
phonetic seq. unigrams	t.a.k., 1v.dZ.a.k.
phonetic seq. bigram	⟨t.a.k., 1v.dZ.a.k.⟩
number of morphemes	2
morpheme lengths	3, 4

Table 1: Example of features used in the supervised filters for the segmentation *tak-acak*. Each phone is followed by a dot for clarity.

In our supervised systems, we can encode features that go beyond individual boundaries, like the total number of morphemes in the segmentation. This global view distinguishes our classifier/ranker from traditional approaches that model segmentation as a sequence tagging task (Ruokolainen et al., 2013; Kudo et al., 2004; Kruengkrai et al., 2006). Another departure of our approach is the use of phonetic information, in the form of phonetic sequences corresponding to the morpheme unigrams and bigrams. The hypothesis is that syllabic boundaries are correlated with morpheme boundaries to some extent. The phonetic sequences for words are obtained using a publicly available Text-to-Phone (T2P) system (Lenzo, 1998).

Unsupervised Morphological Segmentation We employ a widely-used unsupervised system Morfessor (Creutz and Lagus, 2005) which achieves state-of-the-art unsupervised performance in the MorphoChallenge evaluation. Morfessor uses probabilistic generative models with sparse priors which are motivated by the Minimum Description Length (MDL) principle. The system derives segmentations from raw data, without reliance on extra linguistic sources. It outputs a single segmentation per word.

Random Segmentation As a baseline, we include sub-word units from random segmentations, where we mark a segmentation boundary at each character position in a word with a fixed probability p. For comparison purposes, we consider two

[1] http://research.ics.aalto.fi/events/morphochallenge2010/

Sub-word units	Example
Morphemes	tak - acak
Random	t - aka - c - a - k
Phones	t - a - k - 1v - dZ - a - k
Syllables	ta - k1v - dZak

Table 2: Segmentations of the word *takacak* into different types of sub-word units.

types of random segmentations that match the supervised morphological segmentations in terms of the number of uniques morphemes and the average morpheme length, respectively. These segmentations are obtained by adjusting the segmentation probability p appropriately.

Phones and Syllables In addition to letter-based segmentation, we also consider other subword units that stem from word acoustics. In particular, we consider segmentation using phones and syllables, which are available for the Babel data we work with.

Table 2 shows examples of different segmentations for the Turkish word *takacak*.

4 Keyword Spotting

The keyword spotting system used in this work follows, to a large extent, the pipeline of (Bulyko et al., 2012). Using standard speech recognition machinery, the system produces a detailed lattice of word hypotheses. The resulting lattice is used to extract keyword hits with nominal posterior probability scores.

We modify this basic architecture in two ways. First, we use subwords instead of whole-words in the decoding lexicon. Second, we represent keywords using all possible paths in a lattice of subwords. For each sequence of matching arcs in the lattice, the posteriors of these arcs are multiplied together to form the score of detection (hit). A post-processing step adds up (or takes the max of) the scores of all hits of each keyword which have significant overlap in time. Finally, the hit lists are processed by the score normalization and combination method described in (Karakos et al., 2013).

We use whole-word extraction for words in vocabulary, but rely on subword models for OOV words. Since we combine the hits separately for IV and OOV keywords, using subwords can only improve the performance of the overall system.

Language	Dev Set	Eval Set
Turkish	403	226
Assamese	158	563
Bengali	176	629
Haitian	107	319
Lao	110	194
Tamil	238	700
Zulu	323	1251

Table 3: Number of OOV keywords in the different Dev and Eval sets.

5 Experimental Setup

Data The segmentation algorithms described in Section 3 are tested using the setup of the KWS system described in Section 4. Our experiments are conducted using the IARPA Babel Program language collections for Turkish, Assamese, Bengali, Haitian, Lao, Tamil and Zulu (Babel, 2013)[2]. The dataset contains audio corpora and a set of keywords. The training corpus for KWS consists of 10 hours of speech, while the development and test sets have durations of 10 and 5 hours, respectively. We evaluate KWS performance over the OOV keywords in the data, which are unseen in the training set, but appear in the development/test set. Table 3 contains statistics on the number of OOV keywords in the data for each language.

In our experiments, we consider the pre-indexed condition, where the keywords are known only after the decoding of the speech has taken place.

Evaluation Measures We consider two different evaluation metrics. To evaluate the accuracy of the different segmentations, we compare them against gold segmentations from the MorphoChallenge data for Turkish. This set consists of 1760 words, which are manually segmented. We use a measure of word accuracy (**WordAcc**), which captures the accuracy of all segmentation decisions within the word. If one of the segmentation boundaries is wrong in a proposed segmentation, then that segmentation does not contribute towards the WordAcc score. We use 10-fold cross-validation for the supervised segmentations, while we use the entire set for unsupervised and acoustic cases.

We evaluate the performance of our KWS system using a widely used metric in KWS, the Ac-

[2]We perform the experiments with supervised segmentation only on Turkish, due to the lack of gold morphological data for the other languages.

tual Term Weighted Value (ATWV) measure, as described in (Fiscus et al., 2007). This measure uses a combination of penalties for misses and false positives to score the system. The maximum score achievable is 1.0, if there are no misses and false positives, while the score can be lower than 0.0 if there are a lot of misses or false positives.

6 Results

Table 4 summarizes the performance of all considered segmentation systems in the KWS task on Turkish. The quality of the segmentations compared to the gold standard is also shown. Table 5 shows the OOV ATWV performance on the six other languages, used in the second year of the IARPA Babel project. We summarize below our conclusions based on these results.

Using sub-word units improves overall KWS performance If we use a word-based KWS system, the ATWV score will be 0.0 since the OOV keywords are not present in the lexicon. Enriching our KWS system with sub-word segments yields performance gains for all the segmentation methods, including random segmentations. However, the observed gain exhibits significant variance across the segmentation methods. For instance, the gap between the performance of the KWS system using the best supervised classifier-based segmenter (*CP*) and that using the unsupervised segmenter (*U*) is 0.059, which corresponds to a 43.7% in relative gain. Table 4 also shows that while methods with shorter sub-units (*U*, *P*) yield lower OOV rate, they do not necessarily fare better in the KWS evaluation.

Syllabic units rival the performance of morphological units A surprising discovery from our experiments is the good performance of the syllabic segmentation-based KWS system (*S*). It outperforms all the alternative segmentations on the test set, and ranks second on the development set behind the *CP* system. These units are particularly attractive as they can easily be computed from acoustic input and do not require any prior linguistic knowledge. We hypothesize that the granularity of this segmentation is crucial to its success. For instance, a finer-grained phone-based segmentation (*P*) performs substantially worse than other segmentation algorithms as the derived sub-units are shorter and hence, harder to recognize.

Improving morphological accuracy beyond a certain level does not translate into improved KWS performance We observe that the segmentation accuracy and KWS performance are not positively correlated. Clearly, bad segmentations translate into poor ATWV scores, as in the case of random and unsupervised segmentations. However, gains on segmentation accuracy do not always result in better KWS performance. For instance, the ranker systems (*RP, RNP*) have better accuracies on MC2010, while the classifier systems (*CP, CNP*) perform better on the KWS task. This discrepancy in performance suggests that further gains can be obtained by optimizing segmentations directly with respect to KWS metrics.

Adding phonetic information improves morphological segmentation For all the morphological systems, adding phonetic information results in consistent performance gains. For instance, it increases segmentation accuracy by 4% when added to the classifier (*CNP* and *CP* in table 4). The phonetic information used in our experiments is computed automatically using a T2P system (Lenzo, 1998), and can be easily obtained for a range of languages. This finding sheds new light on the relation between phonetic and morphological systems, and can be beneficial for morphological analyzers developed for other applications.

Combining morphological, phonetic and syllabic segmentations gives better results than either in isolation As table 4 shows, the best KWS results are achieved when syllabic and morphemic systems are combined. The best combination system (*CP+P+S*) outperforms the best individual system (*S*) by 5.5%. This result suggests that morphemic, phonemic and syllabic segmentations encode complementary information which benefits KWS systems in handling OOV keywords.

Morphological segmentation helps KWS across different languages Table 5 demonstrates that we can obtain gains in KWS performance across different languages using unsupervised segmentation. The improvement is significant in 3 of the 6 languages - as high as 3.2% for Assamese and Bengali, and 2.7% for Tamil (absolute percentages). As such, the results of Table 2 cannot be directly compared to those of Table 1 since the system architecture is slightly different[3]. How-

[3]The keyword spotting pipeline is based on the one used by the Babelon team in the 2014 NIST evaluation (Tsakalidis, 2014). The pipeline was much more involved than the one described for Turkish; multiple search methods (with/without fuzzy search) and data structures (lattices, confusion networks and generalized versions of these) were all used in combination (Karakos and Schwartz, 2014). The recognition

Method	Unique units	Avg. unit length	Reduction in OOV (abs)	WordAcc	Dev ATWV	Test ATWV
Phone-based (P)	51	1	36.06%	0.06%	0.099	0.164
Syllable-based (S)	2.1k	3.62	23.91%	10.29%	0.127	0.201
Classifier w/ phone info (CP)	18.5k	6.39	18.20%	80.41%	0.146	0.194
Classifier w/o phone info (CNP)	19k	6.42	21.50%	75.66%	0.133	0.181
Ranker w/ phone info (RP)	10k	5.62	16.86%	**86.03%**	0.104	0.153
Ranker w/o phone info (RNP)	10k	5.71	16.44%	84.19%	0.109	0.159
Unsupervised (U)	2.4k	5.44	22.45%	39.57%	0.080	0.135
RANDLen-Classifier	11.7k	6.39	0.73%	5.11%	0.061	0.086
RANDNum-Classifier	18.2k	3.03	8.56%	3.69%	0.111	0.154
RANDLen-Ranker	11.6k	5.62	1.94%	5.79%	0.072	0.136
RANDNum-Ranker	11.7k	6.13	1.15%	5.34%	0.081	0.116
CP + P	-	-	-	-	0.190	0.246
RP + P	-	-	-	-	0.150	0.210
CP + P + S	-	-	-	-	**0.208**	**0.257**
RP + P + S	-	-	-	-	0.186	0.249
Word-based for IV words	-	-	-	-	0.385	0.400

Table 4: Segmentation Statistics and ATWV scores on Babel Turkish data along with WordAcc on MorphoChallenge 2010 data. All rows except the last are for OOV words. Absolute reduction is from an initial OOV of 36.06%. Higher ATWV scores are better. Best system scores are shown in bold.

	Assamese		Bengali		Haitian		Lao		Tamil		Zulu	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
P + S	0.213	0.230	0.277	0.296	0.371	0.342	0.228	0.139	0.349	0.267	0.279	0.215
P + S + U	0.214	**0.263**	**0.294**	**0.328**	**0.393**	0.342	0.237	0.146	**0.395**	**0.284**	0.275	0.218

Table 5: ATWV scores for languages used in the second year of the IARPA Babel project, using two KWS systems: Phone + Syllable (P+S) and Phone + Syllable + Unsupervised Morphemes (P+S+U). Bold numbers show significant performance gains obtained by adding morphemes to the system.

ever, they are indicative of the large gains (1.5%, on average, over the six languages) that can be obtained through unsupervised morphology, on top of a very good combined phonetic/syllabic system.

7 Conclusion

We explore the extent of impact of morphological segmentation on keyword spotting (KWS). To investigate this issue, we augmented a KWS system with sub-word units derived by multiple segmentation algorithms. Our experiments demonstrate that morphemes improve the overall performance of KWS systems. Syllabic units, however, rival the performance of morphemes in the KWS task. Furthermore, we demonstrate that substantial performance gains in KWS performance are obtained by combining morphological, phonetic and syllabic segmentations. Finally, we also show that adding phonetic information improves the quality of morphological segmentation.

was done with audio features supplied by BUT (Karafiát et al., 2014), which were improved versions of those used for Turkish.

Acknowledgements

This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense US Army Research Laboratory contract number W911NF-12-C-0013. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government. We thank the MIT NLP group and the EMNLP reviewers for their comments and suggestions.

References

IARPA Babel. 2013. Language collection releases; Turkish: IARPA-babel105b-v0.4, Assamese: IARPA-babel102b-v0.5a, Bengali: IARPA-babel103b-0.4b, Haitian Creole: IARPA-babel201b-v0.2b, Lao: IARPA-babel203b-v3.1a, Tamil: IARPA-babel204b-v1.1b, Zulu: IARPA-babel206b-v0.1e.

Ivan Bulyko, Owen Kimball, Man-Hung Siu, José Herrero, and Dan Blum. 2012. Detection of unseen words in conversational Mandarin. In *Proc. of ICASSP*, Kyoto, Japan, Mar.

Victor Chahuneau, Eva Schlinger, Noah A. Smith, and Chris Dyer. 2013a. Translating into morphologically rich languages with synthetic phrases. In *EMNLP*, pages 1677–1687. ACL.

Victor Chahuneau, Noah A. Smith, and Chris Dyer. 2013b. Knowledge-rich morphological priors for bayesian language models. In *HLT-NAACL*, pages 1206–1215. The Association for Computational Linguistics.

Çağrı Çöltekin. 2010. A freely available morphological analyzer for Turkish. In *Proceedings of the 7th International conference on Language Resources and Evaluation (LREC2010)*, pages 820–827.

Mathias Creutz and Krista Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR)*, pages 106–113.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for MT. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 406–414, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jonathan G. Fiscus, Jerome Ajot, John S. Garofolo, and George Doddington. 2007. Results of the 2006 spoken term detection evaluation. In *Workshop on Searching Spontaneous Conversational Speech*.

Nizar Habash. 2008. Four techniques for online handling of out-of-vocabulary words in arabic-english statistical machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 57–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

Martin Karafiát, František Grézl, Mirko Hannemann, Karel Veselý, Igor Szoke, and Jan "Honza" Černocký. 2014. BUT 2014 Babel system: Analysis of adaptation in NN based systems. In *Proceedings of Interspeech 2014*, Singapore, September. IEEE.

Damianos Karakos and Richard Schwartz. 2014. Subword modeling. In *IARPA Babel PI Meeting*, July.

Damianos Karakos, Richard Schwartz, Stavros Tsakalidis, Le Zhang, Shivesh Ranjan, Tim Ng, Roger Hsiao, Guruprasad Saikumar, Ivan Bulyko, Long Nguyen, John Makhoul, Frantisek Grezl, Mirko Hannemann, Martin Karafiat, Igor Szoke, Karel Vesely, Lori Lamel, and Viet-Bac Le. 2013. Score normalization and system combination for improved keyword spotting. In *Proc. ASRU 2013*, Olomouc, Czech Republic.

Katrin Kirchhoff, Dimitra Vergyri, Jeff Bilmes, Kevin Duh, and Andreas Stolcke. 2006. Morphology-based language modeling for conversational arabic speech recognition. *Computer Speech and Language*, 20(4):589–608.

Canasai Kruengkrai, Virach Sornlertlamvanich, and Hitoshi Isahara. 2006. A conditional random field framework for Thai morphological analysis. In *LREC*.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *In Proc. of EMNLP*, pages 230–237.

Kevin Lenzo. 1998. Text-to-phoneme converter builder. `http://www.cs.cmu.edu/afs/cs.cmu.edu/user/lenzo/html/areas/t2p/`. Accessed: 2014-03-11.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 29–37, Sofia, Bulgaria, August. Association for Computational Linguistics.

David Stallard, Jacob Devlin, Michael Kayser, Yoong Keok Lee, and Regina Barzilay. 2012. Unsupervised morphology rivals supervised morphology for Arabic MT. In *ACL (2)*, pages 322–327. The Association for Computer Linguistics.

Stavros Tsakalidis. 2014. The Babelon OpenKWS14 systems. In *IARPA Babel PI Meeting*, July.

What Can We Get From 1000 Tokens? A Case Study of Multilingual POS Tagging For Resource-Poor Languages

Long Duong,[12] **Trevor Cohn,**[1] **Karin Verspoor,**[1] **Steven Bird,**[1] and **Paul Cook**[1]
[1]Department of Computing and Information Systems,
The University of Melbourne
[2]National ICT Australia, Victoria Research Laboratory
lduong@student.unimelb.edu.au
{t.cohn, karin.verspoor, sbird, paulcook}@unimelb.edu.au

Abstract

In this paper we address the problem of multilingual part-of-speech tagging for resource-poor languages. We use parallel data to transfer part-of-speech information from resource-rich to resource-poor languages. Additionally, we use a small amount of annotated data to learn to "correct" errors from projected approach such as tagset mismatch between languages, achieving state-of-the-art performance (91.3%) across 8 languages. Our approach is based on modest data requirements, and uses minimum divergence classification. For situations where no universal tagset mapping is available, we propose an alternate method, resulting in state-of-the-art 85.6% accuracy on the resource-poor language Malagasy.

1 Introduction

Part-of-speech (POS) tagging is a crucial task for natural language processing (NLP) tasks, providing basic information about syntax. Supervised POS tagging has achieved great success, reaching as high as 95% accuracy for many languages (Petrov et al., 2012). However, supervised techniques need manually annotated data, and this is either lacking or limited in most resource-poor languages. Fully unsupervised POS tagging is not yet useful in practice due to low accuracy (Christodoulopoulos et al., 2010). In this paper, we propose a semi-supervised method to narrow the gap between supervised and unsupervised approaches. We demonstrate that even a small amount of supervised data leads to substantial improvement.

Our method is motivated by the availability of parallel data. Thanks to the development of multilingual documents from government projects, book translations, multilingual websites, and so forth, parallel data between resource-rich and resource-poor languages is relatively easy to acquire. This parallel data provides the bridge that permits us to transfer POS information from a resource-rich to a resource-poor language.

Systems that make use of cross-lingual tag projection typically face several issues, including mismatches between the tagsets used for the languages, artifacts from noisy alignments and cross-lingual syntactic divergence. Our approach compensates for these issues by training on a small amount of annotated data on the target side, demonstrating that only 1k tokens of annotated data is sufficient to improve performance.

We first tag the resource-rich language using a supervised POS tagger. We then project POS tags from the resource-rich language to the resource-poor language using parallel word alignments. The projected labels are noisy, and so we use various heuristics to select only "good" training examples. We train the model in two stages. First, we build a maximum entropy classifier T on the (noisy) projected data. Next, we train a supervised classifier P on a small amount of annotated data (1,000 tokens) in the target language, using a minimum divergence technique to incorporate the first model, T. Compared with the state of the art (Täckström et al., 2013), we make more-realistic assumptions (e.g. relying on a tiny amount of annotated data rather than a huge crowd-sourced dictionary) and use less parallel data, yet achieve a better overall result. We achieved 91.3% average accuracy over 8 languages, exceeding Täckström et al. (2013)'s result of 88.8%.

The test data we employ makes use of mappings from language-specific POS tag inventories to a universal tagset (Petrov et al., 2012). However, such a mapping might not be available for resource-poor languages. Therefore, we also pro-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 886–897,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

pose a variant of our method which removes the need for identical tagsets between the projection model T and the correction model P, based on a two-output maximum entropy model over tag pairs. Evaluating on the resource-poor language Malagasy, we achieved 85.6% accuracy, exceeding the state-of-the-art of 81.2% (Garrette et al., 2013).

2 Background and Related Work

There is a wealth of prior work on multilingual POS tagging. The simplest approach takes advantage of the typological similarities that exist between languages pairs such as Czech and Russian, or Serbian and Croatian. They build the tagger — or estimate part of the tagger — on one language and apply it to the other language (Reddy and Sharoff, 2011, Hana et al., 2004).

Yarowsky and Ngai (2001) pioneered the use of parallel data for projecting tag information from a resource-rich language to a resource-poor language. Duong et al. (2013b) used a similar method on using sentence alignment scores to rank the goodness of sentences. They trained a seed model from a small part of the data, then applied this model to the rest of the data using self-training with revision.

Das and Petrov (2011) also used parallel data but additionally exploited graph-based label propagation to expand the coverage of labelled tokens. Each node in the graph represents a trigram in the target language. Each edge connects two nodes which have similar context. Originally, only some nodes received a label from direct label projection, and then labels were propagated to the rest of the graph. They only extracted the dictionary from the graph because the labels of nodes are noisy. They used the dictionary as the constraints for a feature-based HMM tagger (Berg-Kirkpatrick et al., 2010). Both Duong et al. (2013b) and Das and Petrov (2011) achieved 83.4% accuracy on the test set of 8 European languages.

Goldberg et al. (2008) pointed out that, with the presence of a dictionary, even an incomplete one, a modest POS tagger can be built using simple methods such as expectation maximization. This is because most of the time, words have a very limited number of possible tags, thus a dictionary that specifies the allowable tags for a word helps to restrict the search space. With a gold-standard dictionary, Das and Petrov (2011) achieved an accuracy of approximately 94% on the same 8 languages. The effectiveness of a gold-standard dictionary is undeniable, however it is costly to build one, especially for resource-poor languages. Li et al. (2012) used the dictionary from Wiktionary,[1] a crowd-sourced dictionary. They scored 84.8% accuracy on the same 8 languages. Currently, Wiktionary covers over 170 languages, but the coverage varies substantially between languages and, unsurprisingly, it is poor for resource-poor languages. Therefore, relying on Wiktionary is not effective for building POS taggers for resource-poor languages.

Täckström et al. (2013) combined both token information (from direct projected data) and type constraints (from Wiktionary's dictionary) to form the state-of-the-art multilingual tagger. They built a tag lattice and used these token and type constraints to prune it. The remaining paths are the training data for a CRF tagger. They achieved 88.8% accuracy on the same 8 languages.

Table 1 summarises the performance of the above models across all 8 languages. Note that these methods vary in their reliance on external resources. Duong et al. (2013b) use the least, i.e. only the Europarl Corpus (Koehn, 2005). Das and Petrov (2011) additionally use the United Nation Parallel Corpus. Li et al. (2012) didn't use any parallel text but used Wiktionary instead. Täckström et al. (2013) exploited more parallel data than Das and Petrov (2011) and also used a dictionary from Li et al. (2012).

Another approach for resource-poor languages is based on the availability of a small amount of annotated data. Garrette et al. (2013) built a POS tagger for Kinyarwanda and Malagasy. They didn't use parallel data but instead exploited four hours of manual annotation to build ∼4,000 tokens or ∼3,000 word-types of annotated data. These tokens or word-types were used to build a tag dictionary. They employed label propagation for expanding the coverage of this dictionary in a similar vein to Das and Petrov (2011), but they also used an external dictionary. They built training examples using the combined dictionary and then trained the tagger on this data. They achieved 81.9% and 81.2% accuracy for Kinyarwanda and Malagasy respectively. Note that their usage of an external dictionary compromises their claim of using only 4 hours of annotation.

[1] http://www.wiktionary.org/

	da	nl	de	el	it	pt	es	sv	Average
Das and Petrov (2011)	83.2	79.5	82.8	82.5	86.8	87.9	84.2	80.5	83.4
Duong et al. (2013b)	85.6	84.0	85.4	80.4	81.4	86.3	83.3	81.0	83.4
Li et al. (2012)	83.3	86.3	85.4	79.2	86.5	84.5	86.4	86.1	84.8
Täckström et al. (2013)	88.2	85.9	90.5	89.5	89.3	91.0	87.1	88.9	88.8

Table 1: Previously published token-level POS tagging accuracy for various models across 8 languages — Danish (da), Dutch (nl), German (de), Greek (el), Italian (it), Portuguese (pt), Spanish (es), Swedish (sv) — evaluated on CoNLL data (Buchholz and Marsi, 2006).

The method we propose in this paper is similar in only using a small amount of annotation. However, we directly use the annotated data to train the model rather than using a dictionary. We argue that with a proper "guide", we can take advantage of very limited annotated data.

2.1 Annotated data

Our annotated data mainly comes from CoNLL shared tasks on dependency parsing (Buchholz and Marsi, 2006). The language specific tagsets are mapped into the universal tagset. We will use this annotated data mainly for evaluation. Table 2 shows the size of annotated data for each language. The 8 languages we are considering in this experiment are not actually resource-poor languages. However, running on these 8 languages makes our system comparable with previously proposed methods. Nevertheless, we try to use as few resources as possible, in order to simulate the situation for resource-poor languages. Later in Section 6 we adapt the approach for Malagasy, a truly resource-poor language.

2.2 Universal tagset

We employ the universal tagset from (Petrov et al., 2012) for our experiment. It consists of 12 common tags: *NOUN, VERB, ADJ* (adjective), *ADV* (adverb), *PRON* (pronoun), *DET* (determiner and article), *ADP* (preposition and postposition), *CONJ* (conjunctions), *NUM* (numerical), *PRT* (particle), *PUNC* (punctuation) and *X* (all other categories including foreign words and abbreviations). Petrov et al. (2012) provide the mapping from each language-specific tagset to the universal tagset.

The idea of using the universal tagset is of great use in multilingual applications, enabling comparison across languages. However, the mapping is not always straightforward. Table 2 shows the size of the annotated data for each language, the number of tags presented in the data, and the list of tags that are not matched. We can see that only 8 tags are presented in the annotated data for Danish, i.e, 4 tags (*DET, PRT, PUNC,* and *NUM*) are missing.[2] Thus, a classifier using all 12 tags will be heavily penalized in the evaluation.

Li et al. (2012) considered this problem and tried to manually modify the Danish mappings. Moreover, *PRT* is not really a universal tag since it only appears in 3 out of the 8 languages. Plank et al. (2014) pointed out that *PRT* often gets confused with *ADP* even in English. We will later show that the mapping problem causes substantial degradation in the performance of a POS tagger exploiting parallel data. The method we present here is more target-language oriented: our model is trained on the target language, in this way, only relevant information from the source language is retained. Thus, we automatically correct the mapping, and other incompatibilities arising from incorrect alignments and syntactic divergence between the source and target languages.

Lang	Size(k)	# Tags	Not Matched
da	94	8	DET, PRT, PUNC, NUM
nl	203	11	PRT
de	712	12	
el	70	12	
it	76	11	PRT
pt	207	11	PRT
es	89	11	PRT
sv	191	11	DET
AVG	205		

Table 2: The size of annotated data from CoNLL (Buchholz and Marsi, 2006), and the number of tags included and missing for 8 languages.

[2]Many of these are mistakes in the mapping, however, they are indicative of the kinds of issues expected in low-resource languages.

3 Directly Projected Model (DPM)

In this section we describe a maximum entropy tagger that only uses information from directly projected data.

3.1 Parallel data

We first collect Europarl data having English as the source language, an average of 1.85 million parallel sentences for each of the 8 language pairs. In terms of parallel data, we use far less data compared with other recent work. Das and Petrov (2011) used Europarl and the ODS United Nation dataset, while Täckström et al. (2013) additionally used parallel data crawled from the web. The amount of parallel data is crucial for alignment quality. Since DPM uses alignments to transfer tags from source to target language, the performance of DPM (and other models that exploit projection) largely depends on the quantity of parallel data. The "No LP" model of Das and Petrov (2011), which only uses directly projected labels (without label propagation), scored 81.3% for 8 languages. However, using the same model but with more parallel data, Täckström et al. (2013) scored 84.9% on the same test set.

3.2 Label projection

We use the standard alignment tool Giza++ (Och and Ney, 2003) to word align the parallel data. We employ the Stanford POS tagger (Toutanova et al., 2003) to tag the English side of the parallel data and then project the label to the target side. It has been confirmed in many studies (Täckström et al., 2013, Das and Petrov, 2011, Toutanova and Johnson, 2008) that directly projected labels are noisy. Thus we need a method to reduce the noise. We employ the strategy of Yarowsky and Ngai (2001) of ranking sentences using a their alignment scores from IBM model 3.

Firstly, we want to know how noisy the projected data is. Thus, we use the test data to build a simple supervised POS tagger using the TnT tagger (Brants, 2000) which employs a second-order Hidden Markov Model (HMM). We tag the projected data and compare the label from direct projection and from the TnT tagger. The labels from the TnT Tagger are considered as pseudo-gold labels. Column "Without Mapping" from Table 3 shows the average accuracy for the first n-sentences ($n = 60k, 100k, 200k, 500k$) for 8 languages according to the ranking. Column "Coverage" shows the percentages of projected label (the other tokens are Null aligned). We can see that when we select more data, both coverage and accuracy fall. In other words, using the sentence alignment score, we can rank sentences with high coverage and accuracy first. However, even after ranking, the accuracy of projected labels is less than 80% demonstrating how noisy the projected labels are.

Table 3 (column "With Mapping") additionally shows the accuracy using simple tagset mapping, i.e. mapping each tag to the tag it is assigned most frequently in the test data. For example *DET, PRT, PUNC, NUM*, missing from Danish gold data, will be matched to *PRON, X, X, ADJ* respectively. This simple matching yields a $\sim$ 4% (absolute) improvement in average accuracy. This illustrates the importance of handling tagset mapping carefully.

3.3 The model

In this section, we introduce a maximum entropy tagger exploiting the projected data. We select the first 200k sentences from Table 3 for this experiment. This number represents a trade-off between size and accuracy. More sentences provide more information but at the cost of noisier data. Duong et al. (2013b) also used sentence alignment scores to rank sentences. Their model stabilizes after using 200k sentences. We conclude that 200k sentences is enough and capture most information from the parallel data.

Features	Descriptions
W@-1	Previous word
W@+1	Next word
W@0	Current word
CAP	First character is capitalized
NUMBER	Is number
PUNCT	Is punctuation
SUFFIX@k	Suffix up to length 3 ($k <= 3$)
WC	Word class

Table 4: Feature template for a maximum entropy tagger

We ignore tokens that don't have labels, which arise from null alignments and constitute approximately 14% of the data. The remaining data ($\sim$1.4 million tokens) are used to train a maximum entropy (MaxEnt) model. MaxEnt is one of the simplest forms of probabilistic classifier, and is appropriate in this setting due to the incomplete

Data Size (k)	Coverage (%)	Without Mapping	With Mapping
60	91.5	79.9	84.2
100	89.1	79.4	83.6
200	86.1	79.1	82.9
500	82.4	78.0	81.5

Table 3: The coverage, and POS tagging accuracy with and without tagset mapping of directly projected labels, averaged over 8 languages for different data sizes

Model	da	nl	de	el	it	pt	es	sv	Avg
All features	64.4	83.3	86.3	79.7	82.0	86.5	82.5	76.5	80.2
- Word Class	64.7	82.6	86.6	79.0	82.8	84.6	82.2	76.9	79.9
- Suffix	64.0	82.8	86.3	78.1	81.0	85.9	82.3	76.2	79.6
- Prev, Next Word	62.6	82.5	87.4	79.0	81.9	86.5	82.2	74.8	79.6
- Cap, Num, Punct	64.0	81.9	84.0	78.0	79.1	86.3	81.8	75.6	78.8

Table 5: The accuracy of Directed Project Model (DPM) with different feature sets, removing one feature set at a time

sequence data. While sequence models such as HMMs or CRFs can provide more accurate models of label sequences, they impose a more stringent training requirement.[3] We also experimented with a first-order linear chain CRF trained on contiguous sub-sequences but observed $\sim 4\%$ (absolute) drop in performance.

The maximum entropy classifier estimates the probability of tag t given a word w as

$$P(t|w) = \frac{1}{Z(w)} \exp \sum_{j=1}^{D} \lambda_j f_j(w,t)\,,$$

where $Z(w) = \sum_t \exp \sum_{j=1}^{D} \lambda_j f_j(w,t)$ is the normalization factor to ensure the probabilities $P(t|w)$ sum to one. Here f_j is a feature function and λ_j is the weight for this feature, learned as part of training. We use Maximum A Posteriori (MAP) estimation to maximize the log likelihood of the training data, $\mathcal{D} = \{w_i, t_i\}_{i=1}^{N}$, subject to a zero-mean Gaussian regularisation term,

$$\begin{aligned}\mathcal{L} &= \log P(\Lambda) \prod_{i=1}^{N} P(t^{(i)}|w^{(i)}) \\ &= -\sum_{j=1}^{D} \frac{\lambda_j^2}{2\delta^2} + \sum_{i=1}^{N} \sum_{j=1}^{D} \lambda_j f_j(w_i, t_i) - \log Z(w_i)\end{aligned}$$

where the regularisation term limits over-fitting, an important concern when using large feature sets. For our experiments we set $\delta^2 = 1$. We use L-BFGS which performs gradient ascent to maximize $\mathcal{L}$. Table 4 shows the features we considered for building the DPM. We use $mkcls$, an unsupervised method for word class induction which is widely used in machine translation (Och, 1999). We run $mkcls$ to obtain 100 word classes, using only the target language side of the parallel data.

Table 5 shows the accuracy of the DPM evaluated on 8 languages ("All features model"). DPM performs poorly on Danish, probably because of the tagset mapping issue discussed above. The DPM result of 80.2% accuracy is encouraging, particularly because the model had no explicit supervision.

To see what features are meaningful for our model, we remove features in turn and report the result. The result in Table 5 disagrees with Täckström et al. (2013) on the word class features. They reported a gain of approximately 3% (absolute) using the word class. However, it seems to us that these features are not especially meaningful (at least in the present setting). Possible reasons for the discrepancy are that they train the word class model on a massive quantity of external monolingual data, or their algorithms for word clustering are better (Uszkoreit and Brants, 2008). We can see that the most informative features are Capitalization, Number and Punctuation. This makes sense because in languages such as German, capitalization is a strong indicator of *NOUN*. Number and punctuation features ensure that we classify *NUM* and *PUNCT* tags correctly.

[3] Täckström et al. (2013) train a CRF on incomplete data, using a tag dictionary heuristic to define a 'gold standard' lattice over label sequences.

4 Correction Model

In this section we incorporate the directly projected model into a second *correction* model trained on a small supervised sample of 1,000 annotated tokens. Our DPM model is not very accurate; as we have discussed it makes many errors, due to invalid or inconsistent tag mappings, noisy alignments, and cross-linguistic syntactic divergence. However, our aim is to see how effectively we can exploit the strengths of the DPM model while correcting for its inadequacies using direct supervision. We select only 1,000 annotated tokens to reflect a low resource scenario. A small supervised training sample is a more realistic form of supervision than a tag dictionary (noisy or otherwise). Although used in most prior work, a tag dictionary for a new language requires significant manual effort to construct. Garrette and Baldridge (2013) showed that a 1,000 token dataset could be collected very cheaply, requiring less than 2 hours of non-expert time.

Our correction model makes use of a *minimum divergence* (MD) model (Berger et al., 1996), a variant of the maximum entropy model which biases the target distribution to be similar to a static reference distribution. The method has been used in several language applications including machine translation (Foster, 2000) and parsing (Plank and van Noord, 2008, Johnson and Riezler, 2000). These previous approaches have used various sources of reference distribution, e.g., incorporating information from a simpler model (Johnson and Riezler, 2000) or combining in- and out-of-domain models (Plank and van Noord, 2008). Plank and van Noord (2008) concluded that this method for adding prior knowledge only works with high quality reference distributions, otherwise performance suffers.

In contrast to these previous approaches, we consider the specific setting where both the learned model and the reference model $s_o = P(t|w)$ are both maximum entropy models. In this case we show that the MD setup can be simplified to a regularization term, namely a Gaussian prior with a non-zero mean. We model the classification probability, $P'(t|w)$ as the product between a base model and a maximum entropy classifier,

$$P'(t|w) \propto P(t|w) \exp \sum_{j=1}^{D} \gamma_j f_j(w,t)$$

where here we use the DPM model as base model $P(t|w)$. Under this setup, where P' uses the same features as P, and both are log-linear models, this simplifies to

$$\begin{aligned} P'(t|w) &\propto \exp\left(\sum_{j=1}^{D} \lambda_j f_j(w,t) + \sum_{j=1}^{D} \gamma_j f_j(w,t)\right) \\ &\propto \exp \sum_{j=1}^{D} (\lambda_j + \gamma_j) f_j(w,t) \end{aligned} \quad (1)$$

where the constant of proportionality is $Z'(w) = \sum_t \exp \sum_{j=1}^{D} (\lambda_j + \gamma_j) f_j(w,t)$. It is clear that Equation (1) also defines a maximum entropy classifier, with parameters $\alpha_j = \lambda_j + \gamma_j$, and consequently this might seem to be a pointless exercise. The utility of this approach arises from the prior: MAP training with a zero mean Gaussian prior over γ is equivalent to a Gaussian prior over the aggregate weights, $\alpha_j \sim \mathcal{N}(\lambda_j, \sigma^2)$. This prior enforces parameter sharing between the two models by penalising parameter divergence from the underlying DPM model λ. The resulting training objective is

$$\mathcal{L}^{\text{corr}} = \log P(\mathbf{t}|\mathbf{w}, \alpha) - \frac{1}{2\sigma^2} \sum_{j=1}^{D} (\alpha_j - \lambda_j)^2$$

which can be easily optimised using standard gradient-based methods, e.g., L-BFGS. The contribution of the regulariser is scaled by the constant $\frac{1}{2\sigma^2}$.

4.1 Regulariser sensitivity

Careful tuning of the regularisation term σ^2 is critical for the correction model, both to limit overfitting on the very small training sample of 1,000 tokens, and to control the extent of the influence of the DPM model over the correction model. A larger value of σ^2 lessens the reliance on the DPM and allows for more flexible modelling of the training set, while a small value of σ^2 forces the parameters to be close to the DPM estimates at the expense of data fit. We expect the best value to be somewhere between these extremes, and use line-search to find the optimal value for σ^2. For this purpose, we hold out 100 tokens from the 1,000 instance training set, for use as our development set for hyper-parameter selection.

From Figure 1, we can see that the model performs poorly on small values of σ^2. This is understandable because the small σ^2 makes the model

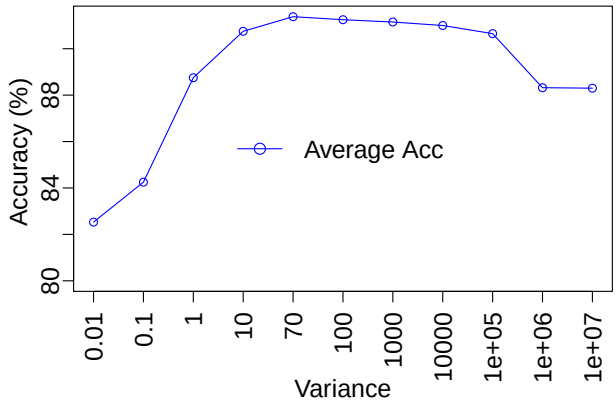


Figure 1: Sensitivity of regularisation parameter σ^2 against the average accuracy measured on 8 languages on the development set

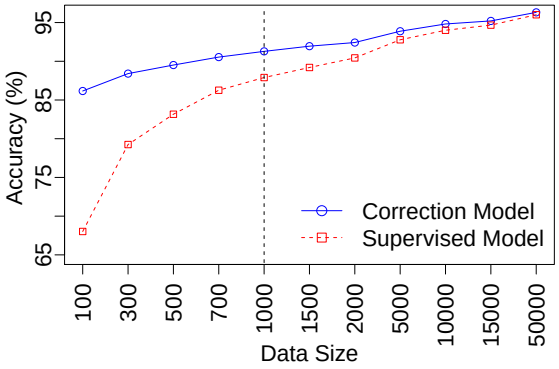


Figure 2: Learning curve for correction model and supervised model: the x-axis is the size of data (number of tokens); the y-axis is the average accuracy measured on 8 languages; the dashed line shows the data condition reported in Table 6

too similar to DPM, which is not very accurate (80.2%). At the other extreme, if σ^2 is large, the DPM model is ignored, and the correction model is equivalent with the supervised model ($\sim$ 88% accuracy). We select the value of $\sigma^2 = 70$, which maximizes the accuracy on the development set.

4.2 The model

Using the value of $\sigma^2 = 70$, we retrain the model on the whole 1,000-token training set and evaluate the model on the rest of the annotated data. Table 6 shows the performance of DPM, Supervised model, Correction model and the state-of-the-art model (Täckström et al., 2013). The supervised model trains a maximum entropy tagger using the same features as in Table 4 on this 1000 tokens. The only difference between the supervised model and the correction model is that in the correction model we additionally incorporate DPM as the prior.

The supervised model performs surprisingly well confirming that our features are meaningful in distinguishing between tags. This model achieves high accuracy on Danish compared with other languages probably because Danish is easier to learn since it contains only 8 tags. Despite the fact that the DPM is not very accurate, the correction model consistently outperforms the supervised model on all considered languages, approximately 4.3% (absolute) better on average. This shows that our method of incorporating DPM to the model is efficient and robust.

The correction model performs much better than the state-of-the-art for 7 languages but slightly worse for 1 language. On average we achieve 91.3% accuracy compared with 88.8% for the state-of-the-art, an error rate reduction of 22.3%. This is despite using fewer resources and only modest supervision.

5 Analysis

Tagset mismatch In the correction model, we implicitly resolve the mismatched tagset issue. DPM might contain tags that don't appear in the target language or generally are errors in the mapping. However, when incorporating DPM into the correction model, only the feature weight of tags that appear in the target language are retained. In general, because we don't explicitly do any mapping between languages, we might have trouble if the tagset size of the target language is bigger than the source language tagset. However, this is not the case for our experiment because we choose English as the source-side and English has the full 12 tags.

Learning curve We investigate the impact of the number of available annotated tokens on the correction model. Figure 2 shows the learning curve of the correction model and the supervised model. We can clearly see the differences between 2 models when the size of training data is small. For example, at 100 tokens, the difference is very large, approximately 18% (absolute), it is also 6% (absolute) better than DPM. This difference diminishes as we add more data. This make sense because when we add more data, the supervised model become stronger, while the effective-

Model	da	nl	de	el	it	pt	es	sv	Avg
DPM	64.4	83.3	86.3	79.7	82.0	86.5	82.5	76.5	80.2
Täckström et al. (2013)	88.2	85.9	90.5	89.5	89.3	91.0	87.1	**88.9**	88.8
Supervised model	90.1	84.6	89.6	88.2	81.4	87.6	88.9	85.4	87.0
Correction Model	**92.1**	**91.1**	**92.5**	**92.1**	**89.9**	**92.5**	**91.6**	88.7	**91.3**
DPM (with dict)	65.2	83.9	87.0	79.1	83.5	87.1	83.0	77.5	80.8
Correction Model (with dict)	93.3	92.2	93.7	93.2	92.2	93.1	92.8	90.0	92.6

Table 6: The comparison of our Directly Projected Model, Supervised Model, Correction Model and the state-of-the-art system (Täckström et al., 2013). The best performance for each language is shown in bold. The models that are built with a dictionary are provided for reference.

ness of the DPM prior on the correction model is wearing off. An interesting observation is that the correction model is always better, even when we add massive amounts of annotated data. At 50,000 tokens, when the supervised model reaches 96% accuracy, the correction model is still 0.3% (absolute) better, reaching 96.3%. It means that even at that high level of confidence, some information can still be added from DPM to the correction model. This improvement probably comes from the observation that the ambiguity in one language is explained through the alignment. It also suggests that this method could improve the performance of a supervised POS tagger even for resource-rich languages.

Our methods are also relevant for annotation projects for resource-poor languages. Assuming that it is very costly to annotate even 100 tokens, applying our methods can save annotation effort but maintain high performance. For example, we just need 100 tokens to match the accuracy of a supervised method trained on 700 tokens, or we just need 500 tokens to match the performance with nearly 2,000 tokens of supervised learning.

Our method is simple, but particularly suitable for resource-poor languages. We need a small amount of annotated data for a high performance POS tagger. For example, we need only around 300 annotated tokens to reach the same accuracy as the state-of-the-art unsupervised POS tagger (88.8%).

Tag dictionary Although, it is not our objective to rely on the dictionary, we are interested in whether the gains from the correction model still persist when the DPM performance is improved. We attempt to improve DPM, following the method of Li et al. (2012) by building a tag dictionary using Wiktionary. This dictionary is then used as a feature which fires for word-tag pairings present in the dictionary. We expect that when we add this additional supervision, the DPM model should perform better. Table 6 shows the performance of DPM and the correction model when incorporating the dictionary. The DPM model only increases 0.6% absolute but the correction model increases 1.3%. Additionally, it shows that our model can improve further by incorporating external information where available.

CRF Our approach of using simple classifiers begs the question of whether better results could be obtained using sequence models, such as conditional random fields (CRFs). As mentioned previously, a CRF is not well suited for incomplete data. However, as our second 'correction' model is trained on complete sequences, we now consider using a CRF in this stage. The training algorithm is as follows: first we estimate the DPM feature weights on the incomplete data as before, and next we incorporate the feature weights into a CRF trained on the 1,000 annotated tokens. This is complicated by the different feature sets between the MaxEnt classifier and the CRF, however the classifier uses a strict subset of the CRF features. Thus, we use the minimum divergence prior for the token level features, and a standard zero-mean prior for the sequence features. That is, the objective function of the CRF correction model becomes:

$$\mathcal{L}_{\text{crf}}^{\text{corr}} = \log P(\mathbf{t}|\mathbf{w}, \alpha) - \frac{1}{2\delta_1^2}\sum_{j\in F_1}(\alpha_j - \lambda_j)^2 - \frac{1}{2\delta_2^2}\sum_{j\in F_2}\alpha_j^2 \quad (2)$$

where F_1 is the set of features referring to only one label as in the DPM maxent model and F_2 is the set of features over label pairs. The union of $F = F_1 \cup F_2$ is the set of all features for the CRF. We perform grid search using held out

data as before for δ_1^2 and δ_2^2. The CRF correction model scores 88.1% compared with 86.5% of the supervised CRF model trained on the 1,000 tokens. Clearly, this is beneficial, however, the CRF correction model still performs worse than the MaxEnt correction model (91.3%). We are not sure why but one reason might be overfitting of the CRF, due to its large feature set and tiny training sample. Moreover, this CRF approach is orthogonal to Täckström et al. (2013): we could use their CRF model as the DPM model and train the CRF correction model using the same minimum divergence method, presumably resulting in even higher performance.

6 Two-output model

Garrette and Baldridge (2013) also use only a small amount of annotated data, evaluating on two resource-poor languages Kinyarwanda (KIN) and Malagasy (MLG). As a simple baseline, we trained a maxent supervised classifier on this data, achieving competitive results of 76.4% and 80.0% accuracy compared with their published results of 81.9% and 81.2% for KIN and MLG, respectively. Note that the Garrette and Baldridge (2013) method is much more complicated than this baseline, and additionally uses an external dictionary.

We want to further improve the accuracy of MLG using parallel data. Applying the technique from Section 4 will not work directly, due to the tagset mismatch (the Malagasy tagset contains 24 tags) which results in highly different feature sets. Moreover, we don't have the language expertise to manually map the tagset. Thus, in this section, we propose a method capable of handling tagset mismatch. For data, we use a parallel English-Malagasy corpus of $\sim$100k sentences,[4] and the POS annotated dataset developed by Garrette and Baldridge (2013), which comprises 4230 tokens for training and 5300 tokens for testing.

6.1 The model

Traditionally, MaxEnt classifiers are trained using a single label.[5] The method we propose is trained with pairs of output labels: one for the Malagasy tag (t_M) and one for the universal tag (t_U), which are both predicted conditioned on a Malagasy word (w_M) in context. Our two-output model is defined as

$$P(t_M, t_U|w_M) = \frac{1}{Z(w_M)} \exp\Bigg(\sum_{j=1}^{D} \lambda_j f_j^M(w, t_M) + \sum_{j=1}^{E} \gamma_j f_j^U(w, t_U) + \sum_{j=1}^{F} \alpha_j f_j^B(w, t_M, t_U)\Bigg) \quad (3)$$

where f^M, f^U, f^B are the feature functions considering t_M only, t_U only, and over both outputs t_M and t_U respectively, and $Z(w_M)$ is the partition function. We can think of Eq. (3) as the combination of 3 models: the Malagasy maxent supervised model, the DPM model, and the tagset mapping model. The central idea behind this model is to learn to predict not just the MLG tags, as in a standard supervised model, but also to learn the mapping between MLG and the noisy projected universal tags. Framing this as a two output model allows for information to flow both ways, such that confident taggings in either space can inform the other, and accordingly the mapping weights α are optimised to maximally exploit this effect.

One important question is how to obtain labelled data for training the two-output model, as our small supervised sample of MLG text is only annotated for MLG labels t_M. We resolve this by first learning the DPM model on the projected labels, after which we automatically label our correction training set with predicted tags from the DPM model. That is, we augment the annotated training data from (t_M, w_M) to become (t_M, t_U, w_M). This is then used to train the two-output maxent classifier, optimising a MAP objective using standard gradient descent. Note that it would be possible to apply the same minimum divergence technique for the two-output maxent model. In this case the correction model would include a regularization term over the λ to bias towards the DPM parameters, while γ and α would use a zero-mean regularizer. However, we leave this for future work.

Table 7 summarises the performance of the state-of-the-art (Garrette et al., 2013), the supervised model and the two-output maxent model evaluated on the Malagasy test set. The two-output maxent model performs much better than the supervised model, achieving $\sim$5.3% (absolute) im-

[4] http://www.ark.cs.cmu.edu/global-voices/

[5] Or else a sequence of labels, in the case of a conditional random field (Lafferty et al., 2001). However, even in this case, each token is usually assigned a single label. An exception is the factorial CRF (Sutton et al., 2007), which models several co-dependent sequences. Our approach is equivalent to a factorial CRF without edges between tags for adjacent tokens in the input.

Model	Accuracy (%)
Garrette et al. (2013)	81.2
MaxEnt Supervised	80.0
2-output MaxEnt (Universal tagset)	85.3
2-output MaxEnt (Penn tagset)	85.6

Table 7: The performance of different models for Malagasy.

provement. An interesting property of this approach is that we can use different tagsets for the DPM. We also tried the original Penn treebank tagset which is much larger than the universal tagset (48 vs. 12 tags). We observed a small improvement reaching 85.6%, suggesting that some pertinent information is lost in the universal tagset. All in all, this is a substantial improvement over the state-of-the-art result of 81.2% (Garrette et al., 2013) and an error reduction of 23.4%.

7 Conclusion

In this paper, we thoroughly review the work on multilingual POS tagging of the past decade. We propose a simple method for building a POS tagger for resource-poor languages by taking advantage of parallel data and a small amount of annotated data. Our method also efficiently resolves the tagset mismatch issue identified for some language pairs. We carefully choose and tune the model. Comparing with the state-of-the-art, we are using the more realistic assumption that a small amount of labelled data can be made available rather than requiring a crowd-sourced dictionary. We use less parallel data which as we pointed out in section 3.1, could have been a huge disadvantage for us. Moreover, we did not exploit any external monolingual data. Importantly, our method is simpler but performs better than previously proposed methods. With only 1,000 annotated tokens, less than 1% of the test data, we can achieve an average accuracy of 91.3% compared with 88.8% of the state-of-the-art (error reduction rate $\sim$22%). Across the 8 languages we are substantially better at 7 and slightly worse at one. Our method is reliable and could even be used to improve the performance of a supervised POS tagger.

Currently, we are building the tagger and evaluating through several layers of mapping. Each layer might introduce some noise which accumulates and leads to a biased model. Moreover, the tagset mappings are not available for many resource-poor languages. We therefore also proposed a method to automatically match between tagsets based on a two-output maximum entropy model. On the resource-poor language Malagasy, we achieved the accuracy of 85.6% compared with the state-of-the-art of 81.2% (Garrette et al., 2013). Unlike their method, we didn't use an external dictionary but instead use a small amount of parallel data.

In future work, we would like to improve the performance of DPM by collecting more parallel data. Duong et al. (2013a) pointed out that using a different source language can greatly alter the performance of the target language POS tagger. We would like to experiment with different source languages other than English. We assume that we have 1,000 tokens for each language. Thus, for the 8 languages we considered we will have 8,000 annotated tokens. Currently, we treat each language independently, however, it might also be interesting to find some way to incorporate information from multiple languages simultaneously to build the tagger for a single target language.

Acknowledgments

We would like to thank Dan Garreette, Jason Baldridge and Noah Smith for Malagasy and Kinyarwanda datasets. This work was supported by the University of Melbourne and National ICT Australia (NICTA). NICTA is funded by the Australian Federal and Victoria State Governments, and the Australian Research Council through the ICT Centre of Excellence program. Dr Cohn is the recipient of an Australian Research Council Future Fellowship (project number FT130101105).

References

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceeding of HLT-NAACL*, pages 582–590.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *COMPUTATIONAL LINGUISTICS*, 22:39–71.

Thorsten Brants. 2000. TnT: A statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing (ANLP '00)*, pages 224–231, Seattle, Washington, USA.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised pos induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 575–584.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 600–609.

Long Duong, Paul Cook, Steven Bird, and Pavel Pecina. 2013a. Increasing the quality and quantity of source language data for Unsupervised Cross-Lingual POS tagging. Proceedings of the Sixth International Joint Conference on Natural Language Processing, pages 1243–1249. Asian Federation of Natural Language Processing.

Long Duong, Paul Cook, Steven Bird, and Pavel Pecina. 2013b. Simpler unsupervised POS tagging with bilingual projections. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 634–639. Association for Computational Linguistics.

George Foster. 2000. A maximum entropy/minimum divergence translation model. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 45–52.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. pages 138–147, June.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of pos-taggers for low-resource languages. pages 583–592, August.

Yoav Goldberg, Meni Adler, and Michael Elhadad. 2008. Em can find pretty good hmm pos-taggers (when given a good start. In *In Proc. ACL*, pages 746–754.

Jiri Hana, Anna Feldman, and Chris Brew. 2004. A resource-light approach to Russian morphology: Tagging Russian using Czech resources. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP '04)*, pages 222–229, Barcelona, Spain, July.

Mark Johnson and Stefan Riezler. 2000. Exploiting auxiliary distributions in stochastic unification-based grammars. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 154–161.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the Tenth Machine Translation Summit (MT Summit X)*, pages 79–86, Phuket, Thailand. AAMT.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289.

Shen Li, João V. Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1389–1398.

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

Franz Josef Och. 1999. An efficient method for determining bilingual word classes. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, EACL '99, pages 71–76.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Barbara Plank and Gertjan van Noord. 2008. Exploring an auxiliary distribution based approach to domain adaptation of a syntactic disambiguation model. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 9–16.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014. Learning part-of-speech taggers with inter-annotator agreement loss. In *Proceedings of the 14th Conference of the European Chapter of the Association for*

Computational Linguistics, pages 742–751, Gothenburg, Sweden, April.

Siva Reddy and Serge Sharoff. 2011. Cross language POS taggers (and other tools) for Indian languages: An experiment with Kannada using Telugu resources. In *Proceedings of IJCNLP workshop on Cross Lingual Information Access: Computational Linguistics and the Information Need of Multilingual Societies. (CLIA 2011 at IJNCLP 2011)*, Chiang Mai, Thailand, November.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *J. Mach. Learn. Res.*, 8:693–723, May.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

Kristina Toutanova and Mark Johnson. 2008. A bayesian lda-based model for semi-supervised part-of-speech tagging. In J.C. Platt, D. Koller, and Y. Singer a nd S.T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1521–1528. Curran Associates, Inc.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1 (NAACL '03)*, pages 173–180, Edmonton, Canada.

Jakob Uszkoreit and Thorsten Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *In ACL International Conference Proceedings*.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, NAACL '01, pages 1–8.

An Experimental Comparison of Active Learning Strategies for Partially Labeled Sequences

Diego Marcheggiani
Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
Pisa, Italy
diego.marcheggiani@isti.cnr.it

Thierry Artières
LIP6
Pierre et Marie Curie University
Paris, France
thierry.artieres@lip6.fr

Abstract

Active learning (AL) consists of asking human annotators to annotate automatically selected data that are assumed to bring the most benefit in the creation of a classifier. AL allows to learn accurate systems with much less annotated data than what is required by pure supervised learning algorithms, hence limiting the tedious effort of annotating a large collection of data.

We experimentally investigate the behavior of several AL strategies for sequence labeling tasks (in a *partially-labeled* scenario) tailored on Partially-Labeled Conditional Random Fields, on four sequence labeling tasks: phrase chunking, part-of-speech tagging, named-entity recognition, and bio-entity recognition.

1 Introduction

Today, the state-of-the-art methods in most natural language processing tasks are supervised machine learning approaches. Their main problem lies in their need of large human-annotated training corpus, which requires a tedious and expensive work from domain experts. The process of *active learning* (AL) employs one or more human annotators by asking them to label new samples which are supposed to be the most informative in the creation of a new classifier. A classifier is incrementally retrained with all the data labeled by the annotator. AL has been demonstrated to work well and to produce accurate classifiers while saving much human annotation effort. One critical issue is to define a measure of the informativeness which should reflect how much new information a new example would give in the learning of a new classifier once annotated.

A lot of work has been done on the AL field in the past years (see (Settles, 2012) for an exhaustive overview). In particular, AL proved its usefulness in sequence labeling tasks (Settles and Craven, 2008). Yet, researchers have always adopted as annotation unit an entire sequence (i.e., the annotator is asked to annotate the whole sequence) while it looks like it could be much more relevant to ask for labeling only small parts of it (e.g., the ones with highest ambiguity). A few works have investigated this idea. For instance, Wanvarie et al. (2011) proposed to use Partially-Labeled Conditional Random Fields (PL-CRFs) (Tsuboi et al., 2008), a semi-supervised variation of Conditional Random Fields (CRFs) (Lafferty et al., 2001) able to deal with partially-labeled sequences, thus enabling to adopt as annotation unit single tokens and still learning from full sequences. AL with partially labeled sequences has proven to be effective in substantially reducing the amount of annotated data with respect to common AL approaches (see (Wanvarie et al., 2011)).

In this work we focus on AL strategies for partially labeled sequences adopting the single token as annotation unit and PL-CRFs as learning algorithm given its nature in dealing with partially labeled sequences. We propose several AL strategies based on measures of *uncertainty* adapted for the AL with partially labeled sequences scenario and tailored on PL-CRFs. We further propose two strategies that exploit the finer granularity given by the partially-labeled scenario. We also show that the choice of single-token annotation can bring to unpredictable results on sequence labeling tasks in which the structure of the sequences is not regular, e.g., named-entity recognition. We propose a first solution to the problem of unpredictability. The aim of this work is thoroughly compare the effectiveness and the behavior of all the proposed AL strategies on four standard sequence labeling tasks, phrase chunking, part-of-speech tagging, named-entity recognition and bio-entity recognition.

The remainder of this paper is as follows. In Section 2 we summarize the related work in AL, in Section 3 we describe PL-CRFs, the semi-supervised algorithm we adopt in this work. Section 4 describes in details the AL framework and the AL strategies we propose. Section 5 provides a description of the experimental setting, the datasets, and discusses the empirical results. Section 6 summarizes our findings.

2 Related Work

Our work belongs to the pool-based AL framework. It considers the case in which a large amount (pool) of unlabeled examples is available, from which samples to be labeled must be chosen. This framework fits all the sequence labeling problems we consider here. For a more exhaustive survey on other AL frameworks see

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 898–906,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

(Settles, 2012).

Most of the AL works on sequence labeling adopted the entire sequence as annotation unit (Settles and Craven, 2008) which was demonstrated by Wanvarie et al. (2011) to be less effective than using the single token as annotation unit. The main AL works in this latter line of work are (Shen et al., 2004), (Tomanek and Hahn, 2009) and (Wanvarie et al., 2011). Shen et al. (2004) adopted SVMs as learning algorithm and proposed two strategies that combine three criteria, informativeness, representativeness and diversity. SVMs allowed them to use as annotation unit a subset of the tokens in a sequence, without annotating, in any way, the rest of the tokens in the sequence. In (Tomanek and Hahn, 2009), the most uncertain tokens of the sequence are singularly annotated, but the rest of the labels in the sequence are then chosen by the classifier in a semi-supervised fashion. Wanvarie et al. (2011) is the closest work to ours, they adopt a minimum confidence selection strategy with re-estimation using the PL-CRFs. Differently from our work, Wanvarie et al. (2011) show that adopting the AL with partially labeled sequences using re-estimation, the annotation cost can be dramatically reduced (by annotating from 8% to 10% of the tokens of the entire training set), obtaining the same level of performance of the classifier trained on the entire, fully-labeled, training set. We started our work from this conclusion and we focused on AL with partially labeled sequences using re-estimation by comparing several AL strategies in order to find the strategy that allows to create the best classifier with the minimum annotation effort.

3 Partially-Labeled Conditional Random Fields

Nowadays, CRFs are the *de-facto* standard for the solution of sequence labeling tasks (Sarawagi, 2008). In traditional CRFs (Lafferty et al., 2001) the conditional probability of a sequence of labels $\mathbf{y}$ given a sequence of observed feature vectors $\mathbf{x}$ is given by:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Psi_t(\mathbf{y}, \mathbf{x}) \tag{1}$$

where a standard choice for sequence labeling tasks are the so called Linear-chain CRFs:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Psi_t(y_t, y_{t-1}, \mathbf{x}_t) \tag{2}$$

with:

$$\Psi_t(y_t, y_{t-1}, \mathbf{x}_t) = \Psi_u(y_t, x_t)\Psi_b(y_t, y_{t-1}) \tag{3}$$

where $\Psi_u(y_t, x_t)$ models the co-occurrence between features x_t, and label y_t at time t, and $\Psi_b(y_t, y_{t-1})$ models the co-occurrence between two adjacent labels y_t and y_{t-1}.

PL-CRFs introduced by Tsuboi et al. (2008) allow to learn a CRF model using partially-labeled sequences, marginalizing on those tokens that do not have an assigned label. In PL-CRFs, $\mathbf{L}$ denotes a partially labeled information about a sequence. It consists of a sequence of sets L_t in which $L_t = \mathcal{Y}$ (where $\mathcal{Y}$ is the set of all the possible labels) if there is no label information for token at time t. L_t is a singleton containing y_t if the label of the token at time t is known, and $\mathbf{Y_L}$ is the set of label sequences that fits the partial label information $\mathbf{L}$. Then the probability of a partial labeling may be computed as:

$$p(\mathbf{Y_L}|\mathbf{x}) = \sum_{\mathbf{y} \in \mathbf{Y_L}} p(\mathbf{y}|\mathbf{x}) \tag{4}$$

In order to perform inference and parameter learning on PL-CRFs, some modifications on traditional CRFs inference algorithms are required.

3.1 Forward-Backward Algorithm

Differently from traditional CRFs, the forward and backward scores (respectively α and β), are calculated as follows:

$$\alpha_{t,\mathbf{L}}(j) = \begin{cases} 0 & \text{if } j \notin L_t \\ \Psi_1(j, y_0, x_1) & \text{else if } t = 1 \\ & \text{and } j \in L_t \\ SA(j) & \text{otherwise} \end{cases} \tag{5}$$

$$\beta_{t,\mathbf{L}}(i) = \begin{cases} 0 & \text{if } j \notin L_t \\ 1 & \text{else if } t = T \\ & \text{and } j \in L_t \\ SB(j) & \text{otherwise} \end{cases} \tag{6}$$

where

$$SA(j) = \sum_{i \in L_{t-1}} \alpha_{t-1,\mathbf{L}}(i)\Psi_t(j, i, x_t) \tag{7}$$

$$SB(j) = \sum_{j \in L_{t+1}} \beta_{t+1,\mathbf{L}}(j)\Psi_{t+1}(j, i, x_{t+1}) \tag{8}$$

and y_0 is a special label that encodes the beginning of a sequence.

3.2 Marginal Probability

The marginal probability $p(y_t = j|\mathbf{x}, \mathbf{L})$ is calculated as:

$$p(y_t = j|\mathbf{x}, \mathbf{L}) = \frac{\alpha_{t,\mathbf{L}}(j) \cdot \beta_{t,\mathbf{L}}(j)}{Z_{\mathbf{L}}(\mathbf{x})} \tag{9}$$

with:

$$\forall t, Z_{\mathbf{L}}(\mathbf{x}) = \sum_{j \in L_t} \alpha_{t,\mathbf{L}}(j) \cdot \beta_{t,\mathbf{L}}(j) \tag{10}$$

In case there is no label information, the formulas for forward and backward scores (Equations (5) and (6)) and for the marginal probabilities (Equation (9)) yield the standard results of CRFs.

3.3 Viterbi Algorithm

The most probable sequence assignment may be derived with a Viterbi algorithm by recursively computing the following quantities:

$$\delta_{t,\mathbf{L}}(j) = \begin{cases} 0 & \text{if } j \notin L_t \\ \Psi_1(j, y_0, x_1) & \text{else if } t = 1 \\ & \text{and } j \in L_t \\ M(j) & \text{otherwise} \end{cases} \quad (11)$$

where

$$M(j) = \max_{i \in L_{t-1}} \delta_{t-1,\mathbf{L}}(i) \Psi_t(j, i, x_t) \quad (12)$$

The most probable assignment is then calculated as: $\mathbf{y}^* = argmax_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}, \mathbf{L})$

3.4 Log-Likelihood

PL-CRFs's parameters θ are learnt through maximum log-likelihood estimation, that is to maximize the log-likelihood function $LL(\theta)$:

$$\begin{aligned} LL(\theta) &= \sum_{i=1}^{N} \log p(\mathbf{Y}_{\mathbf{L}^{(i)}}|\mathbf{x}^{(i)}) \\ &= \sum_{i=1}^{N} \log Z_{\mathbf{Y}_{L^{(i)}}}(\mathbf{x}^{(i)}) - \log Z_{\mathbf{Y}}(\mathbf{x}^{(i)}) \end{aligned} \quad (13)$$

The parameters θ that maximize Equation (13) are computed via the LBFGS optimization method (Byrd et al., 1994).

4 Active Learning Strategies

Pool-based AL (see (Lewis and Catlett, 1994)) is probably the most common scenario in AL, where one has a large amount (pool) of unlabeled examples $\mathcal{U}_1$ and a small amount of labeled examples $\mathcal{T}_1$. In this scenario, the process of AL consists in a series of n iterations where a classifier Φ_i is trained with labeled examples $\mathcal{T}_i$, and then is used to classify the unlabeled examples $\mathcal{U}_i$. At this point an AL strategy $\mathcal{S}$ will select a number of examples B that once labeled will hopefully improve the performance of the next classifier Φ_{i+1}.

Algorithm 1 shows the pool-based AL framework for partially annotated sequences as introduced in (Wanvarie et al., 2011). Differently from AL for fully labeled sequences (Esuli et al., 2010), thanks to the finer granularity of the partially labeled model, we use the token as basic annotation unit, instead of the entire sequence.

The point of using the partial labeling is in saving the request for human annotations on tokens whose labels are already known (inferred) by the classifier and concentrate on those tokens that the classifier finds hard to label. Using the semi-supervised approach of the PL-CRFs we can take advantage of single-labeled tokens instead of an entire labeled sequence.

The entire pool-based AL process with partially labeled sequences is summarized in Algorithm 1. The

Algorithm 1 Pool-based active learning framework

Require: $\mathcal{T}_1$, the initial training set
$\mathcal{U}_1$, the initial unlabeled set
$\mathcal{S}$, the selected AL strategy
n, the number of iterations
B, the dimension of the update batch

1: **for** $i \leftarrow 1$ to n **do**
2: $\quad \Phi_i \leftarrow \text{train}(\mathcal{T}_i)$
3: $\quad \mathcal{L}_i \leftarrow \Phi_i(\mathcal{U}_i)$
4: $\quad$ **for** $b \leftarrow 1$ to B **do**
5: $\quad\quad \mathbf{x}_*^{(b)} \leftarrow \arg\min_{\mathbf{x}_t \in \mathbf{x}, \mathbf{x} \in \mathcal{L}_i} \mathcal{S}(t, \mathbf{x})$
6: $\quad\quad \mathcal{L}_i \leftarrow \mathcal{L}_i - \mathbf{x}^{(b)} \cup \Phi_i(\mathbf{x}^{(b)}, y_*)$
7: $\quad\quad \mathcal{U}_i \leftarrow \mathcal{U}_i - \mathbf{x}_*^{(b)} \cup (\mathbf{x}_*^{(b)}, y_*)$
8: $\quad\quad \mathcal{T}_i \leftarrow \mathcal{T}_i - \mathbf{x}_*^{(b)} \cup (\mathbf{x}_*^{(b)}, y_*)$
9: $\quad \mathcal{U}_{i+1} \leftarrow \mathcal{U}_i$
10: $\quad \mathcal{T}_{i+1} \leftarrow \mathcal{T}_i$

function $\mathcal{S}(t, \mathbf{x})$ is what, hereafter, we call an AL *strategy*. $\mathcal{S}(t, \mathbf{x})$ takes as input an automatically annotated sequence $\mathbf{x}$ and an element t of this sequence, from the set of sequences $\mathcal{L}_i$ annotated by the PL-CRF classifier Φ_i, and returns a measure of informativeness as a function of the classifier decision.

For each iteration through the update batch B, the most informative element $\mathbf{x}_*^{(b)}$, according to the AL strategy, is chosen. The subscript $*$, in this case, represents the most informative token, while the superscript (b) represents the sequence in which the token appears. After the choice of the most informative token the sets $\mathcal{L}_i$, $\mathcal{U}_i$ and $\mathcal{T}_i$ are updated. $\mathcal{L}_i$ is updated by removing the annotated sequence $\mathbf{x}^{(b)}$ and all the information given by the classifier, and by adding the same sequence with the new manually labeled token (y_*) and all the re-estimated annotation given by the classifier $\Phi_i(\mathbf{x}^{(b)}, y_*)$. In the unlabeled set $\mathcal{U}_i$ and the training set $\mathcal{T}_i$ the most informative token $\mathbf{x}_*^{(b)}$ is updated with its manually labeled version $(\mathbf{x}_*^{(b)}, y_*)$[1]. After B token annotations, the unlabeled set and the training set for the next iteration, respectively $\mathcal{U}_{i+1}$ and $\mathcal{T}_{i+1}$, are updated.

The inference methods of Section 3 allow not only to train a CRF model with partially labeled sequences, but give the possibility of classifying partially labeled sequences, using the known labels as support for the prediction of the other ones. Thus, in this AL scenario, each time a token is chosen it is immediately labeled, and this new information, as we can see from line 6 of Algorithm 1, is promptly used to re-estimate the informativeness of the other tokens in the sequence in which the chosen token appears.

One may argue that, for a human annotator, anno-

[1] In order to have a light notation we omit the fact that when the most informative token is the first annotated token of a sentence, the whole sentence, with just one annotated token, is added to the training set $\mathcal{T}_i$

tating only one or few tokens, instead of the entire sequence, is a difficult task. This would be correct in the scenario in which the text is presented to the human annotator without any visual clue about the annotations. However, in (Culotta and McCallum, 2005) it is shown that presenting to the human annotator the highlighted sequence to be annotated along with the associated sequence of labels obtained by the classifier requires much less effort from the annotator than performing the annotation without any visual and contextual clue.

4.1 Greedy Strategies

In this section we present three AL strategies that select the most informative tokens, regardless of the assignment performed by the Viterbi algorithm. The rationale behind these strategies is that, even though we are looking for the most probable sequence assignment, we also want to annotate the most informative tokens singularly.

The **Minimum Token Probability (MTP)** strategy employs as measure of informativeness the probability of the most probable assignment at time t. This strategy greedily samples the tokens whose highest probability among the labels is lowest.

$$\mathcal{S}^{MTP}(t, \mathbf{x}) = \max_{j \in \mathcal{Y}} p(y_t = j | \mathbf{x}, \mathbf{L}) \tag{14}$$

The **Maximum Token Entropy (MTE)** strategy relies on the entropy measure to evaluate the ambiguity about the label of a token. The rationale of it is that, if more than one label have the same assigned marginal probability, the entropy will be high, that is,

$$\mathcal{S}^{MTE}(t, \mathbf{x}) = \sum_{j \in \mathcal{Y}} p(y_t = j | \mathbf{x}, \mathbf{L}) \cdot \log p(y_t = j | \mathbf{x}, \mathbf{L}) \tag{15}$$

In order to directly plug the $\mathcal{S}^{MTE}$ strategy into the AL framework of Algorithm 1, we removed the minus sign at the beginning of the entropy formula. This allow us to use the $\min$ operator with a maximum entropy approach.

The **Minimum Token Margin (MTM)** strategy is a variant of the margin sampling strategy introduced in (Scheffer et al., 2001). It calculates the informativeness by considering the two most probable assignments and by subtracting the highest probability by the lowest. With $\max'$ that calculates the second maximum value, MTM is defined as:

$$\mathcal{S}^{MTM}(t, \mathbf{x}) = \max_{j \in \mathcal{Y}} p(y_t = j | \mathbf{x}, \mathbf{L}) - \max_{j \in \mathcal{Y}}{}' p(y_t = j | \mathbf{x}, \mathbf{L}) \tag{16}$$

4.2 Viterbi Strategies

The following AL strategies take into consideration the most probable sequence assignments obtained from the Viterbi algorithm computed on already known labels in the sequence.

The rationale is that, with these strategies, the measure of uncertainty is chosen according to the information obtained from the outcome of the Viterbi algorithm (i.e., the most probable sequence assignment).

The **Minimum Viterbi Probability (MVP)** is the base strategy adopted in (Wanvarie et al., 2011). It takes as measure of informativeness the probability of the label chosen by the Viterbi algorithm.

$$\mathcal{S}^{MVP}(t, \mathbf{x}) = p(y_t^* | \mathbf{x}, \mathbf{L}) \tag{17}$$

where y_t^* is the label assignment chosen by the Viterbi algorithm. In general, the token assignments that maximize the probability of the sequence assignment y_t^* are different from the token assignments that maximize the probability of the individual token assignments $argmax_{j \in \mathcal{Y}} p(y_t = j)$.

The **Maximum Viterbi Pseudo-Entropy (MVPE)** strategy calculates for each token the "pseudo" entropy of the most probable sequences at the variation of the label at position t. The prefix pseudo is used because even though it is calculated as an entropy, the summation is over all the possible labels that can be associated to a token, and not all the possible sequence assignments.

$$\mathcal{S}^{MVPE}(t, \mathbf{x}) = \sum_{j \in \mathcal{Y}} p(\mathbf{y}^*_{y_t=j} | \mathbf{x}, \mathbf{L}) \cdot \log p(\mathbf{y}^*_{y_t=j} | \mathbf{x}, \mathbf{L}) \tag{18}$$

where $\mathbf{y}^*_{y_t=j}$ represents the most probable assignment with the label at time t constrained to the value j. As in the MTE strategy the minus sign is removed in order to plug the functions directly into the AL framework of Algorithm 1.

The **Minimum Viterbi Margin (MVM)** strategy calculates the difference of the sequence probabilities of the two most probable sequence assignments at the variation of the label at time t. When the difference at time t is low, the Viterbi algorithm, in that time, chooses between two almost equally probable, sequence assignments. Formally:

$$\mathcal{S}^{MVM}(t, \mathbf{x}) = p(\mathbf{y}^*_{y_t^*} | \mathbf{x}, \mathbf{L}) - p(\mathbf{y}'^*_{y_t'^*} | \mathbf{x}, \mathbf{L}) \tag{19}$$

where $\mathbf{y}'^*$ is the second most probable assignment.

PL-CRFs allow us to inspect one token at time in order to decide if it is worth to annotate. This fact give us the possibility of exploit two quantities in order to estimate the informativeness of a token, the sequence probability, usually adopted in the traditional AL for sequence labeling, and the marginal probabilities of the single tokens as in Section 4.1. The **Minimum Expectation (ME)** strategy combines the marginal probabilities, $p(y_t = j | \mathbf{x}, \mathbf{L})$ and $p(\mathbf{y}^*_{y_t=j} | \mathbf{x}, \mathbf{L})$.

$$\mathcal{S}^{ME}(t, \mathbf{x}) = \sum_{j \in \mathcal{Y}} p(y_t = j | \mathbf{x}, \mathbf{L}) \cdot p(\mathbf{y}^*_{y_t=j} | \mathbf{x}, \mathbf{L}) \tag{20}$$

Here the maximum sequence probability is seen as a function, and what we calculate is the expected value of this very function. The rationale of this strategy is picking those tokens in which both, the sequence probability returned by the Viterbi algorithm, and the marginal probability of the considered labels are low.

Given that the ME strategy gives a high weight to the sequence probability, one might expect that tokens that belongs to longer sequences are selected more frequently, given that, the sequence probability of longer sequences is usually lower than shorter ones. One way to normalize this difference is subtracting the current maximum sequence probability, that is, the maximum sequence probability calculated without any new label estimation, to the expected value obtained from the estimation of the label assignment of the token. This is the **Minimum Expectation Difference (MED)** strategy.

$$\mathcal{S}^{MED}(t, \mathbf{x}) = \mathcal{S}^{ME}(t, \mathbf{x}) - p(\mathbf{y}^*|\mathbf{x}, \mathbf{L}) \qquad (21)$$

The rationale of this strategy is that when the expected value is far from the maximum value, that is the value returned by the Viterbi algorithm, it means that we have uncertainty on the token taken into consideration.

The **Random (RAND)** strategy samples random tokens without any external information. It is used as *baseline* to compare the real effectiveness of the proposed strategy.

At the best of our knowledge the strategies presented in this section (with the exception of the MVP strategy) have never been applied in the context of AL with partially labeled sequences scenario.

5 Experiments

5.1 Datasets

We have experimented and evaluated the AL strategies of Section 4 on four sequence labeling tasks, part-of-speech tagging, phrase chunking, named-entity recognition and bio-entity recognition. We used the CoNLL2000 dataset (Tjong Kim Sang and Buchholz, 2000) for the phrase chunking task, the CoNLL2003 dataset (Tjong Kim Sang and De Meulder, 2003), for the named-entity recognition task, the NLPBA2004 dataset (Kim et al., 2004), for the biomedical entity recognition task and the CoNLL2000POS dataset[2] for the part-of-speech labeling task. All the datasets are publicly available and are standard benchmarks in sequence labeling tasks. Table 1 shows some statistics of the datasets in terms of dimensions, number of labels, distribution of the labels, etc. The data heterogeneity of the different datasets allowed us to test the AL strategies on different "experimental settings", thus to have a more robust empirical evaluation.

[2]This is the CoNLL2000 dataset annotated with part-of-speech labels instead of chunking labels.

5.2 Experimental Setting

We tested the AL strategies described in Section 4 on test sets composed by 2012 sequences and 47377 tokens for the CoNLL2000 and CoNLL2000POS datasets, by 3452 sequences and 46394 tokens for the CoNLL2003 dataset and by 3856 sequences and 101039 tokens for the NLPBA2004 dataset. We chose an initial training set $\mathcal{T}_1$ of $\sim$5 sequences on CoNLL2000 and CoNLL2000POS datasets, $\sim$7 sequences on CoNLL2003 dataset and $\sim$4 sequences on NLPBA2004 dataset, for a total of $\sim$100 labeled tokens for each dataset. The dimension of the batch update B has been chosen as a trade-off between an ideal case in which the system is retrained after every single annotation (i.e., $B = 1$) and a practical case with higher B to limit the algorithmic complexity (since the PL-CRF classifier must be retrained every iteration). We used in our experiments $B = 50$. We fixed the number of AL iterations n at 40 because what matters here is how the strategies behave in the beginning of AL process when the annotation effort remains low. For each strategy and for each dataset, we report averaged results of three runs with a different randomly sampled initial training set $\mathcal{T}_1$.

For each dataset we adopted a standard set of features. For the CoNLL2000 dataset we adopted the same standard features used in (Wanvarie et al., 2011) for the same dataset, for the CoNLL2003 and the NLPBA2004 dataset we adopted the features used in (Wanvarie et al., 2011) for the CoNLL2003 dataset, while for the CoNLL2000POS dataset we used the features presented in (Ratnaparkhi, 1996). As evaluation measure we adopted the token variant of the F_1 measure, introduced by Esuli and Sebastiani (2010). This variant, instead of the entire annotation (chunk/entity), calculates TPs, FPs, and FNs, singularly for each token that compose the annotation, bringing to a finer evaluation.

5.3 Results

From the learning curves of Figure 1 and Figure 2 it is clear that most of the strategies have the same trend throughout the different datasets. This results is somewhat different from the results obtained in (Settles and Craven, 2008) in which there is not a clear winner among the strategies they proposed in a fully-labeled scenario. The strategies that perform particularly bad (worse than the RAND strategy in CoNLL2000POS and in CoNLL2003 dataset) in all the datasets are the MTE and MTP. This is expected, because the choice of the measure of informativeness related to the token without taking in consideration the Viterbi path is suboptimal in this task. Surprisingly, the MTM strategy even though based on the same principle of MTE and MTP, is very effective in most of the datasets. The most effective strategies, that is, the ones that are the faster at helping the classifier to reach a better accuracy are the MTM, MVM, and MVP, in particular the margin-based strategies perform very good in all the

Table 1: Training Data Statistics. #S is the number of total sequences in the dataset, #T is the number of tokens in the dataset, #L is the number of positive labels (labels different from the negative label O), AAL is the average length, in tokens, of annotations (sequence of tokens that refer to the same instance of a label), APT is the average number of token in a sequence annotated with a positive label, ASL is the average length of a sequence, AA is the average number of annotations in a sequence, %AC is the percentage of sequences with more than one positive annotation, %DAC is the percentage of sequences that have two or more annotations with different labels.

Dataset	#S	#T	#L	AAL	APT	ASL	AA	%AC	%DAC
CoNLL2000	8936	211727	11	1.6	20.6	24	12.0	98%	98%
CoNLL2000POS	8936	211727	35	1.0	20.8	24	20.8	100%	99%
CoNLL2003	17290	254979	4	1.4	2.5	15	2.2	45%	32%
NLPBA2004	18546	492551	5	2.5	5.9	27	3.1	72%	47%

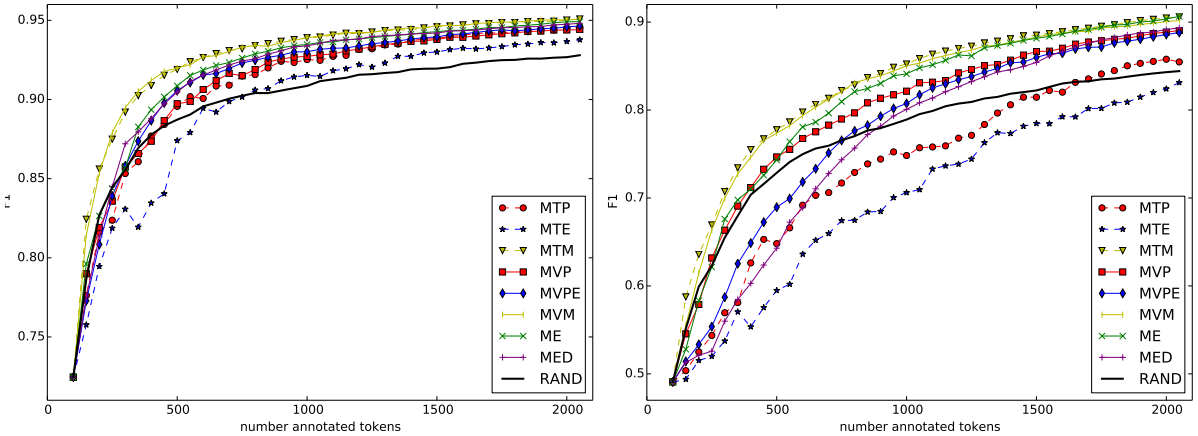


Figure 1: F_1 results on CoNLL2000 dataset (left) and CoNLL2000POS dataset (right). For both datasets the maximum number of annotated tokens used (2100) represents ∼1% of the entire training set.

datasets. The MVPE strategy performs particularly bad in the CoNLL2003 dataset but it performs better than the RAND strategy in the other datasets. The performance of the ME strategy is always above the average, in particular it is the best performing strategy in the NLPBA2004 dataset. However, in the CoNLL2003 dataset its performance is similar to the RAND's performance. Looking at the data, as expected, ME tends to choose tokens belonging to the longest sequences, regardless if the sequence is already partially annotated, that is, it tends to choose tokens from the same sequences. This behavior is not particularly relevant on the CoNLL2003 dataset given that the average number of positive tokens per sentence is not high (2.5, see Table 1). For the other datasets, the average number of positive tokens per sentence is high, and so the ME strategy is particularly effective. The MED strategy has the most heterogeneous behavior among the datasets. It shows average performances in the CoNLL2000 dataset and NLPBA2004 dataset, but is slower than the RAND strategy in the CoNLL2003 and CoNLL2000POS datasets.

In Figure 2 (left) we can notice that there are some strategies that are consistently worse than the RAND strategy. The difference between the strategies below the RAND strategy and the RAND strategy itself might be due to the fact that those strategies ask to label tokens that are "outliers" (if we imagine tokens as points of the features space) that rarely appear in the training and test set, and on which the classifier is very uncertain. Given that we are in a semi-supervised setting, with very few training examples, these "outliers" can introduce a lot of noise in the created models and so yielding poor results. This phenomenon does not happen in the RAND strategy given that it samples uniformly from the unlabeled set and given that the "outliers" (special cases) are not many, the probability of randomly selecting an "outlier" is low.

5.3.1 Performance Drop

The AL strategies applied on the CoNLL2003 dataset (Figure 2 (left)) suffer of some "random" drop of performance. We believe that the first reason that yield such a behavior is that named entities often appear once in a sentence, and have heterogeneous structures with respect to some homogenous structures as the chunk and POS. The second reason is that, it may happen that the strategies are not accurate enough to localize precisely the best token to label or that getting the label of an isolated token does not help the classifier much

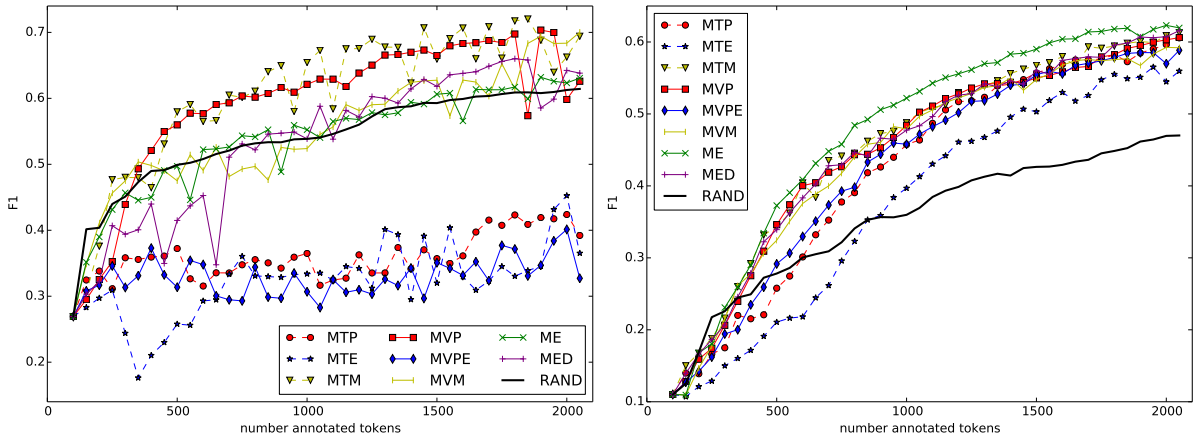


Figure 2: F_1 results on CoNLL2003 dataset (left) and NLPBA2004 dataset (right). 2100 annotated tokens represent the $\sim$0.8% and $\sim$0.4% respectively of the CoNLL2003 training set and the NLPBA2004 training set.

for the remaining of the (unlabeled) tokens in the sequence.

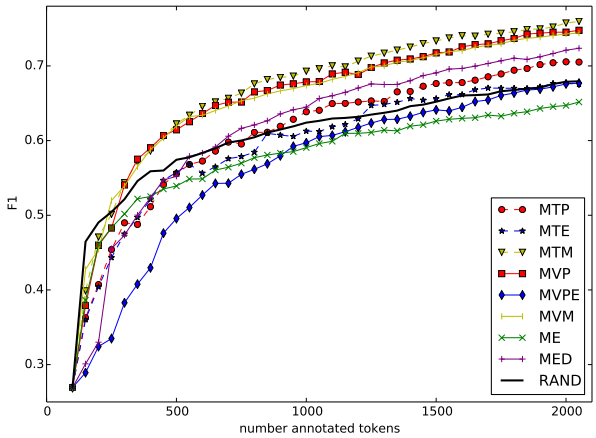


Figure 3: F_1 results on CoNLL2003 dataset, three tokens annotation. 6100 annotated tokens represent the $\sim$2.4% of the CoNLL2003 training set.

A similar phenomenon, called *missed class effect* (Tomanek et al., 2009), happens in AL when the strategies inspect regions of the example space around the decision boundaries, bringing to a slow AL process. In (Tomanek et al., 2009) the missed class effect problem is solved by helping the AL strategies to inspect regions far from the decision boundaries, that is, by choosing an entire sequence instead of a single token. This solution is not suitable in this context given that we will loose all the advantages we have in the partially-labeled scenario, thus, we decided to annotate for each chosen token the previous token and the next token. The learning curves of the AL strategies adopting this method (Figure 3) show a monotonically increasing performance in function of the number of annotated tokens.

By annotating three tokens at time, the tokens that were considered "outliers" in the scenario with a single token annotation are now supported by other tokens of the sequence. This fact helps to decrease the noise introduced in the semi-supervised model yielding better results.

5.3.2 Statistical Analysis

Figure 4 reports a few statistics that highlight the behavior of the methods on one of the datasets. One may see for instance that the MVM and ME strategies are very different from the other methods in that they select tokens that belong to significantly longer sentences on average. Also it may be seen that MVM in particular selects tokens that are far from already annotated tokens in the sentence. This strategy probably yields a particular behavior with respect to exploration and exploitation that seems to suit the two tasks well. The other strategies do exhibit different behaviors that intuitively should not work well. For instance the MED and the MVPE strategies select tokens from new fully unlabeled sentences (not shown statistics), preferably short, so that the distance from selected tokens to already labeled tokens in the sentence (when any) is low. These curves look like relevant indicators of the behavior of the methods, and it would probably be worth monitoring these all along the AL process to make sure the learning exhibit a suitable behavior. This will be a future study that is out of the scope of this work.

6 Conclusion

In this paper we have presented several AL strategies tailored for the PL-CRFs in a pool-based scenario. We have tested the proposed strategies on four different datasets for four different sequence labeling tasks. Differently from other similar work in the field of AL, in this study we have shown that margin-based strategies constantly achieve good performance on four tasks with very different data characteristics. Furthermore, we have found that on datasets with certain characteristics a particular phenomenon that makes the entire

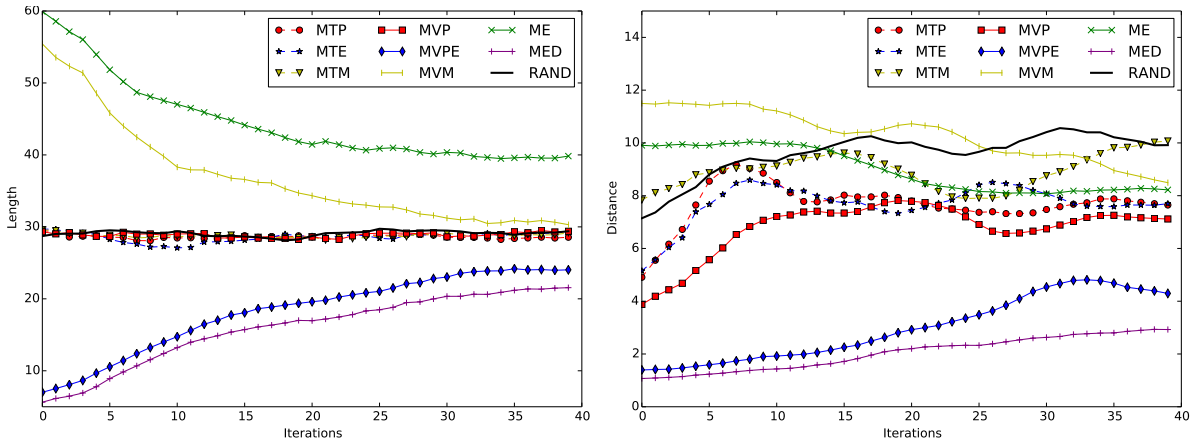


Figure 4: Behavior of the methods on CoNLL2000 dataset as a function of the number of the iterations (x-axis, from 1 to 40). Average length of the sentence the tokens that are selected by the AL strategy belong to (left) and average distance from a token that is selected to the closest already labeled token in the sentence, if any (right).

AL process highly unpredictable shows up. This phenomenon consists in random drops of accuracy of the classifiers learnt during the AL process. We have proposed a first solution for this problem that does not have a relevant impact on the human annotation effort.

Acknowledgments

We kindly thank Fabrizio Sebastiani and Andrea Esuli for their help and valuable comments, and Dittaya Wanvarie for providing us her implementation of partially-labeled CRFs.

References

Richard H. Byrd, Jorge Nocedal, and Robert B. Schnabel. 1994. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63:129–156.

Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *Proceedings of the Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, (AAAI 2005)*, pages 746–751, Pittsburgh, US.

Andrea Esuli and Fabrizio Sebastiani. 2010. Evaluating information extraction. In *Proceedings of the Conference on Multilingual and Multimodal Information Access Evaluation (CLEF 2010)*, pages 100–111, Padova, IT.

Andrea Esuli, Diego Marcheggiani, and Fabrizio Sebastiani. 2010. Sentence-based active learning strategies for information extraction. In *Proceedings of the First Italian Information Retrieval Workshop (IIR 2010)*, pages 41–45, Padua, Italy.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 70–75, Geneva, CH.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, pages 282–289, Williamstown, US.

David D. Lewis and Jason Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of 11th International Conference on Machine Learning (ICML 1994)*, pages 148–156, New Brunswick, US.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1, pages 133–142.

Sunita Sarawagi. 2008. Information extraction. *Foundations and Trends in Databases*, 1(3):261—377.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. 2001. Active hidden Markov models for information extraction. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis (IDA 2001)*, pages 309–318, Cascais, PT.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 1070–1079, Honolulu, US.

Burr Settles. 2012. *Active learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. 2004. Multi-criteria-based active

learning for named entity recognition. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL 2004)*, pages 589–596, Barcelona, ES.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 2nd Workshop on Learning Language in Logic and 4th Conference on Computational Natural Language Learning (LLL/CoNLL 2000)*, pages 127–132. Lisbon, PT.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning (CONLL 2003)*, pages 142–147, Edmonton, CA.

Katrin Tomanek and Udo Hahn. 2009. Semi-supervised active learning for sequence labeling. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP 2009)*, pages 1039–1047, Singapore.

Katrin Tomanek, Florian Laws, Udo Hahn, and Hinrich Schütze. 2009. On proper unit selection in active learning: co-selection effects for named entity recognition. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 9–17, Boulder, US.

Yuta Tsuboi, Hisashi Kashima, Hiroki Oda, Shinsuke Mori, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, pages 897–904, Manchester, UK.

Dittaya Wanvarie, Hiroya Takamura, and Manabu Okumura. 2011. Active learning with subsequence sampling strategy for sequence labeling tasks. *Information and Media Technologies*, 6(3):680–700.

Language Modeling with Functional Head Constraint for Code Switching Speech Recognition

Ying Li and Pascale Fung
Human Language Technology Center
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology
eewing@ee.ust.hk, pascale@ece.ust.hk

Abstract

In this paper, we propose novel structured language modeling methods for code mixing speech recognition by incorporating a well-known syntactic constraint for switching code, namely the Functional Head Constraint (FHC). Code mixing data is not abundantly available for training language models. Our proposed methods successfully alleviate this core problem for code mixing speech recognition by using bilingual data to train a structured language model with syntactic constraint. Linguists and bilingual speakers found that code switch do not happen between the functional head and its complements. We propose to learn the code mixing language model from bilingual data with this constraint in a weighted finite state transducer (WFST) framework. The constrained code switch language model is obtained by first expanding the search network with a translation model, and then using parsing to restrict paths to those permissible under the constraint. We implement and compare two approaches - lattice parsing enables a sequential coupling whereas partial parsing enables a tight coupling between parsing and filtering. We tested our system on a lecture speech dataset with 16% embedded second language, and on a lunch conversation dataset with 20% embedded language. Our language models with lattice parsing and partial parsing reduce word error rates from a baseline mixed language model by 3.8% and 3.9% in terms of word error rate relatively on the average on the first and second tasks respectively. It outperforms the interpolated language model by 3.7% and 5.6% in terms of word error rate relatively, and outperforms the adapted language model by 2.6% and 4.6% relatively. Our proposed approach avoids making early decisions on code-switch boundaries and is therefore more robust. We address the code switch data scarcity challenge by using bilingual data with syntactic structure.

1 Introduction

In multilingual communities, it is common for people to mix two or more languages in their speech. A single sentence spoken by bilingual speakers often contains the main, matrix language and an embedded second language. This type of linguistic phenomenon is called "code switching" by linguists. It is increasingly important for automatic speech recognition (ASR) systems to recognize code switching speech as they exist in scenarios such as meeting and interview speech, lecture speech, and conversational speech. Code switching is common among bilingual speakers of Spanish-English, Hindi-English, Chinese-English, and Arabic-English, among others. In China, lectures, meetings and conversations with technical contents are frequently peppered with English terms even though the general population is not considered bilingual in Chinese and English. Unlike the thousands and tens of thousands of hours of monolingual data available to train, for example, voice search engines, transcribed code switch data necessary for training language models is hard to come by. Code switch language modeling is therefore an even harder problem than acoustic modeling.

One approach for code switch speech recognition is to explicitly recognizing the code switch points by language identification first using phonetic or acoustic information, before applying speech recognizers for the matrix and embedded languages (Chan et. al, 2004; Shia et. al,

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 907–916,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

2004; Lyu and Lyu, 2008). This approach is extremely error-prone as language identification at each frame of the speech is necessary and any error will be propagated in the second speech recognition stage leading to fatal and irrecoverable errors.

Meanwhile, there are two general approaches to solve the problem of lack of training data for language modeling. In a first approach, two language models are trained from both the matrix and embedded language separately and then interpolated together (Vu et. al, 2012; Chan et. al, 2006). However, an interpolated language model effectively allows code switch at all word boundaries without much of a constraint. Another approach is to adapt the matrix language language model with a small amount of code switch data (Tsai et. al, 2010; Yeh et. al, 2010; Bhuvanagiri and Kopparapu, 2010; Cao et. al, 2010). The effectiveness of adaptation is also limited as positions of code switching points are not generalizable from the limited data. Significant progress in speech recognition has been made by using deep neural networks for acoustic modeling and language model. However, improvement thus gained on code switch speech recognition remains very small. Again, we propose that syntactic constraints of the code switching phenomenon can help improve performance and model accuracy. Previous work of using part-of-speech tags (Zhang et. al, 2008; Vu et al 2012) and our previous work using syntactic constraints (Li and Fung, 2012, 2013) have made progress in this area. Part-of-speech is relatively weak in predicting code switching points. It is generally accepted by linguists that code switching follows the so-called Functional Head Constraint, where words on the nodes of a syntactic sub tree must follow the language of that of the headword. If the headword is in the matrix language then none of its complements can switch to the embedded language.

In this work, we propose two ways to incorporate the Functional Head Constraint into speech recognition and compare them. We suggest two approaches of introducing syntactic constraints into the speech recognition system. One is to apply the knowledge sources in a sequential order. The acoustic model and a monolingual language model are used first to produce an intermediate lattice, then a second pass choose the best result using the syntactic constraints. Another approach uses tight coupling. We propose using structured language model (Chelba and Jelinek, 2000) to build the syntactic structure incrementally.

Following our previous work, we suggest incorporating the acoustic model, the monolingual language model and a translation model into a WFST framework. Using a translation model allows us to learn what happens when a language switches to another with context information. We will motivate and describe this WFST framework for code switching speech recognition in the next section. The Functional Head Constraint is described in Section 3. The proposed code switch language models and speech recognition coupling is described in Section 4. Experimental setup and results are presented in Section 5. Finally we conclude in Section 6.

2 Code Switch Language Modeling in a WFST Framework

As code switch text data is scarce, we do not have enough data to train the language model for code switch speech recognition. We propose instead to incorporate language model trained in the matrix language with a translation model to obtain a code switch language model. We propose to integrate a bilingual acoustic model (Li et. al, 2011) and the code switch language model in a weighted finite state transducer framework as follows.

Suppose X denotes the observed code switch speech vector, w_1^J denotes a word sequence in the matrix language, the hypothesis transcript v_1^I is as follows:

$$\begin{aligned}
\hat{v}_1^I &= \arg\max_{v_1^I} P(v_1^I|X) \\
&= \arg\max_{v_1^I} P(X|v_1^I)P(v_1^I) \\
&= \arg\max_{v_1^I} P(X|v_1^I) \sum_{w_1^J} P(v_1^I|w_1^J)P(w_1^J) \\
&\cong \arg\max_{v_1^I} P(X|v_1^I)P(v_1^I|w_1^J)P(w_1^J) \quad (1)
\end{aligned}$$

where $P(X|v_1^I)$ is the acoustic model and $P(v_1^I)$ is the language model in the mixed language.

Our code switch language model is obtained from a translation model $P(v_1^I|w_1^J)$ from the matrix language to the mixed language, and the language model in the matrix language $P(w_1^J)$.

Instead of word-to-word translation, the transduction of the context dependent lexicon transfer is constrained by previous words. Assume the transduction depends on the previous n words:

$$
\begin{aligned}
P(v_1^I|w_1^J) &= \prod_{i=1}^{I} P(v_i|v_1^{i-1}, w_1^i) \\
&\cong \prod_{i=1}^{I} P(v_{i-n+1}^{i-1}|w_{i-n+1}^{i}) \\
&= \prod_{i=1}^{I} \frac{P(v_i, w_i|v_{i-n+1}^{i-1}, w_{i-n+1}^{i-1})}{P(w_i|v_{i-n+1}^{i-1}, w_{i-n+1}^{i-1})} \\
&= \prod_{i=1}^{I} \frac{P(v_i, w_i|v_{i-n+1}^{i-1}, w_{i-n+1}^{i-1})}{P(w_i|\sum_{v_i} v_{i-n+1}^{i-1}, w_{i-n+1}^{i-1})}
\end{aligned} \tag{2}
$$

There are C-level and H-level search networks in the WFST framework. The C-level search network is composed of the universal phone model P, the context model C, the lexicon L, and the grammar G

$$N = P \circ C \circ L \circ G \tag{3}$$

The H-level search network is composed of the state model H, the phoneme model P, the context model C, the lexicon L, and the grammar G

$$N = H \circ P \circ C \circ L \circ G \tag{4}$$

The C-level requires less memory then the H-level search network. We propose to use a weighted finite state transducer framework incorporating the bilingual acoustic model P, the context model C, the lexicon L, and the code switching language models G_{CS} into a C-level search network for mixed language speech recognition. The output of the recognition result is in the mixed language after projection $\pi(G_{CS})$.

$$N = P \circ C \circ L \circ \pi(G_{CS}) \tag{5}$$

The WFST implementation to obtain the code switch language model G_{CS} is as follows:

$$G_{cs} = \mathcal{T} \circ G \tag{6}$$

where T is the translation model

$$P(\tilde{v}_1^L|w_1^J) = \prod_{l=1}^{L} P_l(\tilde{v}_l|w_l) \tag{7}$$

$P_l(\tilde{v}_l|w_l)$ is the probability of w_l translated into $\tilde{v}_l$.

In order to make use of the text data in the matrix language to recognize speech in the mixed language, the translation model $P(v_1^I|w_1^J)$ transduce the language model in the matrix language to the mixed language.

$$
\begin{aligned}
P(v_1^I|w_1^J) = \sum_{\tilde{v}_1^L, c_1^L, r_1^K, \tilde{w}_1^K} & P(\tilde{w}_1^K|w_1^J) \\
& \cdot P(r_1^K|\tilde{w}_1^K, w_1^J) \\
& \cdot P(c_1^L, r_1^K, \tilde{w}_1^K, w_1^J) \\
& \cdot P(\tilde{v}_1^K|c_1^L, r_1^K, \tilde{w}_1^K, w_1^J) \\
& \cdot P(v_1^I|\tilde{v}_1^K, r_1^K, \tilde{w}_1^K, w_1^J)
\end{aligned} \tag{8}
$$

where $P(\tilde{w}_1^K|w_1^J)$ is the word-to-phrase segmentation model, $P(r_1^K|\tilde{w}_1^K, w_1^J)$ is the phrasal reordering model, $P(c_1^L, r_1^K, \tilde{w}_1^K, w_1^J)$ is the chunk segmentation model, $P(\tilde{v}_1^K|c_1^L, r_1^K, \tilde{w}_1^K, w_1^J)$ is the chunk-to-chunk transduction model, $P(v_1^I|\tilde{v}_1^K, r_1^K, \tilde{w}_1^K, w_1^J)$ is the chunk-to-word reconstruction model.

The word-to-phrase segmentation model extracts a table of phrases $\{\tilde{v}_1, \tilde{v}_2, ..., \tilde{v}_K\}$ for the transcript in the embedded language and $\{\tilde{w}_1, \tilde{w}_2, ..., \tilde{w}_K\}$ for the transcript in the matrix language based on word-to-word alignments trained in both directions with GIZA++ (Och and Ney, 2003). The chunk segmentation model performs the segmentation of a phrase sequence $\tilde{w}_1^K$ into L phrases $\{c_1, c_2, ..., c_L\}$ using a segmentation weighted finite-state transducer. Assumes that a chunk c_l is code-switched to the embedded language independently by each chunk, the chunk-to-chunk transduction model is the probability of a chunk to be code switched to the embedded language trained on parallel data. The reconstruction model generates word sequence from chunk sequences and operates in the opposite direction to the segmentation model.

3 Functional Head Constraint

Many linguistics (Abney 1986; Belazi et. al, 1994; Bhatt 1994) have discovered the so-called Functional Head Constraint in code switching. They have found that code switches between a functional head (a complementizer, a determiner, an inflection, etc.) and its complement (sentence, noun-phrase, verb-phrase) do not happen in natural speech. In addition, the Functional Head Constraint is language independent.

In this work, we propose to investigate and incorporate the Functional Head Constraint into code switching language modeling in a WFST framework. Figure 1 shows one of the Functional Head Constraint examples. Functional heads are

the roots of the sub trees and complements are part of the sub trees. Actual words are the leaf nodes. According to the Functional Head Constraint, the leave nodes of a sub tree must be in either the matrix language or embedded language, following the language of the functional head. For instance, the third word "東西/something" is the head of the constituents "非常/very 重要的/important 東西/something". These three constituent words cannot be switched. Thus, it is not permissible to code switch in the constituent. More precisely, the language of the constituent is constrained to be the same as the language of the headword. In the following sections, we describe the integration of the Functional Head Constraint and the language model.

We have found this constraint to be empirically sound as we look into our collected code mixing speech and language data. The only violation of the constraint comes from rare cases of borrowed words such as brand names with no translation in the local, matrix language. Borrowed words are used even by monolingual speakers so they are in general part of the matrix language lexicon and require little, if any, special treatment in speech recognition.

In the following sections, we describe the integration of Functional Head Constraint and the language model.

4 Code Switching Language Modeling with Functional Head Constraint

We propose two approaches of language modeling with Functional Head Constraint: 1) lattice-parsing and sequential-coupling (Chapplerler et. al, 1999); 2) partial-parsing and tight-coupling (Chapplerler et. al, 1999). The two approaches will be described in the followed sections.

4.1 Sequential-coupling by Lattice-based Parsing

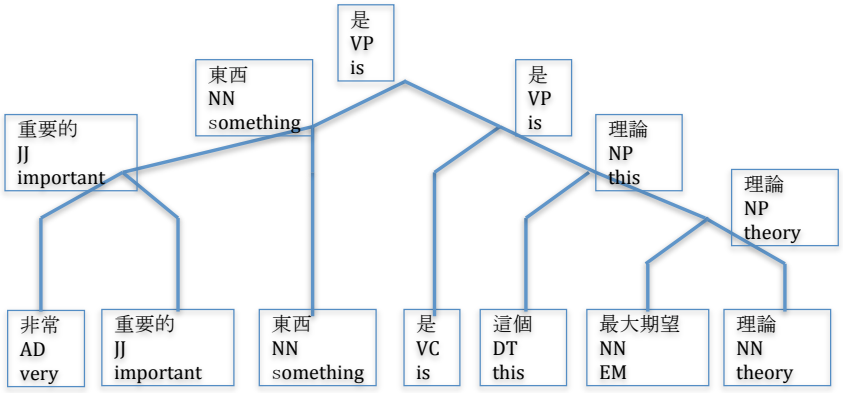
In this first approach, the acoustic models, the code switch language model and the syntactic constraint are incorporated in a sequential order to progressively constrain the search. The acoustic models and the matrix language model are used first to produce an intermediate output. The intermediate output is a lattice in which word sequences are compactly presented. Lattice-based parsing is used to expand the word lattice generated from the first decoding step according to the Functional Head Constraint.

We have reasons to use word lattice instead of N-best hypothesis. The number of hypothesis of word lattice is larger than N-best hypothesis. Moreover, different kinds of errors correspond to the language model would be observed if N-best list is extracted after the first decoding step. The second pass run over the N-best list will prevent the language model with Functional Head Constraint from correcting the errors. In order to obtain a computational feasible number of hypotheses without bias to the language model in the first decoding step, word lattice is used as the intermediate output of the first decoding step.

A Probabilistic Context-Free Grammar (PCFG) parser is trained on Penn Treebank data. The PCFG parser is generalized to take the lattice generated by the recognizer as the input. Figure 2 illustrates a word lattice which is a compact representation of the hypothesis transcriptions of a an input sentence. All the nodes of the word-lattice are ordered by increasing depth.

A CYK table is obtained by associating the arcs with their start and end states in the lattice instead of their sentence position and initialized all the cells in the table corresponding to the arcs (Chapplerler et. al, 1999). Each cell $C_{k,j}$ of the table is filled by a n-tuple of the non-terminal A, the length k and the starting position of the word sequence $w_j...w_{j+k}$ if there exists a PCFG rule $A \rightarrow w_j...w_{j+k}$, where A is a non-terminal which parse sequences of words $w_j...w_{j+k}$. In order to allow all hypothesis transcriptions of word lattice to be taken into account, multiple word sequences of the same length and starting point are initialized in the same cell. Figure 2 mapped the word lattice of the example to the table, where the starting node label of the arc is the column index and the length of the arc is the row index.

The sequential-coupling by lattice-parsing consists of the standard cell-filling and the self-filling steps. First, the cells $C_{k,j}$ and $C_{i-k,j+k}$ are combined to produce a new interpretation for cell $C_{i,j}$. In order to handle the unary context-free production $A \rightarrow B$ and update the cells after the standard cell-filling, a n-tuple of A, i and j is added for each n-tuple of the non-terminal B, the length i and the start j in the cell $C_{i,j}$. The parse trees extracted are associated with the input lattice from the table starting from the non-terminal label of the top cell. After the parse tree is obtained, we re-



Hypotheses: 非常重要的是這個 EM 理論.
非常重要的是這個 EM theory.
非常重要的東西是 this EM theory.
非常重要的東西 is this EM theory.
非常重要的 something is this EM theory. (not permissible)
.
.
.

Figure 1: A Functional Head Constraint example.

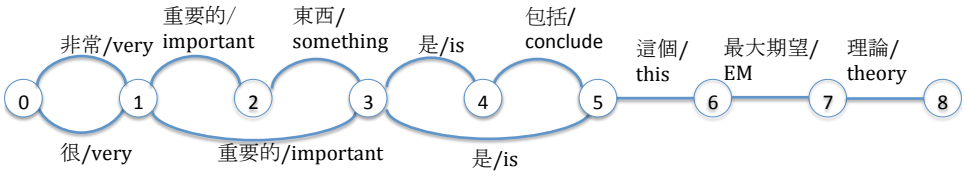


Figure 2: An example word lattice in the matrix language.

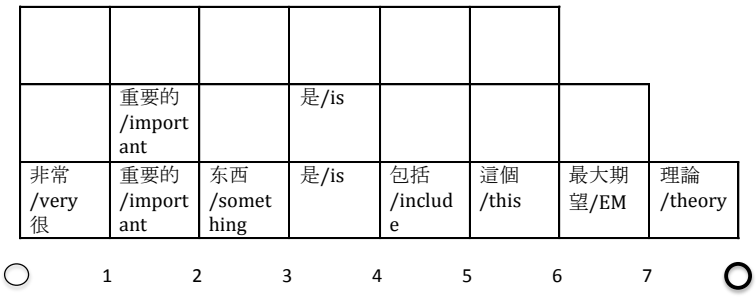


Figure 3: The mapping of the example word lattice to the table.

cursively enumerate all its subtrees. Each subtree is able to code-switch to the embedded language with a translation probability $P_l(\tilde{v}_l|w_l)$.

The lattice parsing operation consists of the an encoding of a given word sequence along with a parse tree (W,T) and a sequence of elementary model actions. In order to obtain a correct probability assignment $P(W,T)$ one simply assign proper conditional probabilities to each transition in the weighted finite states.

The probability of a parse T of a word sequence $WP(W,T)$ can be calculated as the product of the probabilities of the subtrees.

$$P(W,T) = \prod_{k=1}^{n+1} [P(wk|W_{k-1}T_{k-1}) \quad (9)$$

Where $W_k = w_0...w_k$ is the first k words in the sentence, and (W_k, T_k) is the word-and-parse k-prefix. The probability of the n-tuple of the non-terminal A, the length i and the starting position j is the probability of the subtree corresponding to A parsing throughout the sequence $w_j...w_{j+i-1}$. The probability of the partial parsing is the product of probabilities of the subtree parses it is made of. The probability of an n-tuple is the maximum over the probabilities of probable parsing path.

The N most probable parses are obtained during the lattice-parsing.

The probability of a sentence is computed by adding on the probability of each new context-free rule in the sentences.

4.2 Tight-coupling by Incremental Parsing

To integrate the acoustic models, language model and the syntactic constraint in time synchronous decoding, an incremental operation is used in this approach. The final word-level probability assigned by our model is calculated using the acoustic models, the matrix language model, the structured language model and the translation model. The structured language model uses probabilistic parameterization of a shift-reduce parse (Chelba and Jelinek, 2000). The tight-coupled language model consists of three transducers, the word predictor, the tagger and the constructor. As shown in Figure 3, $W_k = w0...wk$ is the first k words of the sentence, T_k contains only those binary subtrees whose leaves are completely included in W_k, excluding w_0 =<s>. Single words along with their POS tag can be regarded as root-only trees. The exposed head h_k is a pair of the headword of the constituent W_k and the non-terminal label. The exposed head of single words are pairs of the words and their POS tags.

Given the word-and-parse (k-1)-prefix $W_{k-1}T_{k-1}$, the new word w_k is predicted by the word-predictor $P(w_k|W_{k-1}T_{k-1})$. Taking the word-and-parse $k-1$-prefix and the next word as input, the tagger $P(t_k|w_k, W_{k-1}T_{k-1})$ gives the POS tag t_k of the word w_k. Constructor $P(p_i^k|W_kT_k)$ assigns a non-terminal label to the constituent W_{k+1}. The headword of the newly built constituent is inherited from either the headword of the constituent W_k or the next word w_{k+1}.

$$\begin{aligned} &P(w_k|W_{k-1}T_{k-1}) \\ &= P(w_k|[W_{k-1}T_{k-1}]) \\ &= P(w_k|h_0, h_{-1}) \quad (10) \\ &P(t_k|w_k, W_{k-1}T_{k-1}) \\ &= P(t_k|w_k, [W_{k-1}T_{k-1}]) \\ &= P(t_k|w_k, h_0.tag, h_{-1}.tag) \quad (11) \\ &P(p_i^k|W_kT_k) \\ &= P(p_i^k|[W_kT_k]) \\ &= P(p_i^k|h_0, h_1) \quad (12) \end{aligned}$$

The probability of a parse tree T $P(W,T)$ of a word sequence W and a complete parse T can be calculated as:

$$\begin{aligned} P(W,T) = \prod_{k=1}^{n+1} [&P(w_k|W_{k-1}T_{k-1}) \\ &P(t_k|W_{k-1}T_{k-1}, w_k) \\ &P(T_k|W_{k-1}T_{k-1}, w_k, t_k)] \quad (13) \end{aligned}$$

$$\begin{aligned} &P(T_{k-1}^k|W_{k-1}T_{k-1}, w_k, t_k) \\ &= \prod_{i=1}^{N_k} P(p_k|W_{k-1}T_{k-1}, w_k, t_k, p_1^k...p_{i-1}^k) \end{aligned} \quad (14)$$

Where w_k is the word predicted by the word-predictor, t_k is the POS tag of the word w_k predicted by the tagger, $W_{k-1}T_{k-1}$ is the word-parse (k - 1)-prefix, T_{k-1}^k is the incremental parse structure that generates $T_k = T_{k-1}||T_{k-1}^k$ when attached to T_{k-1}; it is the parse structure built on top of T_{k-1} and the newly predicted word wk; the $||$ notation stands for concatenation; N_{k-1} is the number of operations the constructor executes at

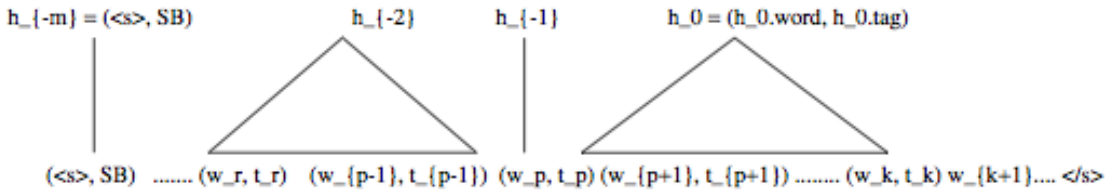


Figure 4: A word-and-parse example.

position k of the input string before passing control to the word-predictor (the N_k th operation at position k is the null transition); N_k is a function of T ; p_i^k denotes the i th constructor action carried out at position k in the word string.

The probability models of word-predictor, tagger and constructor are initialized from the Upenn Treebank with headword percolation and binarization. The headwords are percolated using a context-free approach based on rules of predicting the position of the headword of the constituent. The approach consists of three steps. First a parse tree is decomposed to phrase constituents. Then the headword position is identified and filled in with the actual word percolated up from the leaves of the tree recursively.

Instead of the UPenn Treebank-style, we use a more convenient binary branching tree. The parse trees are binarized using a rule-based approach.

The probability models of the word-predictor, tagger and constructor are trained in a maximization likelihood manner. The possible POS tag assignments, binary branching parse, non-terminal labels and the head-word annotation for a given sentence are hidden. We re-estimate them using EM algorithm.

Instead of generating only the complete parse, all parses for all the subsequences of the sentence are produced. The headwords of the subtrees are code switched to the embedded language with a translation probability $P_l(\tilde{v}_l|w_l)$ as well as the leaves.

4.3 Decoding by Translation

Using either lattice parsing or partial parsing, a two-pass decoding is needed to recognize code switch speech. A computationally feasible first pass generates an intermediate result so that the language model with Functional Head constraint can be used in the second pass. The first decoding pass composes of the transducer of the universal phoneme model P, the transducer C from context-dependent phones to context-independent phones, the lexicon transducer L which maps context-independent phone sequences to word strings and the transducer of the language model G. A T3 decoder is used in the first pass.

$$ASR_1 = P \circ C \circ L \circ G \tag{15}$$

Instead of N-best list, word lattice is used as the intermediate output of the first decoding step.

The language model G_{CS} of the transducer in the second pass is improved from G by composing with the translation model $P_l(\tilde{v}_l|w_l)$. Finally, the recognition transducer is optimized by determination and minimization operations.

$$ASR_2 = P \circ C \circ min(det(L \circ min(det(\pi(G_{CS}))))) \tag{16}$$

5 Experiments

5.1 Experimental Setup

The bilingual acoustic model used for our mixed language ASR is trained from 160 hours of speech from GALE Phase 1 Chinese broadcast conversation, 40 hours of speech from GALE Phase 1 English broadcast conversation, and 3 hours of in-house nonnative English data. The acoustic features used in our experiments consist of 39 components (13MFCC, 13MFCC, 13 MFCC using cepstral mean normalization), which are analyzed at a 10msec frame rate with a 25msec window size. The acoustic models used throughout our paper are state-clustered crossword tri-phone HMMs with 16 Gaussian mixture output densities per state. We use the phone set consists of 21 Mandarin standard initials, 37 Mandarin finals, 6 zero initials and 6 extended English phones. The pronunciation dictionary is obtained by modifying Mandarin and English dictionaries using the phone set. The acoustic models are reconstructed

Table 1: *Code switching point detection evaluation (Precision/Recall/F-measure)*

	Lecture speech	Lunch conversation
MixedLM	0.61/0.64/0.64	0.54/0.63/0.58
InterpolatedLM	0.62/0.66/0.64	0.55/0.63/0.58
AdaptedLM	0.63/0.71/0.67	0.54/0.63/0.58
Sequential coupling	0.66/0.71/0.68	0.55/0.70/0.61
Tight coupling	**0.68/0.71/0.70**	**0.56/0.70/0.62**

by decision tree tying. We also collected two speech databases with Chinese to English code switching - namely, 20 hours of lecture speech corpus (Data 1) and 3 hours of lunch conversation corpus (Data 2). 18 hours of Data 1 is used for acoustic model adaptation and 1 hour of data are used as the test set (Test 1). 2 hours of Data 2 containing 2389 utterances is used to adapt the acoustic model and 280 utterances are used as the test set (Test 2). To train the parser, we use Chinese Treebank Version 5.0 which consists of 500 thousand words and use the standard data split (Petrov and Klein, 2007).

For the language models, transcriptions of 18 hours of Data 1 are trained as a baseline mixed language model for the lecture speech domain. 250,000 sentences from Chinese speech conference papers, power point slides and web data are used for training a baseline Chinese matrix language model for the lecture speech domain (LM 1). Transcriptions of 2 hours of Data 2 are used as the baseline mixed language model in the lunch conversation domain. 250,000 sentences of the GALE Phase 1 Chinese conversational speech transcriptions are used to train a Chinese matrix language model (LM 2). 250,000 of GALE Phase 1 English conversational speech transcription are used to train the English embedded language model (LM 3). To train the bilingual translation model, the Chinese Gale Phase 1 conversational speech transcriptions are used to generate a bilingual corpus using machine translation. For comparison, an interpolated language model for the lunch conversation domain is trained from interpolating LM 2 with LM 3. Also for comparison, an adapted language model for lecture speech is trained from LM 1 and transcriptions of 18 hours of Data 1. An adapted language mode l for conversation is trained from LM 2 and 2 hours of Data 2. The size of the vocabulary for recognition is 20k words. The perplexity of the baseline language model trained on the code switching speech transcription is 236 on the lecture speech and 279 on the conversation speech test sets.

5.2 Experimental Results

Table 1 reports precision, recall and F-measure of code switching point in the recognition results of the baseline and our proposed language models. Our proposed code switching language models with functional head constraint improve both precision and recall of the code switching point detection on the code switching lecture speech and lunch conversation 4.48%. Our method by tight-coupling increases the F-measure by 9.38% relatively on the lecture speech and by 6.90% relatively on the lunch conversation compared to the baseline adapted language model.

The Table 2 shows the word error rates (WERs) of experiments on the code switching lecture speech and Table 3 shows the WERs on the code switching lunch conversations. Our proposed code switching language model with Functional Head Constraints by sequential-coupling reduces the WERs in the baseline mixed language model by 3.72% relative on Test 1, and 5.85% on Test 2. Our method by tight-coupling also reduces WER by 2.51% relative compared to the baseline language model on Test 1, and by 4.57% on Test 2. We use the speech recognition scoring toolkit (SCTK) developed by the National Institute of Standards and Technology to compute the significance levels, which is based on two-proportion z-test comparing the difference between the recognition results of our proposed approach and the baseline. All the WER reductions are statistically significant. For our reference, we also compare the performance of using Functional Head Constraint to that of using inversion constraint in (Li and Fung, 2012, 2013) and found that the present model reduces WER by 0.85% on Test 2 but gives no improvement on Test 1. We hypothesize that since

Table 2: *Our proposed system outperforms the baselines in terms of WER on the lecture speech*

	Matrix	Embedded	Overall
MixedLM	34.41%	39.16%	35.17%
InterpolatedLM	34.11%	40.28%	35.10%
AdaptedLM	35.11%	38.41%	34.73%
Sequential coupling	33.17%	36.84%	33.76%
Tight coupling	**33.14%**	**36.65%**	**33.70%**

Table 3: *Our proposed system outperforms the baselines in terms of WER on the lunch conversation*

	Matrix	Embedded	Overall
MixedLM	46.4%	48.55%	46.83%
InterpolatedLM	46.04%	49.04%	46.64%
AdaptedLM	46.64%	48.39%	46.20%
Sequential coupling	43.24%	46.27%	43.89%
Tight coupling	**42.97%**	**46.03%**	**43.58%**

Test 1 has mostly Chinese words, the proposed method is not as advantageous compared to our previous work. Another future direction is for us to improve the lattice parser as we believe it will lead to further improvement on the final result of our proposed method.

6 Conclusion

In this paper, we propose using lattice parsing and partial parsing to incorporate a well-known syntactic constraint for code mixing speech, namely the Functional Head Constraint, into a continuous speech recognition system. Under the Functional Head Constraint, code switch cannot occur between the functional head and its complements. Since code mixing speech data is scarce, we propose to instead learn the code mixing language model from bilingual data with this constraint. The constrained code switching language model is obtained by first expanding the search network with a translation model, and then using parsing to restrict paths to those permissible under the constraint. Lattice parsing enables a sequential coupling of parsing then constraint filtering whereas partial parsing enables a tight coupling between parsing and filtering. A WFST-based decoder then combines a bilingual acoustic model and the proposed code-switch language model in an integrated approach. Lattice-based parsing and partial parsing are used to provide the syntactic structure of the matrix language. Matrix words at the leave nodes of the syntax tree are permitted to switch to the embedded language if the switch does not violate the Functional Head Constraint. This reduces the permissible search paths from those expanded by the bilingual language model. We tested our system on a lecture speech dataset with 16% embedded second language, and on a lunch conversation dataset with 20% embedded second language. Our language models with lattice parsing and partial parsing reduce word error rates from a baseline mixed language model by 3.72% to 3.89% relative in the first task, and by 5.85% to 5.97% in the second task. They are reduced from an interpolated language model by 3.69% to 3.74%, and by 5.46% to 5.77% in the first and second task respectively. WER reductions from an adapted language model are 2.51% to 2.63%, and by 4.47% to 4.74% in the two tasks. The F-measure for code switch point detection is improved from 0.64 by the interpolated model to 0.68, and from 0.67 by the adapted model to 0.70 by our method. Our proposed approach avoids making early decisions on code-switch boundaries and is therefore more robust. Our approach also avoids the bottleneck of code switch data scarcity by using bilingual data with syntactic structure. Moreover, our method reduces word error rates for both the matrix and the embedded language.

Acknowledgments

This work is partially supported by grant number RGF 612211 of the Hong Kong Research Grants Council, by 1314159-0PAFT20F003 of the Ping An Research Institute and by 13140910 of the Huawei Noah's Ark Lab.

References

J.J. Gumperz, “Discourse strategies”, Cambridge University Press, 1, 1982.

Coulmas, F., “The handbook of sociolinguistics”, Wiley-Blackwell, 1998.

Vu, N.T. and Lyu, D.C. and Weiner, J. and Telaar, D. and Schlippe, T. and Blaicher, F. and Chng, E.S. and Schultz, T. and Li, H. *A first speech recognition system for Mandarin-English code-switch conversational speech'*, ICASSP, 2012

J.Y.C. Chan and PC Ching and T. Lee and H.M. Meng “Detection of language boundary in code-switching utterances by bi-phone probabilities” Chinese Spoken Language Processing, 2004 International Symposium on, 293–296.

C.J. Shia and Y.H. Chiu and J.H. Hsieh and C.H. Wu “Language boundary detection and identification of mixed-language speech based on MAP estimation””, ICASSP 2004.

D.C. Lyu and R.Y. Lyu “Language identification on code-switching utterances using multiple cues” Ninth Annual Conference of the International Speech Communication Association, 2008.

Tsai, T.L. and Chiang, C.Y. and Yu, H.M. and Lo, L.S. and Wang, Y.R. and Chen, S.H. “A study on Hakka and mixed Hakka-Mandarin speech recognition” Chinese Spoken Language Processing (ISCSLP), 2010 7th International Symposium on, 199–204

Yeh, C.F. and Huang, C.Y. and Sun, L.C. and Lee, L.S. “An integrated framework for transcribing Mandarin-English code-mixed lectures with improved acoustic and language modeling” Chinese Spoken Language Processing (ISCSLP), 2010 7th International Symposium on, 214–219

K. Bhuvanagiri and S. Kopparapu, “An Approach to Mixed Language Automatic Speech Recognition”, Oriental COCOSDA, Kathmandu, Nepal, 2010

Cao, H. and Ching, PC and Lee, T. and Yeung, Y.T. “Semantics-based language modeling for Cantonese-English code-mixing speech recognition Chinese Spoken Language Processing (ISCSLP), 2010 7th International Symposium on,246–250

Chelba, Ciprian, and Frederick Jelinek. ”Structured language modeling.” Computer Speech & Language 14, no. 4 (2000): 283-332.

Imseng, D. and Bourlard, H. and Magimai-Doss, M. and Dines, J., “Language dependent universal phoneme posterior estimation for mixed language speech recognition”, ICASSP, 2011.

Q. Zhang and J. Pan and Y. Yan, “Mandarin-English bilingual speech recognition for real world music retrieval”, ICASSP, 2008.

Bouselmi, G. and Fohr, D. and Illina, I., “Combined acoustic and pronunciation modelling for non-native speech recognition”, Eighth Annual Conference of the International Speech Communication Association, 2007.

Woolford, E., “Bilingual code-switching and syntactic theory”, in Linguistic Inquiry, 14(3):520–536, JSTOR, 1983.

MacSwan, J., “13 Code-switching and grammatical theory”, in The Handbook of Bilingualism and Multilingualism, 323 Wiley-Blackwell, 2012.

Poplack, S. and Sankoff, D., “A formal grammar for code-switching”, in Papers in Linguistics: International Journal of Human Communication, 3–45, 1980.

Moore, Robert C and Lewis, William, “Intelligent selection of language model training data” Proceedings of the ACL 2010 Conference Short Papers, 220–224.

Belazi, Heidi; Edward Rubin; Almeida Jacqueline Toribio “Code switching and X-Bar theory: The functional head constraint”. Linguistic Inquiry 25 (2): 221-37, 1994.

Bhatt, Rakesh M., “Code-switching and the functional head constraint” In Janet Fuller et al. Proceedings of the Eleventh Eastern States Conference on Linguistics. Ithaca, NY: Department of Modern Languages and Linguistics. pp. 1-12, 1995

Chappelier, Jean-C?dric, et al., “Lattice parsing for speech recognition.” TALN 1999.

A Polynomial-Time Dynamic Oracle for Non-Projective Dependency Parsing

Carlos Gómez-Rodríguez
Departamento de Computación
Universidade da Coruña, Spain
cgomezr@udc.es

Francesco Sartorio
Department of Information Engineering
University of Padua, Italy
sartorio@dei.unipd.it

Giorgio Satta
Department of Information Engineering
University of Padua, Italy
satta@dei.unipd.it

Abstract

The introduction of dynamic oracles has considerably improved the accuracy of greedy transition-based dependency parsers, without sacrificing parsing efficiency. However, this enhancement is limited to projective parsing, and dynamic oracles have not yet been implemented for parsers supporting non-projectivity. In this paper we introduce the first such oracle, for a non-projective parser based on Attardi's parser. We show that training with this oracle improves parsing accuracy over a conventional (static) oracle on a wide range of datasets.

1 Introduction

Greedy transition-based parsers for dependency grammars have been pioneered by Yamada and Matsumoto (2003) and Nivre (2003). These methods incrementally process the input sentence from left to right, predicting the next parsing action, called transition, on the basis of a compact representation of the derivation history.

Greedy transition-based parsers can be very efficient, allowing web-scale parsing with high throughput. However, the accuracy of these methods still falls behind that of transition-based parsers using beam-search, where the accuracy improvement is obtained at the cost of a decrease in parsing efficiency; see for instance Zhang and Nivre (2011), Huang and Sagae (2010), Choi and McCallum (2013). As an alternative to beam-search, recent research on transition-based parsing has therefore explored possible ways of improving accuracy at no extra cost in parsing efficiency.

The training of transition-based parsers relies on a component called the parsing oracle, which maps parser configurations to optimal transitions with respect to a gold tree. A discriminative model is then trained to simulate the oracle's behavior, and is later used for decoding. Traditionally, so-called static oracles have been exploited in training, where a static oracle is defined only for configurations that have been reached by computations with no mistake, and it returns a single canonical transition among those that are optimal.

Very recently, Goldberg and Nivre (2012), Goldberg and Nivre (2013) and Goldberg et al. (2014) showed that the accuracy of transition-based parsers can be substantially improved using dynamic oracles. A dynamic oracle returns the set of all transitions that are optimal for a given configuration, with respect to the gold tree, and is well-defined and correct for every configuration that is reachable by the parser.

Naïve implementations of dynamic oracles run in exponential time, since they need to simulate all possible computations of the parser for the input configuration. Polynomial-time implementations of dynamic oracles have been proposed by the above mentioned authors for several projective dependency parsers. To our knowledge, no polynomial-time algorithm has been published for transition-based parsers based on non-projective dependency grammars.

In this paper we consider a restriction of a transition-based, non-projective parser originally presented by Attardi (2006). This restriction was further investigated by Kuhlmann and Nivre (2010) and Cohen et al. (2011). We provide an implementation for a dynamic oracle for this parser running in polynomial time.

We experimentally compare the parser trained with the dynamic oracle to a baseline obtained by training with a static oracle. Significant accuracy improvements are achieved on many languages when using our dynamic oracle. To our knowledge, these are the first experimental results on non-projective parsing based on a dynamic oracle.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 917–927,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

2 Preliminary Definitions

Transition-based dependency parsing was originally introduced by Yamada and Matsumoto (2003) and Nivre (2003). In this section we briefly summarize the notation we use for this framework and introduce the notion of dynamic oracle.

2.1 Transition-Based Dependency Parsing

We represent an input sentence as a string $w = w_0 \cdots w_n$, $n \geq 1$, where each w_i with $i \neq 0$ is a lexical symbol and w_0 is a special symbol called root. Set $V_w = \{i \mid 0 \leq i \leq n\}$ denotes the symbol occurrences in w. For $i, j \in V_w$ with $i \neq j$, we write $i \rightarrow j$ to denote a grammatical **dependency** of some unspecified type between w_i and w_j, where w_i is the head and w_j is the dependent.

A **dependency tree** t for w is a directed tree with node set V_w and with root node 0. An arc of t is a pair (i, j), encoding a dependency $i \rightarrow j$; we will often use the latter notation to denote arcs.

A transition-based dependency parser typically uses a stack data structure to process the input string from left to right, in a way very similar to the classical push-down automaton for context-free languages (Hopcroft et al., 2006). Each stack element is a node from V_w, representing the root of a dependency tree spanning some portion of the input w, and no internal state is used. At each step the parser applies some transition that updates the stack and/or consumes one symbol from the input. Transitions may also construct new dependencies, which are added to the current configuration of the parser.

We represent the **stack** as an ordered sequence $\sigma = [h_d, \ldots, h_1]$, $d \geq 0$, of nodes $h_i \in V_w$, with the topmost element placed at the right. When $d = 0$, we have the empty stack $\sigma = []$. We use the vertical bar to denote the append operator for σ, and write $\sigma = \sigma'|h_1$ to indicate that h_1 is the topmost element of σ.

The portion of the input string still to be processed by the parser is called the **buffer**. We represent the buffer as an ordered sequence $\beta = [i, \ldots, n]$ of nodes from V_w, with i the first element of the buffer. We denote the empty buffer as $\beta = []$. Again, we use the vertical bar to denote the append operator, and write $\beta = i|\beta'$ to indicate that i is the first symbol occurrence of β; consequently, we have $\beta' = [i + 1, \ldots, n]$.

In a transition-based parser, the parsing process is defined through the technical notions of configuration and transition. A **configuration** of the parser relative to w is a triple $c = (\sigma, \beta, A)$, where σ and β are a stack and a buffer, respectively, and A is the set of arcs that have been built so far. A **transition** is a partial function mapping the set of parser configurations into itself. Each transition-based parser is defined by means of some finite inventory of transitions. We will later introduce the specific inventory of transitions for the parser that we investigate in this paper. We use the symbol $\vdash$ to denote the binary relation formed by the union of all transitions of a parser.

With the notions of configuration and transition in place, we can define a **computation** of the parser on w as a sequence $c_0, c_1, \ldots, c_m$, $m \geq 0$, of configurations relative to w, under the condition that $c_{i-1} \vdash c_i$ for each i with $1 \leq i \leq m$. We use the reflexive and transitive closure of $\vdash$, written $\vdash^*$, to represent computations.

2.2 Configuration Loss and Dynamic Oracles

A transition-based dependency parser is a nondeterministic device, meaning that a given configuration can be mapped into several configurations by the available transitions. However, in several implementations the parser is associated with a discriminative model that, on the basis of some features of the current configuration, always chooses a single transition. In other words, the model is used to run the parser as a pseudo-deterministic device. The training of the discriminative model relies on a component called the parsing **oracle**, which maps parser configurations to "optimal" transitions with respect to some reference dependency tree, which we call the **gold** tree.

Traditionally, so-called **static** oracles have been used which return a single, canonical transition and they do so only for configurations that can reach the gold tree, that is, configurations representing parsing histories with no mistake. In recent work, Goldberg and Nivre (2012), Goldberg and Nivre (2013) and Goldberg et al. (2014) have introduced **dynamic** oracles, which return the set of all transitions that are optimal with respect to a gold tree, and are well-defined and correct for every configuration that is reachable by the parser. These authors have shown that the accuracy of transition-based dependency parsers can be substantially improved if dynamic oracles are used in place of static ones. In what follows, we provide a mathematical definition of dynamic oracles, following Goldberg et al. (2014).

$$(\sigma, k|\beta, A) \vdash_{\mathsf{sh}} (\sigma|k, \beta, A)$$
$$(\sigma|i|j, \beta, A) \vdash_{\mathsf{la}} (\sigma|j, \beta, A \cup \{j \rightarrow i\})$$
$$(\sigma|i|j, \beta, A) \vdash_{\mathsf{ra}} (\sigma|i, \beta, A \cup \{i \rightarrow j\})$$
$$(\sigma|i|j|k, \beta, A) \vdash_{\mathsf{la}_2} (\sigma|j|k, \beta, A \cup \{k \rightarrow i\})$$
$$(\sigma|i|j|k, \beta, A) \vdash_{\mathsf{ra}_2} (\sigma|i|j, \beta, A \cup \{i \rightarrow k\})$$

Figure 1: Transitions of the non-projective parser.

Let t_1 and t_2 be dependency trees for w, with arc sets A_1 and A_2, respectively. The **loss** of t_1 with respect to t_2 is defined as

$$\mathcal{L}(t_1, t_2) = |A_1 \setminus A_2| \ . \tag{1}$$

Note that $\mathcal{L}(t_1, t_2) = \mathcal{L}(t_2, t_1)$, since $|A_1| = |A_2|$. Furthermore $\mathcal{L}(t_1, t_2) = 0$ if and only if t_1 and t_2 are the same tree.

Let c be a configuration of a transition-based parser relative to w. Let also $\mathcal{D}(c)$ be the set of all dependency trees that can be obtained in a computation of the form $c \vdash^* c_f$, where c_f is a final configuration, that is, a configuration that has constructed a dependency tree for w. We extend the loss function in (1) to configurations by letting

$$\mathcal{L}(c, t_2) = \min_{t_1 \in \mathcal{D}(c)} \mathcal{L}(t_1, t_2) \ . \tag{2}$$

Let t_G be the gold tree for w. Quantity $\mathcal{L}(c, t_G)$ can be used to define a dynamic oracle as follows. For any transition $\vdash_\tau$ in the finite inventory of our parser, we use the functional notation $\tau(c) = c'$ in place of $c \vdash_\tau c'$. We then let

$$\mathsf{oracle}(c, t_G) = \{\tau \mid \mathcal{L}(\tau(c), t_G) - \mathcal{L}(c, t_G) = 0\} \ . \tag{3}$$

In words, (3) provides the set of transitions that do not increase the loss of c; we call these transitions optimal for c.

A naïve way of implementing (3) would be to explicitly compute the set $\mathcal{D}(c)$ in (2), which has exponential size. More interestingly, the implementation of dynamic oracles proposed by the above cited authors all run in polynomial time. These oracles are all defined for projective parsing. In this paper, we present a polynomial-time oracle for a non-projective parser.

3 Non-Projective Dependency Parsing

In this section we introduce a parser for non-projective dependency grammars that is derived from the transition-based parser originally presented by Attardi (2006), and was further investigated by Kuhlmann and Nivre (2010) and Cohen et al. (2011). Our definitions follow the framework introduced in Section 2.1.

We start with some additional notation. Let t be a dependency tree for w and let k be a node of t. Consider the complete subtree t' of t rooted at k, that is, the subtree of t induced by k and all of the descendants of k in t. The **span** of t' is the subsequence of tokens in w represented by the nodes of t'. Node k has **gap-degree** 0 if the span of t' forms a (contiguous) substring of w. A dependency tree is called **projective** if all of its nodes have gap-degree 0; a dependency tree which is not projective is called **non-projective**.

Given w as input, the parser starts with the initial configuration $([], [0, \ldots, n], \emptyset)$, consisting of an empty stack, a buffer with all the nodes representing the symbol occurrences in w, and an empty set of constructed dependencies (arcs). The parser stops when it reaches a final configuration of the form $([0], [], A)$, consisting of a stack with only the root node and of an empty buffer; in any such configuration, set A always implicitly defines a valid dependency tree (rooted in node 0).

The core of the parser consists of an inventory of five transitions, defined in Figure 1. Each transition is specified using the free variables σ, β, A, i, j and k. As an example, the schema $(\sigma|i|j, \beta, A) \vdash_{\mathsf{la}} (\sigma|j, \beta, A \cup \{j \rightarrow i\})$ means that if a configuration c matches the antecedent, then a new configuration is obtained by instantiating the variables in the consequent accordingly.

The transition $\vdash_{\mathsf{sh}}$, called shift, reads a new token from the input sentence by removing it from the buffer and pushing it into the stack. Each of the other transitions, collectively called reduce transitions, has the effect of building a dependency between two nodes in the stack, and then removing the dependent node from the stack. The removal of the dependent ensures that the output dependency tree is built in a bottom-up order, collecting all of the dependents of each node i before linking i to its head.

The transition $\vdash_{\mathsf{la}}$, called left-arc, creates a leftward arc where the topmost stack node is the head and the second topmost node is the dependent, and removes the latter from the stack. The transition $\vdash_{\mathsf{ra}}$, called right-arc, is defined symmetrically, so that the topmost stack node is at-

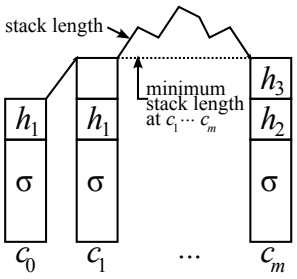


Figure 2: General form of the computations associated with an item $[h_1, h_2, h_3]$.

tached as a dependent of the second topmost node. The combination of the shift, left-arc and right-arc transitions provides complete coverage of projective dependency trees, but no support for non-projectivity, and corresponds to the so-called arc-standard parser introduced by Nivre (2004).

Support for non-projective dependencies is achieved by adding the transitions $\vdash_{\mathsf{la}_2}$ and $\vdash_{\mathsf{ra}_2}$, which are variants of the left-arc and right-arc transitions, respectively. These new transitions create dependencies involving the first and the *third* topmost nodes in the stack. The creation of dependencies between non-adjacent stack nodes might produce crossing arcs and is the key to the construction of non-projective trees.

Recall that transitions are partial functions, meaning that they might be undefined for some configurations. Specifically, the shift transition is only defined for configurations with a non-empty buffer. Similarly, the left-arc and right-arc transitions can only be applied if the length of the stack is at least 2, while the transitions $\vdash_{\mathsf{la}_2}$ and $\vdash_{\mathsf{ra}_2}$ require at least 3 nodes in the stack.

Transitions $\vdash_{\mathsf{la}_2}$ and $\vdash_{\mathsf{ra}_2}$ were originally introduced by Attardi (2006) together with other, more complex transitions. The parser we define here is therefore more restrictive than Attardi (2006), meaning that it does not cover all the non-projective trees that can be processed by the original parser. However, the restricted parser has recently attracted some research interest, as it covers the vast majority of non-projective constructions appearing in standard treebanks (Attardi, 2006; Kuhlmann and Nivre, 2010), while keeping simplicity and interesting properties like being compatible with polynomial-time dynamic programming (Cohen et al., 2011).

4 Representation of Computations

Our oracle algorithm exploits a dynamic programming technique which, given an input string, combines certain pieces of a computation of the parser from Section 3 to obtain larger pieces. In order to efficiently encode pieces of computations, we borrow a representation proposed by Cohen et al. (2011), which is introduced in this section.

Let $w = a_0 \cdots a_n$ and V_w be specified as in Section 2, and let w' be some substring of w. (The specification of w' is not of our concern in this section.) Let also $h_1, h_2, h_3 \in V_w$. We are interested in computations of the parser processing the substring w' and having the form $c_0, c_1, \ldots, c_m$, $m \geq 1$, that satisfy both of the following conditions, exemplified in Figure 2.

- For some sequence of nodes σ with $|\sigma| \geq 0$, the stack associated with c_0 has the form $\sigma|h_1$ and the stack associated with c_m has the form $\sigma|h_2|h_3$.
- For each intermediate configuration c_i, $1 \leq i \leq m-1$, the stack associated with c_i has the form $\sigma\sigma_i$, where σ_i is a sequence of nodes with $|\sigma_i| \geq 2$.

An important property of the above definition needs to be discussed here, which is at the heart of the polynomial-time algorithm in the next section. If in $c_0, c_1, \ldots, c_m$ we replace σ with a different sequence σ', we obtain a valid computation for w' constructing exactly the same dependencies as the original computation. To see this, let $c_{i-1} \vdash_{\tau_i} c_i$ for each i with $1 \leq i \leq m$. Then $\vdash_{\tau_1}$ must be a shift, otherwise $|\sigma_1| \geq 2$ would be violated. Consider now a transition $\vdash_{\tau_i}$ with $2 \leq i \leq m$ that builds some dependency. From $|\sigma_i| \geq 2$ we derive $|\sigma_{i-1}| \geq 3$. We can easily check from Figure 1 that none of the nodes in σ can be involved in the constructed dependency.

Intuitively, the above property asserts that the sequence of transitions $\vdash_{\tau_1}, \vdash_{\tau_2}, \ldots, \vdash_{\tau_m}$ can be applied to parse substring w' independently of the context σ. This suggests that we can group into an equivalence class all the computations satisfying the conditions above, for different values of σ. We indicate such class by means of the tuple $[h_1, h_2 h_3]$, called **item**. It is easy to see that each item represents an exponential number of computations. In the next section we will show how we can process items with the purpose of obtaining an efficient computation for dynamic oracles.

5 Dynamic Oracle Algorithm

Our algorithm takes as input a gold tree t_G for string w and a parser configuration $c = (\sigma, \beta, A)$ relative to w, specified as in Section 2. We assume that t_G can be parsed by the non-projective parser of Section 3 starting from the initial configuration.

5.1 Basic Idea

The algorithm consists of two separate stages, informally discussed in what follows. In the first stage we identify some tree fragments of t_G that can be constructed by the parser after reaching configuration c, in a way that does not depend on the content of σ. This means that these fragments can be precomputed by looking only into β. Furthermore, since these fragments are subtrees of t_G, their computation has no effect on the overall loss of a computation on w.

For each fragment t with the above properties, we replace all the nodes in β that are also nodes of t with the root node of t itself. The result of the first stage is therefore a new node sequence shorter than β, which we call the reduced buffer β_R.

In the second stage of the algorithm we use a variant of the tabular method developed by Cohen et al. (2011), which was originally designed to simulate all computations of the parser in Section 3 on an input string w. We run the above method on the concatenation of the stack and the reduced buffer, with some additional constraints that restrict the search space in two respects. First, we visit only those computations of the parser that step through configuration c. Second, we reach only those dependency trees that contain all the tree fragments precomputed in the first stage. We can show that such search space always contains at least one dependency tree with the desired loss, which we then retrieve performing a Viterbi search.

5.2 Preprocessing of the Buffer

Let t be a complete subtree of t_G, having root node k in β. Consider the following two conditions, defined on t.

- *Bottom-up completeness*: No arc $i \rightarrow j$ in t is such that i is a node in β, $i \neq k$, and j is a node in σ.
- *Zero gap-degree*: The nodes of t that are in β form a (contiguous) substring of w.

We claim that if t satisfies the above conditions, then we can safely reduce the nodes of t appearing in β, replacing them with node k. We only report here an informal discussion of this claim, and omit a formal proof.

As a first remark, recall that our parser implements a purely bottom-up strategy. This means that after a tree has been constructed, all of its nodes but the root are removed from the parser configuration. Then the Bottom-up completeness condition guarantees that if we remove from β all nodes of t but k, the nodes of t that are in σ can still be processed in a way that does not affect the loss, since their parent must be either k or a node that is neither in β nor in σ. Note that the nodes of t that are neither in β nor in σ are irrelevant to the precomputation of t from β, since these nodes have already been attached and are no longer available to the parser.

As a second remark, the Zero gap-degree condition guarantees that the span of t over the nodes of β is not interleaved by nodes that do not belong to t. This is also an important requirement for the precomputation of t from β, since a tree fragment having a discontinuous span over β might not be constructable independently of σ. More specifically, parsing such fragment implies dealing with the nodes in the discontinuities, and this might require transitions involving nodes from σ.

We can now use the sufficient condition above to compute β_R. We process β from left to right. For each node k, we can easily test the Bottom-up completeness condition and the Zero gap-degree condition for the complete subtree t of t_G rooted at k, and perform the reduction if both conditions are satisfied. Note that in this process a node k resulting from the reduction of t might in turn be removed from β if, at some later point, we reduce a supertree of t.

5.3 Computation of the Loss

We describe here our dynamic programming algorithm for the computation of the loss of an input configuration c. We start with some additional notation. Let $\gamma = \sigma\beta_R$ be the concatenation of σ and β_R, which we treat as a string of nodes. For integers i with $0 \leq i \leq |\gamma| - 1$, we write $\gamma[i]$ to denote the $(i+1)$-th node of γ. Let also $\ell = |\sigma|$. Symbol ℓ is used to mark the boundary between the stack and the reduced buffer in γ, thus $\gamma[i]$ with $i < \ell$ is a node of σ, while $\gamma[i]$ with $i \geq \ell$ is a node of β_R.

Algorithm 1 computes the loss of c by processing the sequence γ in a way quite similar to the

standard nested loop implementation of the CKY parser for context-free grammars (Hopcroft et al., 2006). The algorithm uses a two-dimensional array $\mathcal{T}$ whose indexes range from 0 to $|\gamma| = \ell + |\beta_R|$, and only the cells $\mathcal{T}[i,j]$ with $i < j$ are filled.

We view each $\mathcal{T}[i,j]$ as an association list whose keys are items $[h_1, h_2 h_3]$, defined in the context of the substring $\gamma[i] \cdots \gamma[j-1]$ of γ; see Section 4. The value stored at $\mathcal{T}[i,j]([h_1, h_2 h_3])$ is the minimum loss contribution due to the computations represented by $[h_1, h_2 h_3]$. For technical reasons, we assume that our parser starts with a symbol $\$ \notin V_w$ in the stack, denoting the bottom of the stack.

We initialize the table by populating the cells of the form $\mathcal{T}[i, i+1]$ with information about the trivial computations consisting of a single $\vdash_{\mathsf{sh}}$ transition that shifts the node $\gamma[i]$ into the stack. These computations are known to have zero loss contribution, because a $\vdash_{\mathsf{sh}}$ transition does not create any arcs. In the case where the node $\gamma[i]$ belongs to σ, i.e., $i < \ell$, we assign loss contribution 0 to the entry $\mathcal{T}[i, i+1]([\gamma[i-1], \gamma[i-1]\gamma[i]])$ (line 3 of Algorithm 1), because $\gamma[i]$ is shifted with $\gamma[i-1]$ at the top of the stack. On the other hand, if $\gamma[i]$ is in β, i.e., $i \geq \ell$, we assign loss contribution 0 to several entries in $\mathcal{T}[i, i+1]$ (line 6) because, at the time $\gamma[i]$ is shifted, the content of the stack depends on the transitions executed before that point.

After the above initialization, we consider pairs of contiguous substrings $\gamma[i] \cdots \gamma[k-1]$ and $\gamma[k] \cdots \gamma[j-1]$ of γ. At each inner iteration of the nested loops of lines 7-11 we update cell $\mathcal{T}[i,j]$ based on the content of the cells $\mathcal{T}[i,k]$ and $\mathcal{T}[k,j]$. We do this through the procedure ProcessCell$(\mathcal{T}, i, k, j)$, which considers all pairs of keys $[h_1, h_2 h_3]$ in $\mathcal{T}[i,k]$ and $[h_3, h_4 h_5]$ in $\mathcal{T}[k,j]$. Note that we require the index h_3 to match between both items, meaning that their computations can be concatenated. In this way, for each reduce transition τ in our parser, we compute the loss contribution for a new piece of computation defined by concatenating a computation with minimum loss contribution in the first item and a computation with minimum loss contribution in the second item, followed by the transition τ. The fact that the new piece of computation can be represented by an item is exemplified in Figure 3 for the case $\tau = \vdash_{\mathsf{ra}_2}$.

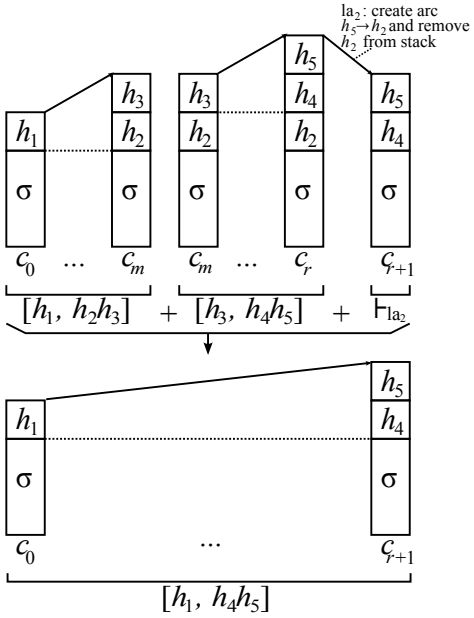


Figure 3: Concatenation of two computations/items and transition $\vdash_{\mathsf{ra}_2}$, resulting in a new computation/item.

The computed loss contribution is used to update the entry in $\mathcal{T}[i,j]$ corresponding to the item associated with the new computation. Observe how the loss contribution provided by the arc created by τ is computed by the δ_G function at lines 17, 20, 23 and 26, which is defined as:

$$\delta_G(i \rightarrow j) \;=\; \begin{cases} 0, & \text{if } i \rightarrow j \text{ is in } t_G; \\ 1, & \text{otherwise.} \end{cases} \quad (4)$$

We remark that the nature of our problem allows us to apply several shortcuts and optimizations that would not be possible in a setting where we actually needed to parse the string γ. First, the range of variable i in the loop in line 8 starts at $\max\{0, \ell - d\}$, rather than at 0, because we do not need to combine pairs of items originating from nodes in σ below the topmost node, as the items resulting from such combinations correspond to computations that do not contain our input configuration c. Second, when we have set values for i such that $i+2 < \ell$, we can omit calling ProcessCell for values of the parameter k ranging from $i+2$ to $\ell-1$, as those calls would use as their input one of the items described above, which are not of interest. Finally, when processing substrings that are entirely in β_R ($i \geq \ell$) we can restrict the transitions that we explore to those that generate arcs that either are in the gold tree t_G, or have a parent node which is not present in γ (see conditions in

Algorithm 1 Computation of the loss function

1: $\mathcal{T}[0,1]([\$,\$0]) \leftarrow 0$ ▷ shift node 0 on top of empty stack symbol \$
2: **for** $i \leftarrow 1$ **to** $\ell - 1$ **do**
3: $\quad \mathcal{T}[i,i+1]([\gamma[i-1],\gamma[i-1]\gamma[i]]) \leftarrow 0$ ▷ shift node $\gamma[i]$ with $\gamma[i-1]$ on top of the stack
4: **for** $i \leftarrow \ell$ **to** $|\gamma|$ **do**
5: $\quad$ **for** $h \leftarrow 0$ **to** $i-1$ **do**
6: $\quad\quad \mathcal{T}[i,i+1]([\gamma[h],\gamma[h]\gamma[i]]) \leftarrow 0$ ▷ shift node $\gamma[i]$ with $\gamma[h]$ on top of the stack
7: **for** $d \leftarrow 2$ **to** $|\gamma|$ **do** ▷ consider substrings of length d
8: $\quad$ **for** $i \leftarrow \max\{0, \ell - d\}$ **to** $|\gamma| - d$ **do** ▷ i = beginning of substring
9: $\quad\quad j \leftarrow i + d$ ▷ $j-1$ = end of substring
10: $\quad\quad$ PROCESSCELL($\mathcal{T}, i, i+1, j$) ▷ We omit the range $k = i+2$ to $\max\{i+2, \ell\} - 1$
11: $\quad\quad$ **for** $k \leftarrow \max\{i+2, \ell\}$ **to** j **do** ▷ factorization of substring at k
12: $\quad\quad\quad$ PROCESSCELL($\mathcal{T}, i, k, j$)
13: **return** $\mathcal{T}[0,|\gamma|]([\$,\$0]) + \sum_{i\in[0,\ell-1]} \mathcal{L}_c(\sigma[i], t_G)$

14: **procedure** PROCESSCELL($\mathcal{T}, i, k, j$)
15: $\quad$ **for each key** $[h_1, h_2h_3]$) **defined in** $\mathcal{T}[i,k]$ **do**
16: $\quad\quad$ **for each key** $[h_3, h_4h_5]$) **defined in** $\mathcal{T}[k,j]$ **do** ▷ h_3 must match between the two entries
17: $\quad\quad\quad loss_{\mathsf{la}} \leftarrow \mathcal{T}[i,k]([h_1,h_2h_3]) + \mathcal{T}[k,j]([h_3,h_4h_5]) + \delta_G(h_5 \rightarrow h_4)$
18: $\quad\quad\quad$ **if** $(i < \ell) \vee \delta_G(h_5 \rightarrow h_4) = 0 \vee (h_5 \notin \gamma)$ **then**
19: $\quad\quad\quad\quad \mathcal{T}[i,j]([h_1,h_2h_5]) \leftarrow \min\{loss_{\mathsf{la}}, \mathcal{T}[i,j]([h_1,h_2h_5])\}$ ▷ cell update $\vdash_{\mathsf{la}}$
20: $\quad\quad\quad loss_{\mathsf{ra}} \leftarrow \mathcal{T}[i,k]([h_1,h_2h_3]) + \mathcal{T}[k,j]([h_3,h_4h_5]) + \delta_G(h_4 \rightarrow h_5)$
21: $\quad\quad\quad$ **if** $(i < \ell) \vee \delta_G(h_4 \rightarrow h_5) = 0 \vee (h_4 \notin \gamma)$ **then**
22: $\quad\quad\quad\quad \mathcal{T}[i,j]([h_1,h_2h_4]) \leftarrow \min\{loss_{\mathsf{ra}}, \mathcal{T}[i,j]([h_1,h_2h_4])\}$ ▷ cell update $\vdash_{\mathsf{ra}}$
23: $\quad\quad\quad loss_{\mathsf{la}_2} \leftarrow \mathcal{T}[i,k]([h_1,h_2h_3]) + \mathcal{T}[k,j]([h_3,h_4h_5]) + \delta_G(h_5 \rightarrow h_2)$
24: $\quad\quad\quad$ **if** $(i < \ell) \vee \delta_G(h_5 \rightarrow h_2) = 0 \vee (h_5 \notin \gamma)$ **then**
25: $\quad\quad\quad\quad \mathcal{T}[i,j]([h_1,h_4h_5]) \leftarrow \min\{loss_{\mathsf{la}_2}, \mathcal{T}[i,j]([h_1,h_4h_5])\}$ ▷ cell update $\vdash_{\mathsf{la}_2}$
26: $\quad\quad\quad loss_{\mathsf{ra}_2} \leftarrow \mathcal{T}[i,k]([h_1,h_2h_3]) + \mathcal{T}[k,j]([h_3,h_4h_5]) + \delta_G(h_2 \rightarrow h_5)$
27: $\quad\quad\quad$ **if** $(i < \ell) \vee \delta_G(h_2 \rightarrow h_5) = 0 \vee (h_2 \notin \gamma)$ **then**
28: $\quad\quad\quad\quad \mathcal{T}[i,j]([h_1,h_2h_4]) \leftarrow \min\{loss_{\mathsf{ra}_2}, \mathcal{T}[i,j]([h_1,h_2h_4])\}$ ▷ cell update $\vdash_{\mathsf{ra}_2}$

lines 18, 21, 24, 27), because we know that incorrectly attaching a buffer node as a dependent of another buffer node, when the correct head is available, can never be an optimal decision in terms of loss.

Once we have filled the table $\mathcal{T}$, the loss for the input configuration c can be obtained from the value of the entry $\mathcal{T}[0,|\gamma|]([\$,\$0])$, representing the minimum loss contribution among computations that reach the input configuration c and parse the whole input string. To obtain the total loss, we add to this value the loss contribution accumulated by the dependency trees with root in the stack σ of c. This is represented in Algorithm 1 as $\sum_{i\in[0,\ell-1]} \mathcal{L}_c(\sigma[i], t_G)$, where $\mathcal{L}_c(\sigma[i], t_G)$ is the count of the descendants of $\sigma[i]$ (the $(i+1)$-th element of σ) that had been assigned the wrong head by the parser with respect to t_G.

5.4 Sample Run

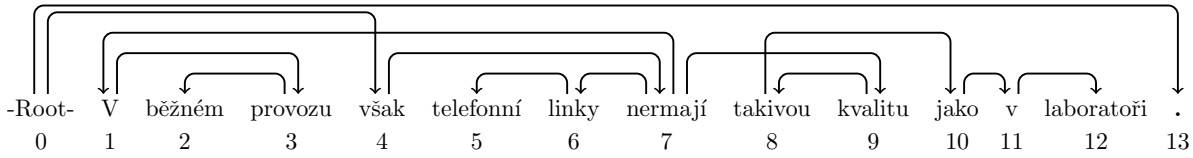
Consider the Czech sentence and the gold dependency tree t_G shown in Figure 4(a). Given the configuration $c = (\sigma, \beta, A)$ where $\sigma = [0,1,3,4]$, $\beta = [5,\ldots,13]$ and $A = \{3 \rightarrow 2\}$, we trace the two stages of the algorithm.

Preprocessing of the buffer The complete subtree rooted at node 7 satisfies the Bottom-up completeness and the Zero gap-degree conditions in Section 5.2, so the nodes $5,\ldots,12$ in β can be replaced with the root 7. Note that all the nodes in the span $5,\ldots,12$ have all their (gold) dependents in that span, with the exception of the root 7, with its dependent node 1 still in the stack. No other reduction is possible, and we have $\beta_R = [7,13]$. The corresponding fragment of t_G is represented in Figure 4(b).

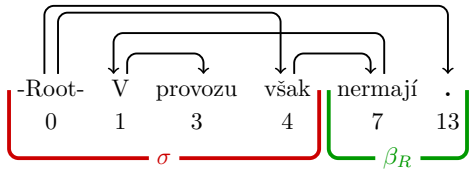
Computation of the loss Let $\gamma = \sigma\beta_R$. Algorithm 1 builds the two-dimensional array $\mathcal{T}$ in Figure 4(c). Each cell $\mathcal{T}[i,j]$ contains an association list, whose (key:value) pairs map items to their loss contribution. Figure 4(c) only shows the pairs involved in the minimum-loss computation.

Lines 1-6 of Algorithm 1 initialize the cells in the diagonal, $\mathcal{T}[0,1],\ldots,\mathcal{T}[5,6]$. The boundary between stack and buffer is $\ell = 4$, thus cells $\mathcal{T}[0,1]$, $\mathcal{T}[1,2]$, and $\mathcal{T}[2,3]$ contain only one element, while $\mathcal{T}[3,4]$, $\mathcal{T}[4,5]$ and $\mathcal{T}[5,6]$ contain as many as the previous elements in γ, although not all of them are shown in the figure.

Lines 7-12 fill the superdiagonals until $\mathcal{T}[0,6]$ is reached. The cells $\mathcal{T}[0,2]$, $\mathcal{T}[0,3]$ and $\mathcal{T}[1,3]$



(a) Non-projective dependency tree from the Prague Dependency Treebank.



(b) Fragment of dependency tree in (a) after buffer reduction.

i \ j	1	2	3	4	5	6
0	[$,$ 0]:0	∅	∅	...	[$,$ 0]:1	[$,$ 0]:1
1		[0,0 1]:0	∅	...	[0,0 4]:1	...
2			[1,1 3]:0	[1,1 4]:1	[1,4 7]:1	...
3				[3,3 4]:0	[3,4 7]:1	...
4					[4,4 7]:0	...
5						[0,0 13]:0

(c) Relevant portion of $\mathcal{T}$ computed by Algorithm 1, with the loss of c in the yellow entry.

Figure 4: Example of loss computation given the sentence in (a) and considering a configuration c with $\sigma = [0, 1, 3, 4]$ and $\beta = [5, \ldots, 13]$.

are left empty because $\ell = 4$. Once $\mathcal{T}[0, 6]$ is calculated, it contains only the entry with key $[\$, \$, 0]$, with the associated value 1 representing the minimum number of wrong arcs that the parsing algorithm has to build to reach a final configuration from c. Then, Line 13 retrieves the loss of the configuration, computed as the sum of $\mathcal{T}[0, 6]([\$, \$, 0])$ with the term $\mathcal{L}_c$, representing the erroneous arcs made *before* reaching c.

Note that in our example the loss of c is 1, even though $\mathcal{L}_c = 0$, meaning that there are no wrong arcs in A. Indeed, given c, there is no single computation that builds all the remaining arcs in t_G. This is reflected in $\mathcal{T}$, where the path to reach the item with minimum loss has to go through either $\mathcal{T}[3, 5]$ or $\mathcal{T}[2, 4]$, which implies building the erroneous arc $(w_7 \rightarrow w_3)$ or $(w_4 \rightarrow w_3)$, respectively.

6 Computational Analysis

The first stage of our algorithm can be easily implemented in time $\mathcal{O}(|\beta|\,|t_G|)$, where $|t_G|$ is the number of nodes in t_G, which is equal to the length n of the input string.

For the worst-case complexity of the second stage (Algorithm 1), note that the number of cell updates made by calling ProcessCell$(\mathcal{T}, i, k, j)$ with $k < \ell$ is $\mathcal{O}(|\sigma|^3\,|\gamma|^2\,|\beta_R|)$. This is because these updates can only be caused by procedure calls on line 10 (as those on line 12 always set $k \geq \ell$) and therefore the index k always equals $i + 1$, while h_2 must equal h_1 because the item $[h_1, h_2 h_3]$ is one of the initial items created on line 3. The variables i, h_1 and h_3 must index nodes on the stack σ as they are bounded by k, while j ranges over β_R and h_4 and h_5 can refer to nodes either on σ or on β_R.

On the other hand, the number of cell updates triggered by calls to ProcessCell such that $k \geq \ell$ is $\mathcal{O}(|\gamma|^4 |\beta_R|^4)$, as they happen for four indices referring to nodes of β_R (k, j, h_4, h_5) and four indices that can range over σ or β_R (i, h_1, h_2, h_3).

Putting everything together, we conclude that the overall complexity of our algorithm is $\mathcal{O}(|\beta|\,|t_G| + |\sigma|^3\,|\gamma|^2\,|\beta_R| + |\gamma|^4\,|\beta_R|^4)$.

In practice, quantities $|\sigma|$, $|\beta_R|$ and $|\gamma|$ are significantly smaller than n, providing reasonable training times as we will see in Section 7. For instance, when measured on the Czech treebank, the average value of $|\sigma|$ is 7.2, with a maximum of 87. Even more interesting, the average value of $|\beta_R|$ is 2.6, with a maximum of 23. Comparing this to the average and maximum values of $|\beta|$, 11 and 192, respectively, we see that the buffer reduction is crucial in reducing training time.

Note that, when expressed as a function of n, our dynamic oracle has a worst-case time complexity of $\mathcal{O}(n^8)$. This is also the time complexity of the dynamic programming algorithm of Cohen et al. (2011) we started with, simulating all computations of our parser. In contrast, the dynamic oracle of Goldberg et al. (2014) for the projective case achieves a time complexity of $\mathcal{O}(n^3)$ from the dynamic programming parser by Kuhlmann et al. (2011) running in time $\mathcal{O}(n^5)$.

The reason why we do not achieve any asymptotic improvement is that some helpful properties that hold with projective trees are no longer satisfied in the non-projective case. In the projective (arc-standard) case, subtrees that are in the buffer can be completely reduced. As a consequence, each oracle step always combines an inferred entry in the table with either a node from the stack or a node from the reduced buffer, asymptotically reducing the time complexity. However, in the non-projective (Attardi) case, subtrees in the buffer can not always be completely reduced, for the reasons mentioned in the second-to-last paragraph of Section 5.2. As a consequence, the oracle needs to make cell updates in a more general way, which includes linking pairs of elements in the reduced buffer or pairs of inferred entries in the table.

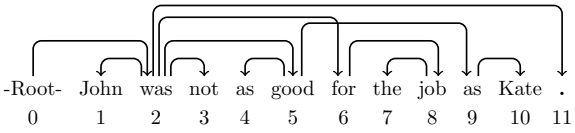


Figure 5: Non-projective dependency tree adapted from the Penn Treebank.

An example of why this is needed is provided by the gold tree in Figure 5. Assume a configuration $c = (\sigma, \beta, A)$ where $\sigma = [0, 1, 2, 3, 4]$, $\beta = [5, \ldots, 11]$, and $A = \emptyset$. It is easy to see that the loss of c is greater than zero, since the gold tree is not reachable from c: parsing the subtree rooted at node 5 requires shifting 6 into the stack, and this makes it impossible to build the arcs $2 \rightarrow 5$ and $2 \rightarrow 6$. However, if we reduced the subtree in the buffer with root 5, we would incorrectly obtain a loss of 0, as the resulting tree is parsable if we start with $\vdash_{\mathsf{sh}}$ followed by $\vdash_{\mathsf{la}}$ and $\vdash_{\mathsf{ra}_2}$. Note that there is no way of knowing whether it is safe to reduce the subtree rooted at 5 without using non-local information. For example, the arc $2 \rightarrow 6$ is crucial here: if 6 depended on 5 or 4 instead, the loss would be zero. These complications are not found in the projective case, allowing for the mentioned asymptotic improvement.

7 Experimental Evaluation

For comparability with previous work on dynamic oracles, we follow the experimental settings reported by Goldberg et al. (2014) for their arc-standard dynamic oracle. In particular, we use the same training algorithm, features, and root node position. However, we train the model for 20 iterations rather than 15, as the increased search space and spurious ambiguity of Attardi's non-projective parser implies that more iterations are required to converge to a stable model. A more detailed description of the experimental settings follows.

	static		dynamic	
	UAS	LAS	UAS	LAS
Arabic	80.90	71.56	**82.23**	**72.63**
Basque	**75.96**	**66.74**	74.32	65.59
Catalan	**90.55**	**85.20**	89.94	84.96
Chinese	84.72	79.93	**85.34**	**81.00**
Czech	79.83	72.69	**82.08**	**74.44**
English	85.52	84.46	**87.38**	**86.40**
Greek	79.84	72.26	**81.55**	**74.14**
Hungarian	**78.13**	**68.90**	76.27	68.14
Italian	83.08	78.94	**84.43**	**80.45**
Turkish	**79.57**	69.44	79.41	**70.32**
Bulgarian	**89.46**	**85.99**	89.32	85.92
Danish	85.58	81.25	**86.03**	**81.59**
Dutch	79.05	75.69	**80.13**	**77.22**
German	88.34	86.48	**88.86**	**86.94**
Japanese	93.06	91.64	**93.56**	**92.18**
Portuguese	84.80	81.38	**85.36**	**82.10**
Slovene	76.33	68.43	**78.20**	**70.22**
Spanish	79.88	76.84	**80.25**	**77.45**
Swedish	**87.26**	**82.77**	87.24	82.49
PTB	89.55	87.18	**90.47**	**88.18**

Table 1: Unlabelled Attachment Score (UAS) and Labelled Attachment Score (LAS) using a *static* and a *dynamic* oracle. Evaluation on CoNLL 2007 (first block) and CoNLL 2006 (second block) datasets is carried out including punctuation, evaluation on the Penn Treebank excludes it.

7.1 Experimental Setup

Training We train a global linear model using the averaged perceptron algorithm and a labelled version of the parser described in Section 3. We perform on-line training using the oracle defined in Section 5: at each parsing step, the model's weights are updated if the predicted transition results into an increase in configuration loss, but the process continues by following the predicted transition independently of the loss increase.

As our baseline we train the model using the static oracle defined by (Cohen et al., 2012). This oracle follows a canonical computation that creates arcs as soon as possible, and prioritizes the $\vdash_{\mathsf{la}}$ transition over the $\vdash_{\mathsf{la}_2}$ transition in situations

where both create a gold arc. The static oracle is not able to deal with configurations that cannot reach the gold dependency tree, so we constrain the training algorithm to follow the zero-loss transition provided by the oracle.

While this version of Attardi's parser has been shown to cover the vast majority of non-projective sentences in several treebanks (Attardi, 2006; Cohen et al., 2012), there still are some sentences which are not parsable. These sentences are skipped during training, but not during test and evaluation of the model.

Datasets We evaluate the parser performance over CoNLL 2006 and CoNLL 2007 datasets. If a language is present in both datasets, we use the latest version. We also include results over the Penn Treebank (PTB) (Marcus et al., 1993) converted to Stanford basic dependencies (De Marneffe et al., 2006). For the CoNLL datasets we use the provided part-of-speech tags and the standard training/test partition; for the PTB we use automatically assigned tags, we train on sections 2-21 and test on section 23.

7.2 Results and Analysis

In Table 1 we report the unlabelled (UAS) and labelled (LAS) attachment scores for the static and the dynamic oracles. Each figure is an average over the accuracy provided by 5 models trained with the same setup but using a different random seed. The seed is only used to shuffle the sentences in random order during each iteration of training.

Our results are consistent with the results reported by Goldberg and Nivre (2013) and Goldberg et al. (2014). For most of the datasets, we obtain a relevant improvement in both UAS and LAS. For Dutch, Czech and German, we achieve an error reduction of 5.2%, 11.2% and 4.5%, respectively. Exceptions to this general trend are Swedish and Bulgarian, where the accuracy differences are negligible, and the Basque, Catalan and Hungarian datasets, where the performance actually decreases.

If instead of testing on the standard test sets we use 10-fold cross-validation and average the resulting accuracies, we obtain improvements for all languages in Table 1 but Basque and Hungarian. More specifically, measured (UAS, LAS) pairs for Swedish are (86.85, 82.17) with dynamic oracle against (86.6, 81.93) with static oracle; for Bulgarian (88.42, 83.91) against (88.20, 83.55); and for Catalan (88.33, 83.64) against (88.06, 83.13). This suggests that the negligible or unfavourable results in Table 1 for these languages are due to statistical variability given the small size of the test sets.

As for Basque, we measure (75.54, 67.58) against (76.77, 68.20); similarly, for Hungarian we measure (75.66, 67.66) against (77.22, 68.42). Unfortunately, we have no explanation for these performance decreases, in terms of the typology of the non-projective patterns found in these two datasets. Note that Goldberg et al. (2014) also observed a performance decrease on the Basque dataset in the projective case, although not on Hungarian.

The parsing times measured in our experiments for the static and the dynamic oracles are the same, since the oracle algorithm is only used during the training stage. Thus the reported improvements in parsing accuracy come at no extra cost for parsing time. In the training stage, the extra processing needed to compute the loss and to explore paths that do not lead to a gold tree made training about 4 times slower, on average, for the dynamic oracle model. This confirms that our oracle algorithm is fast enough to be of practical interest, in spite of its relatively high worst-case asymptotic complexity.

8 Conclusions

We have presented what, to our knowledge, are the first experimental results for a transition-based non-projective parser trained with a dynamic oracle. We have also shown significant accuracy improvements on many languages over a static oracle baseline.

The general picture that emerges from our approach is that dynamic programming algorithms originally conceived for the simulation of transition-based parsers can effectively be used in the development of polynomial-time algorithms for dynamic oracles.

Acknowledgments

The first author has been partially funded by Ministerio de Economía y Competitividad/FEDER (Grant TIN2010-18552-C03-02) and by Xunta de Galicia (Grant CN2012/008). The third author has been partially supported by MIUR under project PRIN No. 2010LYA9RH_006.

References

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*, pages 166–170, New York, USA.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1052–1062, Sofia, Bulgaria, August. Association for Computational Linguistics.

Shay B. Cohen, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Exact inference for generative probabilistic non-projective dependency parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1245, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Shay B. Cohen, Carlos Gómez-Rodríguez, and Giorgio Satta. 2012. Elimination of spurious ambiguity in transition-based dependency parsing. *CoRR*, abs/1206.6735.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, volume 6, pages 449–454.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proc. of the 24th COLING*, Mumbai, India.

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

Yoav Goldberg, Francesco Sartorio, and Giorgio Satta. 2014. A tabular method for dynamic oracles in transition-based parsing. *Transactions of the Association for Computational Linguistics*, 2(April):119–130.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2006. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

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

Marco Kuhlmann and Joakim Nivre. 2010. Transition-based techniques for non-projective dependency parsing. *Northern European Journal of Language Technology*, 2(1):1–19.

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

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

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Workshop on Parsing Technologies (IWPT)*, pages 149–160, Nancy, France.

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

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

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA, June. Association for Computational Linguistics.

Ambiguity Resolution for Vt-N Structures in Chinese

Yu-Ming Hsieh[1,2] **Jason S. Chang**[2] **Keh-Jiann Chen**[1]
[1] Institute of Information Science, Academia Sinica, Taiwan
[2] Department of Computer Science, National Tsing-Hua University, Taiwan
morris@iis.sinica.edu.tw, jason.jschang@gmail.com
kchen@iis.sinica.edu.tw

Abstract

The syntactic ambiguity of a transitive verb (Vt) followed by a noun (N) has long been a problem in Chinese parsing. In this paper, we propose a classifier to resolve the ambiguity of Vt-N structures. The design of the classifier is based on three important guidelines, namely, adopting linguistically motivated features, using all available resources, and easy integration into a parsing model. The linguistically motivated features include semantic relations, context, and morphological structures; and the available resources are treebank, thesaurus, affix database, and large corpora. We also propose two learning approaches that resolve the problem of data sparseness by auto-parsing and extracting relative knowledge from large-scale unlabeled data. Our experiment results show that the Vt-N classifier outperforms the current PCFG parser. Furthermore, it can be easily and effectively integrated into the PCFG parser and general statistical parsing models. Evaluation of the learning approaches indicates that world knowledge facilitates Vt-N disambiguation through data selection and error correction.

1 Introduction

In Chinese, the structure of a transitive verb (Vt) followed by a noun (N) may be a verb phrase (VP), a noun phrase (NP), or there may not be a dependent relation, as shown in (1) below. In general, parsers may prefer VP reading because a transitive verb followed by a noun object is normally a VP structure. However, Chinese verbs can also modify nouns without morphological inflection, e.g., 養殖/*farming* 池/*pond*. Consequently, parsing Vt-N structures is difficult because it is hard to resolve such ambiguities without prior knowledge. The following are some typical examples of various Vt-N structures:

1)
解決/*solve* 問題/*problem* → VP
解決/*solving* 方案/*method* → NP
解決/*solve* 人類/*mankind* (問題/*problem*)→None

To find the most effective disambiguation features, we need more information about the Vt-N → NP construction and the semantic relations between Vt and N. Statistical data from the Sinica Treebank (Chen et al., 2003) indicates that 58% of Vt-N structures are verb phrases, 16% are noun phrases, and 26% do not have any dependent relations. It is obvious that the semantic relations between a Vt-N structure and its context information are very important for differentiating between dependent relations. Although the verb-argument relation of VP structures is well understood, it is not clear what kind of semantic relations result in NP structures. In the next sub-section, we consider three questions: What sets of nouns accept verbs as their modifiers? Is it possible to identify the semantic types of such pairs of verbs and nouns? What are their semantic relations?

1.1 Problem Analysis

Analysis of the instances of NP(Vt-N) structures in the Sinica Treebank reveals the following four types of semantic structures, which are used in the design of our classifier.

Type 1. Telic(Vt) + Host(N): Vt denotes the telic function (purpose) of the head noun N, e.g.,

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 928–937,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

研究/*research* 工具/*tool*; 探測/*explore* 機/*machine*; 賭/*gamble* 館/*house*; 搜尋/*search* 程式/*program*. The telic function must be a salient property of head nouns, such as tools, buildings, artifacts, organizations and people. To identify such cases, we need to know the types of nouns which take telic function as their salient property. Furthermore, many of the nouns are monosyllabic words, such as 員/*people*, 器/*instruments*, 機/*machines*.

Type 2. Host-Event(Vt) + Attribute(N): Head nouns are attribute nouns that denote the attributes of the verb, e.g., 研究/*research* 方法/*method* (*method of research*); 攻擊/*attack* 策略/*strategy* (*attacking strategy*); 書寫/*write* 內容/*context* (*context of writing*); 賭/*gamble* 規/*rule* (*gambling rules*). An attribute noun is a special type of noun. Semantically, attribute nouns denote the attribute types of objects or events, such as *weight*, *color*, *method*, and *rule*. Syntactically, attribute nouns do not play adjectival roles (Liu, 2008). By contrast, object nouns may modify nouns. The number of attributes for events is limited. If we could discover all event-attribute relations, then we can solve this type of construction.

Type 3. Agentive + Host: There is only a limited number of such constructions and the results of the constructions are usually ambiguous, e.g., 炒飯/*fried rice* (NP), 叫聲/*shouting sound*. The first example also has the VP reading.

Type 4. Apposition + Affair: Head nouns are event nouns and modifiers are verbs of apposition events, e.g. 追撞/*collide* 事故/*accident*, 破壞/*destruct* 運動/*movement*, 憤恨/*hate* 行為/*behavior*. There is finite number of event nouns.

Furthermore, when we consider verbal modifiers, we find that verbs can play adjectival roles in Chinese without inflection, but not all verbs play adjectival roles. According to Chang et al. (2000) and our observations, adjectival verbs are verbs that denote event types rather than event instances; that is, they denote a class of events which that are concepts in an upper-level ontology. One important characteristic of adjectival verbs is that they have conjunctive morphological structures, i.e., the words are conjunct with two nearly synonymous verbs, e.g., 研/*study* 究/*search* (*research*), 探/*explore* 測/*detect* (*explore*), and 搜/*search* 尋/*find* (*search*). Therefore, we need a morphological classifier that can detect the conjunctive morphological structure of a verb by checking the semantic parity of two morphemes of the verb.

Based on our analysis, we designed a Vt-N classifier that incorporates the above features to solve the problem. However, there is a data sparseness problem because of the limited size of the current Treebank. In other words, Treebank cannot provide enough training data to train a classifier properly. To resolve the problem, we should mine useful information from all available resources.

The remainder of this paper is organized as follows. Section 2 provides a review of related works. In Section 3, we describe the disambiguation model with our selected features, and introduce a strategy for handling unknown words. We also propose a learning approach for a large-scale unlabeled corpus. In Section 4, we report the results of experiments conducted to evaluate the proposed Vt-N classifier on different feature combinations and learning approaches. Section 5 contains our concluding remarks.

2 Related Work

Most works on V-N structure identification focus on two types of relation classification: modifier-head relations and predicate-object relations (Wu, 2003; Qiu, 2005; Chen, 2008; Chen et al., 2008; Yu et al., 2008). They exclude the independent structure and conjunctive head-head relation, but the cross-bracket relation does exist between two adjacent words in real language. For example, if "遍佈/*all over* 世界/*world* " was included in the short sentence "遍佈/*all over* 世界/*world* 各國/*countries*", it would be an independent structure. A conjunctive head-head relation between a verb and a noun is rare. However, in the sentence "服務 設備 都 甚 周到" (Both service and equipment are very thoughtful.), there is a conjunctive head-head relation between the verb 服務/*service* and the noun 設備/*equipment*. Therefore, we use four types of relations to describe the V-N structures in our experiments. The symbol 'H/X' denotes a predicate-object relation; 'X/H' denotes a modifier-head relation; 'H/H' denotes a conjunctive head-head relation; and 'X/X' denotes an independent relation.

Feature selection is an important task in V-N disambiguation. Hence, a number of studies have suggested features that may help resolve the ambiguity of V-N structures (Zhao and Huang, 1999; Sun and Jurafsky, 2003; Chiu et al., 2004; Qiu, 2005; Chen, 2008). Zhao and Huang used lexicons, semantic knowledge, and word length in-

formation to increase the accuracy of identification. Although they used the Chinese thesaurus CiLin (Mei et al., 1983) to derive lexical semantic knowledge, the word coverage of CiLin is insufficient. Moreover, none of the above papers tackle the problem of unknown words. Sun and Jurafsky exploit the probabilistic rhythm feature (i.e., the number of syllables in a word or the number of words in a phrase) in their shallow parser. Their results show that the feature improves the parsing performance, which coincides with our analysis in Section 1.1. Chiu et al.'s study shows that the morphological structure of verbs influences their syntactic behavior. We follow this finding and utilize the morphological structure of verbs as a feature in the proposed Vt-N classifier. Qiu's approach uses an electronic syntactic dictionary and a semantic dictionary to analyze the relations of V-N phrases. However, the approach suffers from two problems: (1) low word coverage of the semantic dictionary and (2) the semantic type classifier is inadequate. Finally, Chen proposed an automatic VN combination method with features of verbs, nouns, context, and the syllables of words. The experiment results show that the method performs reasonably well without using any other resources.

Based on the above feature selection methods, we extract relevant knowledge from Treebank to design a Vt-N classifier. However we have to resolve the common problem of data sparseness. Learning knowledge by analyzing large-scale unlabeled data is necessary and proved useful in previous works (Wu, 2003; Chen et al., 2008; Yu et al., 2008). Wu developed a machine learning method that acquires verb-object and modifier-head relations automatically. The mutual information scores are then used to prune verb-noun whose scores are below a certain threshold. The author found that accurate identification of the verb-noun relation improved the parsing performance by 4%. Yu et al. learned head-modifier pairs from parsed data and proposed a head-modifier classifier to filter the data. The filtering model uses the following features: a PoS-tag pair of the head and the modifier; the distance between the head and the modifier; and the presence or absence of punctuation marks (e.g., commas, colons, and semi-colons) between the head and the modifier. Although the method improves the parsing performance by 2%, the filtering model obtains limited data; the recall rate is only 46.35%. The authors also fail to solve the problem of Vt-N ambiguity.

Our review of previous works and the observations in Section 1.1 show that lexical words, semantic information, the syllabic length of words, neighboring PoSs and the knowledge learned from large-scale data are important for Vt-N disambiguation. We consider more features for disambiguating Vt-N structures than previous studies. For example, we utilize (1) four relation classification in a real environment, including 'X/H', 'H/X', 'X/X' and 'H/H' relations; (2) unknown word processing of Vt-N words (including semantic type predication and morph-structure predication); (3) unsupervised data selection (a simple and effective way to extend knowledge); and (4) supervised knowledge correction, which makes the extracted knowledge more useful.

3 Design of the Disambiguation Model

The disambiguation model is a Vt-N relation classifier that classifies Vt-N relations into 'H/X' (predicate-object relations), 'X/H' (modifier-head relations), 'H/H' (conjunctive head-head relations), or 'X/X' (independent relations). We use the Maximum Entropy toolkit (Zhang, 2004) to construct the classifier. The advantage of using the Maximum Entropy model is twofold: (1) it has the flexibility to adjust features; and (2) it provides the probability values of the classification, which can be easily integrated into our PCFG parsing model.

In the following sections, we discuss the design of our model for feature selection and extraction, unknown word processing, and world knowledge learning.

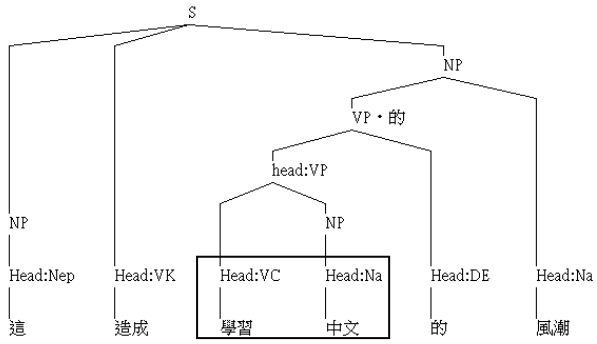
3.1 Feature Selection and Extraction

We divide the selected features into five groups: PoS tags of Vt and N, PoS tags of the context, words, semantics, and additional information. Table 1 shows the feature types and symbol notations. We use symbols of t_1 and t_2 to denote the PoS of Vt and N respectively. The context feature is neighboring PoSs of Vt and N: the symbols of t_{-2} and t_{-1} represent its left PoSs, and the symbol t_3 and t_4 represent its right PoSs. The semantic feature is the lexicon's semantic type extracted from E-HowNet sense expressions (Huang et al., 2008). For example, the E-HowNet expression of "車輛/*vehicles*" is {LandVehicle|車:quantity={mass|眾}}, so its semantic type is {LandVehicle|車}. We discuss the model's performance with different feature combinations in Section 4.

Feature	Feature Description
PoS	PoS of Vt and N t_1; t_2
Context	Neighboring PoSs t_{-2}; t_{-1}; t_3; t_4
Word	Lexical word w_1; w_2
Semantic	Semantic type of word st_1; st_2
Additional Information	Morphological structure of verb *Vmorph*
	Syllabic length of noun *Nlen*

Table 1. The features used in the Vt-N classifier

The example in Figure 1 illustrates feature labeling of a Vt-N structure. First, an instance of a Vt-N structure is identified from Treebank. Then, we assign the semantic type of each word without considering the problem of sense ambiguity for the moment. This is because sense ambiguities are partially resolved by PoS tagging, and the general problem of sense disambiguation is beyond the scope of this paper. Furthermore, Zhao and Huang (1999) demonstrated that the retained ambiguity does not have an adverse impact on identification. Therefore, we keep the ambiguous semantic type for future processing.



zhe zaochen xuexi zhongwen DE fongchao
this cause learn Chinese trend
"This causes the trend of learning Chinese."

Figure 1. An example of a tree with a Vt-N structure

Table 2 shows the labeled features for "學習/*learn* 中文/*Chinese*" in Figure 1. The column *x* and *y* describe relevant features in "學習/*learn*" and "中文/*Chinese*" respectively. Some features are not explicitly annotated in the Treebank, e.g., the semantic types of words and the morphological structure of verbs. We propose labeling methods for them in the next sub-section.

Feature Type	x	y
Word	w_1=學習	w_2=中文
PoS	t_1=VC	t_2=Na
Semantic	st_1=study|學習	st_2=language|語言
Context	t_{-2}=Nep; t_{-1}=VK; t_3=DE; t_4=Na	
Additional Information	Vmorph=VV	Nlen=2
Relation Type	rt = H/X	

Table 2. The feature labels of Vt-N pair in Figure 1

3.2 Unknown Word Processing

In Chinese documents, 3% to 7% of the words are usually unknown (Sproat and Emerson, 2003). By 'unknown words', we mean words not listed in the dictionary. More specifically, in this paper, unknown words means words without semantic type information (i.e., E-HowNet expressions) and verbs without morphological structure information. Therefore, we propose a method for predicting the semantic types of unknown words, and use an affix database to train a morph-structure classifier to derive the morphological structure of verbs.

Morph-Structure Predication of Verbs: We use data analyzed by Chiu et al. (2004) to develop a classifier for predicating the morphological structure of verbs. There are four types of morphological structures for verbs: the coordinating structure (VV), the modifier-head structure (AV), the verb-complement structure (VR), and the verb-object structure (VO). To classify verbs automatically, we incorporate three features in the proposed classifier, namely, the lexeme itself, the prefix and the suffix, and the semantic types of the prefix and the suffix. Then, we use training data from the affix database to train the classifier. Table 3 shows an example of the unknown verb "傳播到/disseminate" and the morph-structure classifier shows that it is a 'VR' type.

Feature	Feature Description
Word=傳播到	Lexicon
PW=傳播	Prefix word
PWST={disseminate|傳播}	Semantic Type of Prefix Word 傳播
SW=到	Suffix Word
SWST={Vachieve|達成}	Semantic Type of Suffix Word 到

Table 3. An example of an unknown verb and feature templates for morph-structure predication

Semantic Type Provider: The system exploits *WORD*, *PoS*, *affix* and *E-HowNet* information to obtain the semantic types of words (see Figure 2). If a word is known and its PoS is given, we can usually find its semantic type by searching the E-HowNet database. For an unknown word, the semantic type of its head morpheme is its semantic type; and the semantic type of the head morpheme is obtained from E-HowNet[1]. For example, the unknown word "傳播到/*disseminate*", its prefix word is "傳播/*disseminate*" and we learn that its semantic type is {disseminate|傳播} from E-HowNet. Therefore, we assign {disseminate|傳播} as the semantic type of "傳播到/*disseminate*". If the word or head morpheme does not exist in the affix database, we assign a general semantic type based on its PoS, e.g., nouns are {thing|萬物} and verbs are {act|行動}. In this matching procedure, we may encounter multiple matching data of words and affixes. Our strategy is to keep the ambiguous semantic type for future processing.

Input: *WORD, PoS*
Output: *Semantic Type (ST)*

```
procedure STP(WORD, PoS)
    (* Initial Step *)
    ST := null;
    (* Step 1: Known word *)
    if WORD already in E-HowNet then
        ST := EHowNet(WORD, PoS);
    else if WORD in Affix database then
        ST := EHowNet(affix of WORD, PoS);
    (* Step 2 : Unknown word *)
    if ST is null and PoS is 'Vt' then
        ST := EHowNet(prefix of WORD, PoS);
    else if ST is null and PoS is 'N' then
        ST := EHowNet(suffix of WORD, PoS);
    (* Step 3 : default *)
    if ST is null and PoS is 'Vt' then
        ST := 'act|行動';
    else if ST is null and PoS is 'N' then
        ST := 'thing|萬物'
    (* Finally *)
    STP := ST;
end;
```

Figure 2. The Pseudo-code of the Semantic Type Predication Algorithm.

[1] The E-HowNet function in Figure 2 will return a null ST value where words do not exist in E-HowNet or Affix database.

3.3 Learning World Knowledge

Based on the features discussed in the previous sub-section, we extract prior knowledge from Treebank to design the Vt-N classifier. However, the training suffers from the data sparseness problem. Furthermore most ambiguous Vt-N relations are resolved by common sense knowledge that makes it even harder to construct a well-trained system. An alternative way to extend world knowledge is to learn from large-scale unlabeled data (Wu, 2003; Chen et al., 2008; Yu et al., 2008). However, the unsupervised approach accumulates errors caused by automatic annotation processes, such as word segmentation, PoS tagging, syntactic parsing, and semantic role assignment. Therefore, how to extract useful knowledge accurately is an important issue.

To resolve the error accumulation problem, we propose two methods: unsupervised NP selection and supervised error correction. The NP selection method exploits the fact that an intransitive verb followed by a noun can only be interpreted as an NP structure, not a VP structure. It is easy to find such instances with high precision by parsing a large corpus. Based on the selection method, we can extend contextual knowledge about NP(V+N) and extract nouns that take adjectival verbs as modifiers. The error correction method involves a small amount of manual editing in order to make the data more useful and reduce the number of errors in auto-extracted knowledge. The rationale is that, in general, high frequency Vt-N word-bigram is either VP or NP without ambiguity. Therefore, to obtain more accurate training data, we simply classify each high frequency Vt-N word bigram into a unique correct type without checking all of its instances. We provide more detailed information about the method in Section 4.3.

4 Experiments and Results

4.1 Experimental Setting

We classify Vt-N structures into four types of syntactic structures by using the bracketed information (tree structure) and dependency relation (head-modifier) to extract the Vt-N relations from treebank automatically. The resources used in the experiments as follows.

Treebank: The Sinica Treebank contains 61,087 syntactic tree structures with 361,834 words. We extracted 9,017 instances of Vt-N structures from the corpus. Then, we randomly

selected 1,000 of the instances as test data and used the remainder (8,017 instances) as training data. Labeled information of word segmentation and PoS-tagging were retained and utilized in the experiments.

E-HowNet: E-HowNet contains 99,525 lexical semantic definitions that provide information about the semantic type of words. We also implement the semantic type predication algorithm in Figure 2 to generate the semantic types of all Vt and N words, including unknown words.

Affix Data: The database includes 13,287 examples of verbs and 27,267 examples of nouns, each example relates to an affix. The detailed statistics of the verb morph-structure categorization are shown in Table 4. The data is used to train a classifier to predicate the morph-structure of verbs. We found that verbs with a conjunctive structure (VV) are more likely to play adjectival roles than the other three types of verbs. The classifier achieved 87.88% accuracy on 10-fold cross validation of the above 13,287 verbs.

	VV	VR	AV	VO
Prefix	920	2,892	904	662
Suffix	439	7,388	51	31

Table 4. The statistics of verb morph-structure categorization

Large Corpus: We used a Chinese parser to analyze sentence structures automatically. The auto-parsed tree structures are used in Experiment 2 (described in the Sub-section 4.3). We obtained 1,262,420 parsed sentences and derived 237,843 instances of Vt-N structure as our dataset (called as ASBC).

4.2 Experiment 1: Evaluation of the Vt-N Classifier

In this experiment, we used the Maximum Entropy Toolkit (Zhang, 2004) to develop the Vt-N classifier. Based on the features discussed in Section 3.1, we designed five models to evaluate the classifier's performance on different feature combinations.

The features and used in each model are described below. The feature values shown in brackets refer to the example in Figure 1.

- **M1** is the baseline model. It uses PoS-tag pairs as features, such as (t_1=VC, t_2=Na).
- **M2** extends the M1 model by adding context features of (t_{-1}=VK, t_1=VC), (t_2=Na, t_3=DE), (t_{-2}=Nep, t_{-1}=VK, t_1=VC), (t_2=Na, t_3=DE, t_4=Na) and (t_{-1}=VK, t_3=DE).
- M3 extends the M2 model by adding lexicon features of (w_1=學習, t_1=VK, w_2=中文, t_2=Na), (w_1＝學習, w_2=中文), (w_1=學習) and (w_2=中文).
- M4 extends the M3 model by adding semantic features of (st_1=study|學習, t_1=VK , st_2=language|語言, t_2=Na), (st_1=study|學習, t_1=VK) and (st_2=language|語言, t_2=Na).
- M5 extends the M4 model by adding two features: the morph-structure of verbs; and the syllabic length of nouns (Vmorph='VV') and (Nlen=2).

Table 5 shows the results of using different feature combinations in the models. The symbol P1(%) is the 10-fold cross validation accuracy of the training data, and the symbol P2(%) is the accuracy of the test data. By adding contextual features, the accuracy rate of M2 increases from 59.10% to 72.30%. The result shows that contextual information is the most important feature used to disambiguate VP, NP and independent structures. The accuracy of M2 is approximately the same as the result of our PCFG parser because both systems use contextual information. By adding lexical features (M3), the accuracy rate increases from 72.30% to 80.20%. For semantic type features (M4), the accuracy rate increases from 80.20% to 81.90%. The 1.7% increase in the accuracy rate indicates that semantic generalization is useful. Finally, in M5, the accuracy rate increases from 81.90% to 83.00%. The improvement demonstrates the benefits of using the verb morph-structure and noun length features.

Models	Feature for Vt-N	P1(%)	P2(%)
M1	(t_1,t_2)	61.94	59.10
M2	+ (t_{-1},t_1) (t_2,t_3) (t_{-2},t_{-1},t_1) (t_2,t_3,t_4) (t_{-1},t_3)	76.59	72.30
M3	+ (w_1,t_1,w_2,t_2) (w_1,w_2) (w_2) (w_1)	83.55	80.20
M4	+ (st_1,t_1,st_2,t_2) (st_1,t_1) (st_2, t_2)	84.63	81.90
M5	+ (Vmorph) (Nlen)	85.01	83.00

Table 5. The results of using different feature combinations

Next, we consider the influence of unknown words on the Vt-N classifier. The statistics shows that 17% of the words in Treebank lack semantic type information, e.g., 留在/*StayIn*, 填飽/*fill*, 貼出/*posted*, and 綁好/*tied*. The accuracy of the Vt-N classifier declines by 0.7% without semantic type information for unknown words. In other words, lexical semantic information improves the accuracy of the Vt-N classifier. Regarding the problem of unknown morph-structure of words, we observe that over 85% of verbs with more than 2 characters are not found in the affix database. If we exclude unknown words, the accuracy of the Vt-N prediction decreases by 1%. Therefore, morph-structure information has a positive effect on the classifier.

4.3 Experiment 2: Using Knowledge Obtained from Large-scale Unlabeled Data by the Selection and Correction Methods.

In this experiment, we evaluated the two methods discussed in Section 3, i.e., unsupervised NP selection and supervised error correction. We applied the data selection method (i.e., *distance*=1, with an intransitive verb (Vi) followed by an object noun (Na)) to select 46,258 instances from the ASBC corpus and compile a dataset called Treebank+ASBC-Vi-N. Table 6 shows the performance of model 5 (M5) on the training data derived from Treebank and Treebank+ASBC-Vi-N. The results demonstrate that learning more nouns that accept verbal modifiers improves the accuracy.

	Treebank+ ASBC-Vi-N	Treebank
size of training instances	46,258	8,017
M5 - P2(%)	83.90	83.00

Table 6. Experiment results on the test data for various knowledge sources

We had also try to use the auto-parsed results of the Vt-N structures from the ASBC corpus as supplementary training data for train M5. It degrades the model's performance by too much error when using the supplementary training data. To resolve the problem, we utilize the supervised error correction method, which manually correct errors rapidly because high frequency instances (w_1, w_2) rarely have ambiguous classifications in different contexts. So we designed an editing tool to correct errors made by the parser in the classification of high frequency Vt-N word pairs. After the manual correction operation, which takes 40 man-hours, we assign the correct classifications (w_1, t_1, w_2, t_2, rt) for 2,674 Vt-N structure types which contains 10,263 instances to creates the ASBC+Correction dataset. Adding the corrected data to the original training data increases the precision rate to 88.40% and reduces the number of errors by approximately 31.76%, as shown in the Treebank+ASBC+Correction column of Table 7.

	Treebank+ ASBC+Correction	Treebank+ ASBC-Vi-N	Treebank
size of training instances	56,521	46,258	8,017
M5 - P2(%)	88.40	83.90	83.00

Table 7. Experiment results of classifiers with different training data

We also used the precision and recall rates to evaluate the performance of the models on each type of relation. The results are shown in Table 8. Overall, the Treebank+ASBC+Correction method achieves the best performance in terms of the precision rate. The results for Treebank+ASBC-Vi-N show that the unsupervised data selection method can find some knowledge to help identify NP structures. In addition, the proposed models achieve better precision rates than the PCFG parser. The results demonstrate that using our guidelines to design a disambiguation model to resolve the Vt-N problem is successful.

		H/X	X/H	X/X
Treebank	R(%)	91.11	67.90	74.62
	P(%)	84.43	78.57	81.86
Treebank+ ASBC-Vi-N	R(%)	91.00	72.22	71.54
	P(%)	84.57	72.67	85.71
Treebank+ ASBC+Correction	R(%)	98.62	60.49	83.08
	P(%)	86.63	88.29	93.51
PCFG	R(%)	90.54	23.63	80.21
	P(%)	78.24	73.58	75.00

Table 8. Performance comparison of different classification models.

4.4 Experiment 3: Integrating the Vt-N classifier with the PCFG Parser

Identifying Vt-N structures correctly facilitates statistical parsing, machine translation, infor-

mation retrieval, and text classification. In this experiment, we develop a baseline PCFG parser based on feature-based grammar representation by Hsieh et al. (2012) to find the best tree structures (*T*) of a given sentence (*S*). The parser then selects the best tree according to the evaluation score *Score(T,S)* of all possible trees. If there are *n* PCFG rules in the tree *T*, the *Score(T,S)* is the accumulation of the logarithmic probabilities of the *i*-th grammar rule (*RPi*). Formula 1 shows the baseline PCFG parser.

$$Score(T,S) = \sum_{i=1}^{n} (RP_i) \quad (1)$$

The Vt-N models can be easily integrated into the PCFG parser. Formula 2 represents the integrated structural evaluation model. We combine *RPi* and *VtNPi* with the weights w_1 and w_2 respectively, and set the value of w_2 higher than that of w_1. *VtNPi* is the probability produced by the Vt-N classifier for the type of the relation between Vt-N bigram determined by the PCFG parsing. The classifier is triggered when a [Vt, N] structure is encountered; otherwise, the Vt-N model is not processed.

$$Score(T,S) = \sum_{i=1}^{n} (w_1 \times RP_i + w_2 \times VtNP_i) \quad (2)$$

The results of evaluating the parsing model incorporated with the Vt-N classifier (see Formula 2) are shown in Table 9 and Table 10. The P2 is the accuracy of Vt-N classification on the test data. The bracketed *f*-score (BF^2) is the parsing performance metric. Based on these results, the integrated model outperforms the PCFG parser in terms of Vt-N classification. Because the Vt-N classifier only considers sentences that contain Vt-N structures, it does not affect the parsing accuracies of other sentences.

	PCFG + M5 (Treebank)	PCFG
P2(%)	80.68	77.09
BF(%)	83.64	82.80

Table 9. The performance of the PCFG parser with and without model M5 from Treebank.

[2] The evaluation formula is (BP*BR*2) / (BP+BR), where BP is the precision and BR is the recall.

	PCFG + M5 (Treebank+ASBC+Correction)	PCFG
P2(%)	87.88	77.09
BF(%)	84.68	82.80

Table 10. The performance of the PCFG parser with and without model M5 from Treebank+ASBC+Correction data set.

4.5 Experiment 4: Comparison of Various Chinese Parsers

In this experiment, we give some comparison results in various parser: 'PCFG Parser' (baseline), 'CDM Parser' (Hsieh et al., 2012), and 'Berkeley Parser' (Petrov et al., 2006). The CDM parser achieves the best score in Traditional Chinese Parsing task of SIGHAN Bake-offs 2012 (Tseng et al., 2012). Petrov's parser (as Berkeley, version is 2009 1.1) is the best PCFG parser for non-English language and it is an open source. In our comparison, we use the same training data for training models and parse the same test dataset based on the gold standard word segmentation and PoS tags. We have already discussed the PCFG parser in Section 4.4. As for CDM parser, we retrain relevant model in our experiments. And since Berkeley parser take different tree structure (Penn Treebank format), we transform the experimental data to Berkeley CoNLL format and re-train a new model with parameters "-treebank CHINESE -SMcycles 4" [3] from training data. Moreover we use "-useGoldPOS" parameters to parse test data and further transform them to Sinica Treebank style from the Berkeley parser's results. The different tree structure formats of Sinica Treebank and Penn Treebank are as follow:

```
Sinica Treebank:
S(NP(Head:Nh:他們)|Head:VC:散播
|NP(Head:Na:熱情))

Penn Treebank:
( (S (NP (Head:Nh (Nh 他們))) (Head:VC
(VC 散播)) (NP (Head:Na (Na 熱情)))))
```

The evaluation results on the testing data, i.e. in P2 metric, are as follows. The accuracy of PCFG parser is 77.09%; CDM parser reaches 78.45% of accuracy; and Berkeley parser is 70.68%. The results show that the problem of Vt-

[3] The "-treebank CHINESE -SMcycles 4" is the best training parameter in Traditional Chinese Parsing task of SIGHAN Bake-offs 2012.

N cannot be well solved by any general parser including CDM parser and Berkeley's parser. It is necessary to have a different approach aside from the general model. So we set the target for a better model for Vt-N classification which can be easily integrated into the existing parsing model. So far our best model achieved the P2 accuracy of 87.88%.

5 Concluding Remarks

We have proposed a classifier to resolve the ambiguity of Vt-N structures. The design of the classifier is based on three important guidelines, namely, adopting linguistically motivated features, using all available resources, and easy integration into parsing model. After analyzing the Vt-N structures, we identify linguistically motivated features, such as lexical words, semantic knowledge, the morphological structure of verbs, neighboring parts-of-speech, and the syllabic length of words. Then, we design a classifier to verify the usefulness of each feature. We also resolve the technical problems that affect the prediction of the semantic types and morph-structures of unknown words. In addition, we propose a framework for unsupervised data selection and supervised error correction for learning more useful knowledge. Our experiment results show that the proposed Vt-N classifier significantly outperforms the PCFG Chinese parser in terms of Vt-N structure identification. Moreover, integrating the Vt-N classifier with a parsing model improves the overall parsing performance without side effects.

In our future research, we will exploit the proposed framework to resolve other parsing difficulties in Chinese, e.g., N-N combination. We will also extend the Semantic Type Predication Algorithm (Figure 2) to deal with all Chinese words. Finally, for real world knowledge learning, we will continue to learn more useful knowledge by auto-parsing to improve the parsing performance.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work was supported by National Science Council under Grant NSC99-2221-E-001-014-MY3.

Reference

Li-li Chang, Keh-Jiann Chen, and Chu-Ren Huang. 2000. Alternation Across Semantic Fields: A Study on Mandarin Verbs of Emotion. *Internal Journal of Computational Linguistics and Chinese Language Processing (IJCLCLP)*, 5(1):61-80.

Keh-Jiann Chen, Chu-Ren Huang, Chi-Ching Luo, Feng-Yi Chen, Ming-Chung Chang, Chao-Jan Chen, , and Zhao-Ming Gao. 2003. Sinica Treebank: Design Criteria, Representational Issues and Implementation. In *(Abeille 2003) Treebanks: Building and Using Parsed Corpora*, pages 231-248. Dordrecht, the Netherlands: Kluwer.

Li-jiang Chen. 2008. Autolabeling of VN Combination Based on Multi-classifier. *Journal of Computer Engineering*, 34(5):79-81.

Wenliang Chen, Daisuke Kawahara, Kiyotaka Uchimoto, Yujjie Zhang, and Hitoshi Isahara. 2008. Dependency Parsig with Short Dependency Relations in Unlabeled Data. In *Proceedings of the third International Joint Conference on Natural Language Processing (IJCNLP)*. pages 88-94..

Chih-ming Chiu, Ji-Chin Lo, and Keh-Jiann Chen. 2004. Compositional Semantics of Mandarin Affix Verbs. In *Proceedings of the Research on Computational Linguistics Conference (ROCLING)*, pages 131-139.

Yu-Ming Hsieh, Ming-Hong Bai, Jason S. Chang, and Keh-Jiann Chen. 2012. Improving PCFG Chinese Parsing with Context-Dependent Probability Re-estimation, In *Proceedings of the Second CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 216–221.

Shu-Ling Huang, You-Shan Chung, Keh-Jiann Chen. 2008. E-HowNet: the Expansion of HowNet. In *Proceedings of the First National HowNet workshop*, pages 10-22, Beijing, China.

Chunhi Liu, *Xiandai Hanyu Shuxing Fanchou Yianjiu (現代漢語屬性範疇研究)*. Chengdu: Bashu Books, 2008.

Jiaju Mei, Yiming Lan, Yunqi Gao, and Yongxian Ying. 1983. *A Dictionary of Synonyms*. Shanghai Cishu Chubanshe.

Slav Petrov, Leon Barrett, Romain Thibaux and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceesings of COLING/ACL*, pages 433-400.

Likun Qiu. 2005. Constitutive Relation Analysis for V-N Phrases. *Journal of Chinese Language and Computing*, 15(3):173-183.

Richard Sproat and Thomas Emerson, 2003. The first International Chinese Word Segmentation Bakeoff. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pages 133-143.

Honglin Sun and Dan Jurafsky. 2003. The Effect of Rhythm on Structural Disambiguation in Chinese. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pages 39-46.

Yuen-Hsieh Tseng, Lung-Hao Lee, and Liang-Chih Yu. 2012. Tranditional Chinese Parsing Evaluation at SIGHAN Bake-offs 2012. In *Proceedings of the Second CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 199-205.

Andi Wu. 2003. Learning Verb-Noun Relations to Improve Parsing. In *Proceedings of the Second SIGHAN workshop on Chinese Language Processing*, pages 119-124.

Kun Yu, Daisuke Kawahara, and Sadao Kurohashi. 2008. Chinese Dependency Parsing with Large Scale Automatically Constructed Case Structures, In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING2008)*, pages 1049-1056.

Jun Zhao and Chang-ning Huang. 1999. The Complex-feature-based Model for Acquisition of VN-construction Structure Templates. *Journal of Software*, 10(1):92-99.

Le Zhang. 2004. Maximum Entropy Modeling Toolkit for Python and C++. Reference Manual.

Neural Networks Leverage Corpus-wide Information for Part-of-speech Tagging

Yuta Tsuboi
IBM Research - Tokyo
yutat@jp.ibm.com

Abstract

We propose a neural network approach to benefit from the non-linearity of corpus-wide statistics for part-of-speech (POS) tagging. We investigated several types of corpus-wide information for the words, such as word embeddings and POS tag distributions. Since these statistics are encoded as dense continuous features, it is not trivial to combine these features comparing with sparse discrete features. Our tagger is designed as a combination of a linear model for discrete features and a feed-forward neural network that captures the non-linear interactions among the continuous features. By using several recent advances in the activation functions for neural networks, the proposed method marks new state-of-the-art accuracies for English POS tagging tasks.

1 Introduction

Almost all of the approaches to NLP tasks such as part-of-speech tagging and syntactic parsing mainly use sparse discrete features to represent local information such as word surfaces in a size-limited window. The non-linearity of those discrete features is often used in many NLP tasks since the simple conjunction (AND) of discrete features represents the co-occurrence of the features and is intuitively understandable. In addition, the thresholding of these combinatorial features by simple counts effectively suppresses the combinatorial increase of the parameters. At the same time, although global information had also been used in several reports (Nakagawa and Matsumoto, 2006; Huang and Yates, 2009; Turian et al., 2010; Schnabel and Schütze, 2014), the non-linear interactions of these features were not well investigated since these features are often dense continuous features and the explicit non-linear expansions are counterintuitive and drastically increase the number of the model parameters. In our work, we investigate neural networks used to represent the non-linearity of global information for POS tagging in a compact way.

We focus on four kinds of corpus-wide information: (1) *word embeddings*, (2) POS tag distributions, (3) *supertag* distributions, and (4) context word distributions. All of them are continuous dense features and we use a feed-forward neural network to exploit the non-linearity of these features. Although all of them except (3) have been used for POS tagging in previous work (Nakamura et al., 1990; Schmid, 1994; Schnabel and Schütze, 2014; Huang and Yates, 2009), we propose a neural network approach to capture the non-linear interactions of these features. By feeding these features into neural networks as an input vector, we can expect our tagger can handle not only the non-linearity of the N-grams of the same kinds of features but also the non-linear interactions among the different kind of features.

Our tagger combines a linear model using sparse high-dimensional features and a neural network using continuous dense features. Although Collobert et al. (2011) seeks to solve NLP tasks without depending on the feature engineering of conventional NLP methods, our architecture is more practical because it integrates the neural networks into a well-tuned conventional method. Thus, our tagger enjoys both the manually explored combinations of discrete features and the automatically learned non-linearity of the continuous features. We also studied some of the newer activation functions: Rectified Linear Units (Nair and Hinton, 2010), Maxout networks (Goodfellow et al., 2013), and L_p-pooling (Gulcehre et al., 2014; Zhang et al., 2014).

Deep neural networks have been a hot topic in many application areas such as computer vi-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 938–950,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

sion and voice recognition. However, although neural networks show state-of-the-art results on a few semantic tasks (Zhila et al., 2013; Socher et al., 2013; Socher et al., 2011), neural network approaches have not performed better than the state-of-the-art systems for traditional syntactic tasks. Our neural tagger shows state-of-the-art results: 97.51% accuracy in the standard benchmark on the Penn Treebank (Marcus et al., 1993) and 98.02% accuracy in POS tagging on CoNLL2009 (Hajič et al., 2009). In our experiments, we found that the selection of the activation functions led to large differences in the tagging accuracies. We also observed that the POS tags of the words are effectively clustered by the hidden activations of the intermediate layer. This observation is evidence that the neural network can find good representations for POS tagging.

The remainder of this paper is organized as follows. Section 2 introduces our deterministic tagger and its learning algorithm. Section 3 describes the continuous features that represent corpus-wide information and Section 4 is about the neural network we used. Section 5 presents our empirical study of the effects of corpus-wide information and neural networks on English POS tagging tasks. Section 6 describes related work, and Section 7 concludes and suggests items for future work.

2 Transition-based tagging

Our tagging model is a deterministic tagger based on Choi and Palmer (2012), which is a one-pass, left-to-right tagging algorithm that uses well-tuned binary features.

Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_T) \in X^T$ be an input token sequence of length T and $\boldsymbol{y} = (y_1, y_2, \ldots, y_T) \in Y^T$ be a corresponding POS tag sequence of $\boldsymbol{x}$. We denote the predicted tags by a tagger as $\hat{\boldsymbol{y}}$ and the subsequence from r to t as $\boldsymbol{y}_r^t$. The prediction of the t-th tag is deterministically done by the classifier:

$$\hat{y}_t = \underset{y \in Y}{\operatorname{argmax}} f_{\boldsymbol{\theta}}(\boldsymbol{z}_t, y), \tag{1}$$

where $f_{\boldsymbol{\theta}}$ is a scoring function with arbitrary parameters, $\boldsymbol{\theta} \in \mathbb{R}^d$, that are to be learned and $\boldsymbol{z}_t$ is an arbitrary feature representation of the t-th position using $\boldsymbol{x}$ and $\hat{\boldsymbol{y}}_1^{t-1}$ which is the prediction history of the previous tokens.

We extend Choi and Palmer (2012) in three ways: (1) an online SVM learning algorithm with L_1 and L_2 regularization, (2) continuous features for corpus-wide information, and (3) the composite function of a linear model for discrete features and a non-linear model for continuous features. Since (2) and (3) are the main topics of this paper, they are explained in detail in Sections 3 and 4 and we describe only (1) here.

First, our learning algorithm trains a multi-class SVM with L_1 and L_2 regularization based on *Follow the Proximally Regularized Leader* (FTRL-Proximal) (McMahan, 2011). In the k-th iteration, the parameter update is done by

$$\boldsymbol{\theta}^k = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{l=1}^{k} \left(\boldsymbol{g}^l \cdot \boldsymbol{\theta} + \frac{1}{2\boldsymbol{\eta}^l} \left|\left| \boldsymbol{\theta} - \boldsymbol{\theta}^l \right|\right|_2^2 \right) + R(\boldsymbol{\theta}),$$

where $\boldsymbol{g}^k \in \mathbb{R}^d$ is a subgradient of the hinge loss function and $R(\boldsymbol{\theta}) = \boldsymbol{\lambda}_1 ||\boldsymbol{\theta}||_1 + \frac{\boldsymbol{\lambda}_2}{2} ||\boldsymbol{\theta}||_2^2$ is the composite function of the L_1 and L_2 regularization terms with hyper-parameters $\boldsymbol{\lambda}_1 \geq 0$ and $\boldsymbol{\lambda}_2 \geq 0$. To incorporate an adaptive learning rate scheduling, Adagrad (Duchi et al., 2010), we use per-coordinate learning rates for $\{i | 1 \leq i < d\}$:

$$\eta_i^k = \frac{\alpha_i}{\left(\beta_i + \sqrt{\sum_{l=1}^{k} (g_i^l)^2} \right)},$$

where $\alpha \geq 0$ and $\beta \geq 0$. Although the naive implementation may require $O(k)$ computation in the k-th iteration, FTRL-Proximal can be implemented efficiently by maintaining two length-d vectors, $\boldsymbol{m} = \sum_l^k \boldsymbol{g}^l - \frac{1}{2\boldsymbol{\eta}^l}\boldsymbol{\theta}^l$ and $\boldsymbol{n} = \sum_l^k (g_i^l)^2$ (McMahan et al., 2013).

Second, to overcome the error propagation problem, we train the classifier with a simple variant of the on-the-fly example generation algorithm from Goldberg and Nivre (2012). Since the scoring function refers to the prediction history, Choi and Palmer (2012) uses the gold POS tags, $\boldsymbol{y}_1^{t-1}$, to generate training examples, which means they assume all of the past decisions are correct. However, this causes error propagation problems, since each state depends on the history of the past decisions. Therefore, at the k-th iteration and the t-th position of the input sequence, we simply use the predictions of the previously learned classifiers to generate training examples, i.e.,

$$\hat{y}_{t-r} = \underset{y \in Y}{\operatorname{argmax}} f_{\boldsymbol{\theta}_{k-r}}(\boldsymbol{z}_{t-r}, y)$$

for all $\{r | 1 \leq r < t-1\}$. Although it is not theoretically justified, it empirically runs as a

stochastic version of DAGGER (Ross et al., 2011) or SEARN (Daumé III et al., 2009) with the speed benefit of online learning.

Algorithm 1 Learning algorithm

function LEARN($\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{m}, \boldsymbol{n}, \boldsymbol{\theta}^k$)
  while $\neg$ stop **do**
    Select a random sentence $(\boldsymbol{x}, \boldsymbol{y})$
    for $t = 1$ to T **do**
      $\boldsymbol{u}$=UPDATE($\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{m}, \boldsymbol{n}, \boldsymbol{\theta}^k$)
      $\hat{y}_t = \mathrm{argmax}_{y \in Y} f_{\boldsymbol{u}}(\boldsymbol{z}_t, y)$
      $\tilde{y} = \mathrm{argmax}_{y \neq y_t} f_{\boldsymbol{u}}(\boldsymbol{z}_t, , y)$
      if $f_{\boldsymbol{u}}(\boldsymbol{z}_t, y_t) - f_{\boldsymbol{u}}(\boldsymbol{z}_t, \tilde{y}) < 1$ **then**
        $\boldsymbol{g} = \partial_{\boldsymbol{u}} \ell(\boldsymbol{z}_t, y_t, \tilde{y})$ ▷ Subgradient
        For all $i \in I$ compute
        $\sigma_i = \left(\sqrt{n_i + g_i^2} - \sqrt{n_i}\right) / \alpha_i$
        $m_i \leftarrow m_i + g_i - \sigma_i u_i$
        $n_i \leftarrow n_i + g_i^2$
      end if
      $k \leftarrow k + 1$
    end for
  end while
  return $\boldsymbol{\theta}^k$
end function

function UPDATE($\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{m}, \boldsymbol{n}, \boldsymbol{\theta}^k$)
  for $i \in I$ **do**
    $\theta_i^k = \begin{cases} 0 & \text{if } |m_i| \leq \lambda_1 \\ \frac{-m_i + \mathrm{sgn}(m_i)\lambda_1}{(\beta_i \lambda_2 + \sqrt{n_i})/\alpha_i + \lambda_2} & \text{otherwise} \end{cases}$
    $u_i \leftarrow \theta_i^k$
    if acceleration **then**
      $u_i \leftarrow \theta_i^k + \frac{k}{k+3}\left(\theta_i^k - \theta_i^{k-1}\right)$
    end if
  end for
  for $i \notin I$ **do**
    $u_i \leftarrow \theta_i^k \leftarrow \theta_i^{k-1}$
    ▷ Leaving all θ for inactive i unchanged
  end for
  return $\boldsymbol{u}$
end function

Algorithm 1 summarizes our training process where $\ell(\boldsymbol{z}_t, y_t, \tilde{y}) := \max(0, 1 - f_{\boldsymbol{\theta}}(\boldsymbol{z}_t, y) + f_{\boldsymbol{\theta}}(\boldsymbol{z}_t, \tilde{y}))$ is the multi-class hinge loss (Crammer and Singer, 2001). I in Algorithm 1 is a set of parameter indexes that correspond to the non-zero features, so the update is sparse for sparse features. In addition, for the parameter update of the neural networks, we also use an accelerated proximal method (Parikh and Boyd, 2013), which is considered as a variant of the momentum methods (Sutskever et al., 2013). Although $\boldsymbol{u}$ and $\boldsymbol{\theta}$ are the same when the acceleration is not used, $\boldsymbol{u}$ in Algorithm 1 is an extrapolation step in the accelerated method. Although we do not focus on the learning algorithm in this work, the algorithm converges quite quickly and the speed is important because the neural network extension described later requires a hyper-parameter search which is computationally demanding.

3 Corpus-wide Information

Since typical discrete features indicate only the occurrence in a local context and do not convey corpus-wide statistics, we studied four kinds of continuous features for POS tagging to represent the corpus-wide information.

3.1 Word embeddings

Word embeddings, or distributed word representations, embed the words into a low-dimensional continuous space. Most of the neural network applications for NLP use word embeddings (Collobert et al., 2011; Socher et al., 2011; Zhila et al., 2013; Socher et al., 2013), and even for linear models, Turian et al. (2010) highlights the benefit of word embeddings on sequential labeling tasks.

In particular, in our experiments, we used two recently proposed algorithms, *word2vec* (Mikolov et al., 2013) and *glove* (Pennington et al., 2014), which are simple and scalable, although our method could use other word embeddings. Word2vec trains the word embeddings to predict the words surrounding each word, and glove trains the word embeddings to predict the logarithmic count of the surrounding words of each word. Thus, these embeddings can be seen as the distributed versions of the distributional features since the word vectors compactly represent the distribution of the context in which a word appears. We normalized the word embeddings to unit length and used the average vector of training vocabulary for the unknown tokens.

3.2 POS tag distribution

In a way similar to Schmid (1994), we use POS tag distribution over a training corpus. Each word is represented by a vector of length $|Y|$ in which the y-th element is the conditional probabilities with which that word gets the y-th POS tag. We also use the POS tag distributions of the affixes and

spelling binary features used in Choi and Palmer (2012). We cite the definitions of these features.

1. Affix: $c_{:1}, c_{:2}, c_{:3}, c_{n:}, c_{n-1:}, c_{n-2:}, c_{n-3:}$ where c_* is a character string in a word. For example $c_{:2}$ is the prefix of length two of a word and $c_{n-1:}$ is the suffix of length two of a word.

2. Spelling: initial uppercase, all uppercase, all lowercase, contains 1/2+ capital(s) not at the beginning, contains a (period/number/hyphen).

The probabilities for a feature b is estimated with additive smoothing as

$$P(y|b) = \frac{C(b,y)+1}{C(b)+|Y|}, \quad (2)$$

where $C(b)$ and $C(b,y)$ are the counts of b and co-occurrences of b and y, respectively. In addition, an extra dimension for sentence boundaries is added to the vector for word-forms. In total, the POS tag distributions for each word are encoded by a vector of dimension $|Y|+1+|Y|\times 14$ ($|Y|$ for lowercase simplified word-forms, 1 for sentence boundaries, $|Y| \times 7$ for affixes, and $|Y| \times 7$ for spellings).

3.3 Supertag distribution

We also use the distribution of *supertags* for dependency parsing. Supertags are lexical templates which are extracted from the syntactic dependency structures and suppertagging is often used for the pre-processing of a parsing task. Since the supertags encode rich syntactic information, we expect the supertag distribution of a word to also provide clues for the POS tagging. We used two types of supertags: One is the dependency relation label of the head of the word and the other is that of the dependents of the word. Following Ouchi et al. (2014), we added the relative position, left (L) or right (R), to the supertags. For example, a word has its dependents in the left direction with a label "nn" and in the right direction with a label "amod", so its supertag set for dependents is {"nn/L", "amod/R"}. A special supertag "NO-CHILD" is used for a word that has no dependent. Note that, although the Model 2 supertag set of Ouchi et al. (2014) is defined as the combination of head and dependent tags, we used them separately. The feature values for each word are defined in the same way as Equation 2 in Section 3.2. Since a word can have more than one dependent, the dependent supertag features are no longer multinomial distributions but we used them in that way. Note that, since the feature values are calculated using the tree annotations from training set, our tagger does not require any dependency parser at runtime.

3.4 Context word distribution

This is the simplest distributional features in which each word is represented by the distributions of its left and right neighbors. Although the context word distribution is similar to word embeddings, we believe they complement each other, as reported by Levy and Goldberg (2014). Following Schnabel and Schütze (2014), we restricted the set of indicator words to the 500 most frequent words in the corpus, and used two special feature entries: One is the marginal probability of the non-indicator words and the other is the probabilities of neighboring sentence boundaries. The conditional probabilities for left and right neighbors are estimated in the same way as Equation 2 in Section 3.2, and there are a total of $1,004$ dimensions of this feature for a word.

4 Neural Networks

The non-linearity of the discrete features has been exploited in many NLP tasks, since the simple conjunction of the discrete features is intuitive and the thresholding of these combinatorial features by their feature counts effectively suppresses the combinatorial increase of the parameters.

In contrast, it is not easy to manually tune the non-linearity of the continuous features. For example, it is not intuitive to design the conjunction features of two kinds of word embeddings, word2vec and glove. Although kernel methods have been used to incorporate non-linearity in prior research, they are rarely used now because their tagging speed is too slow (Giménez and Màrquez, 2003). Our solution is to introduce feed-forward neural networks to capture the non-linearity of the corpus-wide information.

4.1 Hybrid model

We designed our tagger as a hybrid of a linear model and a non-linear model. Wang and Manning (2013) reported that a neural network using both sparse discrete features and dense (low-

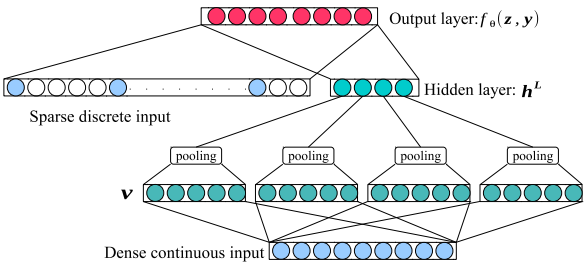


Figure 1: A hybrid architecture of a linear model and a neural network with a pooling activation function

dimensional) continuous features was worse than a linear model using the same two features. At the same time, they also reported that a neural network using only the dense continuous features outperformed a linear model using the same features. Based on their results, we applied neural networks only for the continuous features and used a linear model for the discrete features.

Formally, the scoring function (1) in Section 2 is defined as the composite function of two terms: $f(\boldsymbol{z}, y) := f_{\text{linear}}(\boldsymbol{z}, y) + f_{\text{nn}}(\boldsymbol{z}, y)$. The first f_{linear} is the linear model and the second f_{nn} is a neural network. Since this is a linear combination of two functions, the subgradient of the loss function required for Algorithm 1 is also the linear combination of subgradients of two functions, which means

$$\begin{aligned}\partial_{\boldsymbol{\theta}} \ell(\boldsymbol{z}_t, y_t, \tilde{y}) = \partial_{\boldsymbol{\theta}} f_{\text{linear}}(\boldsymbol{z}_t, \tilde{y}) + \partial_{\boldsymbol{\theta}} f_{\text{nn}}(\boldsymbol{z}_t, \tilde{y}) \\ - \partial_{\boldsymbol{\theta}} f_{\text{linear}}(\boldsymbol{z}_t, y_t) - \partial_{\boldsymbol{\theta}} f_{\text{nn}}(\boldsymbol{z}_t, y_t)\end{aligned}$$

if $f_{\boldsymbol{\theta}}(\boldsymbol{z}_t, y_t) - f_{\boldsymbol{\theta}}(\boldsymbol{z}_t, \tilde{y}) < 0$.

First, the linear model can be defined as

$$f_{\text{linear}}(\boldsymbol{z}, y) := \boldsymbol{\theta}_{\text{d}} \cdot \boldsymbol{\phi}_{\text{d}}(\boldsymbol{z}, y),$$

where $\boldsymbol{\phi}_{\text{d}}(\boldsymbol{z}, y)$ is a feature mapping for the discrete part of z and a POS tag, and $\boldsymbol{\theta}_{\text{d}}$ is the corresponding parameter vector. Since this is a linear model, the gradient of this function is simply $\partial_{\boldsymbol{\theta}} f_{\text{linear}}(\boldsymbol{z}, y) = \boldsymbol{\phi}_{\text{d}}(\boldsymbol{z}, y)$.

Second, each hidden layer of our neural networks non-linearly transforms an input vector $\boldsymbol{h}'$ into an output vector $\boldsymbol{h}$ and we can say $\boldsymbol{h}'$ is the continuous part of $\boldsymbol{z}$ at the first layer. Let $\boldsymbol{h}^L$ be a hidden activation of the top layer, which is the non-linear transformation of the continuous part of $\boldsymbol{z}$. The output layer of the neural network is defined as

$$f_{\text{nn}}(\boldsymbol{z}, y) := \boldsymbol{\theta}_{\text{o}} \cdot \boldsymbol{\phi}_{\text{o}}(\boldsymbol{h}^L, y),$$

where $\boldsymbol{\phi}_{\text{o}}(\boldsymbol{h}, y)$ is a feature mapping for the hidden variables and a POS tag, and $\boldsymbol{\theta}_{\text{o}}$ is the corresponding parameter vector.

4.2 Activation functions

The hidden variables $\boldsymbol{h}$ are computed by the recursive application of a non-linear activation function. Since new styles of the activation functions were recently proposed, we review several activation functions here. Let $\boldsymbol{v} \in \mathbb{R}^{|V|}$ be the input of an activation function and each element is $v_j = \boldsymbol{\theta}_{\text{nn},j} \cdot \boldsymbol{h}' + \theta_{\text{bias},j}$, where $\boldsymbol{\theta}_{\text{nn},j}$ is the parameter vector for v_j and $\theta_{\text{bias},j}$ is the bias parameter for v_j. We also assume $\boldsymbol{v}$ is divided into groups of size G, and denote the j-th element of the i-th group as $\{v_{ij} | 1 \leq i \leq |V|/G \wedge 1 \leq j \leq G\}$. We studied three activation functions:

1. Rectified linear units (ReLUs) (Nair and Hinton, 2010):

   $$h_j = \max(0, v_j) \text{ for all } \{j | 1 \leq j \leq |V|\}.$$

   Note that a subgradient of ReLUs is

   $$\frac{\partial h_j}{\partial \boldsymbol{\theta}} = \begin{cases} \frac{\partial v_j}{\partial \boldsymbol{\theta}} & \text{if } v_j > 0 \\ 0 & \text{otherwise.} \end{cases}$$

2. Maxout networks (MAXOUT) (Goodfellow et al., 2013):

   $$h_i = \max_{1 \leq j \leq G} v_{ij} \text{ for all } \{i | 1 \leq i \leq \frac{|V|}{G}\}.$$

   Note that a subgradient of MAXOUT is

   $$\frac{\partial h_i}{\partial \boldsymbol{\theta}} = \frac{\partial v_{i\hat{j}}}{\partial \boldsymbol{\theta}}, \text{ where } \hat{j} = \operatorname*{argmax}_{1 \leq j \leq G} v_{ij}$$

3. Normalized L_p-pooling (L_p) (Gulcehre et al., 2014):

   $$h_i = \left(\frac{1}{G} \sum_{j=1}^{G} |v_{ij}|^p \right)^{\frac{1}{p}} \text{ for all } \{i | 1 \leq i \leq \frac{|V|}{G}\}.$$

   Note that a subgradient of L_p is

   $$\frac{\partial h_i}{\partial \boldsymbol{\theta}} = \sum_{j=1}^{G} \frac{\partial v_{ij}}{\partial \boldsymbol{\theta}} \frac{v_{ij} |v_{ij}|^{p-2}}{G} \left(\frac{1}{G} \sum_{j=1}^{G} |v_{i,j}|^p \right)^{\frac{1}{p}-1}.$$

The activation inputs for each predefined group, $\{v_{1j}, \ldots, v_{Gj}\}$, are aggregated by a non-linear function in MAXOUT or L_p activation functions, while each input is transformed into a corresponding hidden variable in the ReLUs. When the number of parameters required for these activation functions is the same, the number of output variables $\boldsymbol{h}$ for MAXOUT and L_p is one-G-th smaller than that for ReLUs. Boureau et al. (2010) show pooling operations theoretically reduce the variance of hidden activations, and our experimental results also show MAXOUT and L_p perform better than the ReLUs with the same number of parameters. Note that MAXOUT is a special case of unnormalized L_p pooling when $p = \infty$ and $v_j > 0$ for all j (Zhang et al., 2014). Figure 1 summarizes the proposed architecture with a single hidden layer and a pooling activation function.

4.3 Hyper-parameter search

Finally, the subgradients of the neural network, $f_{\text{nn}}(\boldsymbol{z}, y)$, can be computed through standard back-propagation algorithms and we can apply them in Algorithm 1. However, many of the hyper-parameters have to be determined for the training of the neural networks, and two stages of random hyper-parameter searches (Bergstra and Bengio, 2012) are used in our experiments. Note that the parameters are grouped into three sets, $\boldsymbol{\theta}_d, \boldsymbol{\theta}_{\text{o}}, \boldsymbol{\theta}_{\text{nn}}$, and the same values for $\lambda_1, \lambda_2, \alpha, \beta$ are used for each parameter set.

In the first stage, we randomly select 32 combinations of λ_2 for f_{nn}, λ_1, λ_2 for f_{linear}, the epoch to start the L1/L2 regularizations, and the on and off the acceleration in Algorithm 1. Here are the candidates of three hyper-parameters:

1. λ_1: 0 for the update of f_{nn} and $\{0, 10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}, 1\}$ for the update of f_{linear};
2. λ_2: $\{0.1, 0.5, 1, 5, 10\}$ for the update of f_{nn} and $\{1, 5, 10, 50, 100\}$ for the update of f_{linear}; and
3. Epoch to start the regularizations: $\{0, 1, 2\}$.

In the second stage with each hyper-parameter combination above, we select 8 random combinations of α, β for both f_{linear} and f_{nn} and initial parameter ranges R for f_{nn}. Here are the candidates of the three hyper-parameters:

1. α: $\{0.01, 0.05, 0.1, 0.5, 1, 5\}$;
2. β: $\{0.5, 1, 5\}$;
3. R: $\{[-0.1, 0.1], [-0.05, 0.05], [-0.01, 0.01], [-0.005, 0.005]\}$.

The values of $\boldsymbol{\theta}$ for f_{nn} are uniformly sampled in the range of the randomly selected R. Note that, according to Goodfellow et al. (2014), the biases $\boldsymbol{\theta}_{\text{bias}}$ are initialized as 0 for MAXOUT and L_p, and uniformly sampled from a range $R + \max(R)$, i.e., always initialized with non-negative values. The best combination for the development set is chosen after training that uses random 20% of the training set at the second stage, and Algorithm 1 is terminated when the all token accuracy of the development data has been declining for 5 epochs at the first stage. In other words, 32×8 random combinations of α, β, and $\boldsymbol{\theta}$ for f_{nn} were tested.

Data Set	#Sent.	#Tokens	#Unknown
Training	38,219	912,344	0
Development	5,527	131,768	4,467
Testing	5,462	129,654	3,649

Table 1: Data set splits for PTB.

5 Experiments

5.1 Setup

Our experiments were mainly performed using the Wall Street Journal from Penn Treebank (PTB) (Marcus et al., 1993). We used tagged sentences from the parse trees (Toutanova et al., 2003) and followed the standard approach of splitting the PTB, using sections 0–18 for training, section 19–21 for development, and section 22–24 for testing (Table 1). In addition, we used the CoNLL2009 data sets with the training, development, and test splits used in the shared task (Hajič et al., 2009) for better comparison with a joint model of POS tagging and dependency parsing (Bohnet and Nivre, 2012).

Our baseline tagger was trained by Algorithm 1. As discrete features for our tagger, we used the same binary feature set as Choi and Palmer (2012) which is composed of (a) $1, 2, 3$-grams of the surface word-forms and their predicted/dominated POS tags, (b) the prefixes and suffixes of the words, and (c) the spelling types of the words. In the same way as Choi and Palmer (2012), we used lowercase simplified word-forms which appeared at least 3 times.

In addition to their binary features, we used continuous features which are the concatenation of the corpus-wide features in a context window. The window of size $w = 2s + 1$ is the local context centered around x_t: $x_{t-s}, \cdots, x_t, \cdots, x_{t+s}$. The experimental settings of each feature described in Section 3 are as follows.

Word embeddings

We used two word vectors: 300-dimensional vectors that were learned by word2vec using a part of the Google News dataset (around 100 billion tokens) [1], and 300-dimensional vectors that were learned by glove using a part of the Common Crawl dataset (840 billion tokens) [2]. For sentence boundaries, we use the vector of the special entry "</s>" for the word2vec embeddings and the zero vector for the glove embeddings.

POS tag distribution

The counts are calculated using training data.

Supertag distribution

In the experiments on PTB, we used the Stanford parser v2.0.4[3] to convert from phrase structures to dependency structures so that the dependency relation labels of the Stanford dependencies are used. The size of the supertag set is 85 for both heads and dependents in our experiments. In the experiments on CoNLL2009, the dependency structures and labels defined in CoNLL2009 are used and the size of supertag set is 99 for both heads and dependents.

Context word distribution

To count the neighboring words in our experiments, we used sections 0–18 of the Wall Street Journal and all of the Brown corpus from Penn Treebank (Marcus et al., 1993).

Since the training of the neural networks is computationally demanding, first, we trained the linear classifiers using Algorithm 1 to select the best window sizes for each corpus-wide information of Section 3. Then the best window size setting for the development set of PTB was used for training the neural networks described in Section 4.

[1]The pre-trained vectors are available at https://code.google.com/p/word2vec

[2]The pre-trained vectors are available at http://nlp.stanford.edu/projects/glove/

[3]http://nlp.stanford.edu/software/lex-parser.shtml

#	Window size w2v	glv	pos	stg	cw	Accuracy (%) All	Unk.
1	-	-	-	-	-	97.15	86.81
2	3	-	-	-	-	97.36	88.96
3	-	3	-	-	-	97.34	89.55
4	3	3	-	-	-	97.40	90.44
5	3	3	3	-	1	97.44	90.17
6	3	3	3	1	1	97.44	**90.53**
7	3	3	3	3	1	**97.45**	90.22
8	3	3	6	-	1	97.41	90.51
9	3	3	6	3	1	97.44	90.15

Table 2: Feature and window size selection: development accuracies of all tokens (All) and unknown tokens (Unk.) of linear models trained on PTB (w2v: word2vec; glv: glove; pos: POS tag distribution; stg: supertag distribution; cw: context word distribution).

We fixed the group size at 8 for MAXOUT and L_p, and the number of hidden variables was chosen from $\{32, 48\}$ for MAXOUT and L_p and from $\{32, 64, 128, 256, 384\}$ for ReLUs according to all token accuracy on the development data of PTB. We report the POS tagging accuracy for both all of the tokens and only for the unknown tokens that do not appear in the training set.

5.2 Results

Table 2 shows the accuracies of the linear models on PTB with different window sizes for the continuous features. The window sizes of the word embeddings (word2vec and glove) in Section 3.1, POS tag distributions in Section 3.2, supertag distributions in Section 3.3, and context word distributions in Section 3.4 are shown in the columns of *w2v*, *glv*, *pos*, *stg*, and *cw*, respectively. Note that "-" denotes the corresponding feature was not used at all and the first row with all "-" denotes the results only using the original binary features from Choi and Palmer (2012). The window sizes in Table 2 are chosen mainly to investigate the effect of the word2vec embeddings, glove embeddings, and supertag distributions, since they had not previously been used for POS tagging.

The additions of the word embeddings improve all token accuracy by about 0.2 points according to the results shown in Nos. $1, 2, 3$. Although both word embeddings improved the accuracy of the unknown tokens, the gain of the glove embeddings (No. 3) is larger than that of the

#	Neural Network Settings			Development Set		Test Set	
	Activation functions	**#Hidden**	**Group size (G)**	**All**	**Unk.**	**All**	**Unk.**
1	Linear model	-	-	97.45	90.22	97.46	91.39
2	ReLUs	384	1	97.45	90.87	97.42	91.04
3	$L_p(p=2)$	48	8	**97.52**	90.91	**97.51**	91.64
4	$L_p(p=3)$	32	8	97.51	90.91	**97.51**	91.53
5	MAXOUT	48	8	97.50	90.89	97.50	**91.67**
6	$L_p(p=2)$ (w/o linear part)	48	8	97.39	**91.18**	97.40	91.23

Table 3: Development and test accuracies of all tokens and unknown tokens (%) on PTB.

Tagger	**All**	**Unk.**
Manning (2011)	97.32	90.79
Søgaard (2011)	97.50	N/A
$L_p(p=2)$	97.51	91.64

(a) Test accuracies on PTB

Tagger	**All**	**Unk.**
Bohnet and Nivre (2012)	97.84	N/A
$L_p(p=2)$	98.02	92.01

(b) Test accuracies on CoNLL2009

Table 4: Test accuracies of all tokens and unknown tokens (%) comparing with the previously reported results

word2vec (No. 2). The reason for this difference in the two embeddings may be because the training data for the glove vectors is 8 times larger than that for the word2vec vectors. The usage of the two word embeddings shows further improvement in the tagging accuracy over single word embeddings (No. 4).

The addition of the POS tag distributions and the context word distributions improves all token accuracy (Nos. $5, 8$). The comparison between the results with stag="-" (Nos. $5, 8$) and stag $= \{1, 3\}$ (Nos. $6, 7, 9$) indicates the minor but consistent improvement by using the supertag distribution features in Section 3.3. Finally, the 7th window-size setting in Table 2 achieves the best all token accuracy among the linear models, so we chose this setting for the experiments with the neural networks.

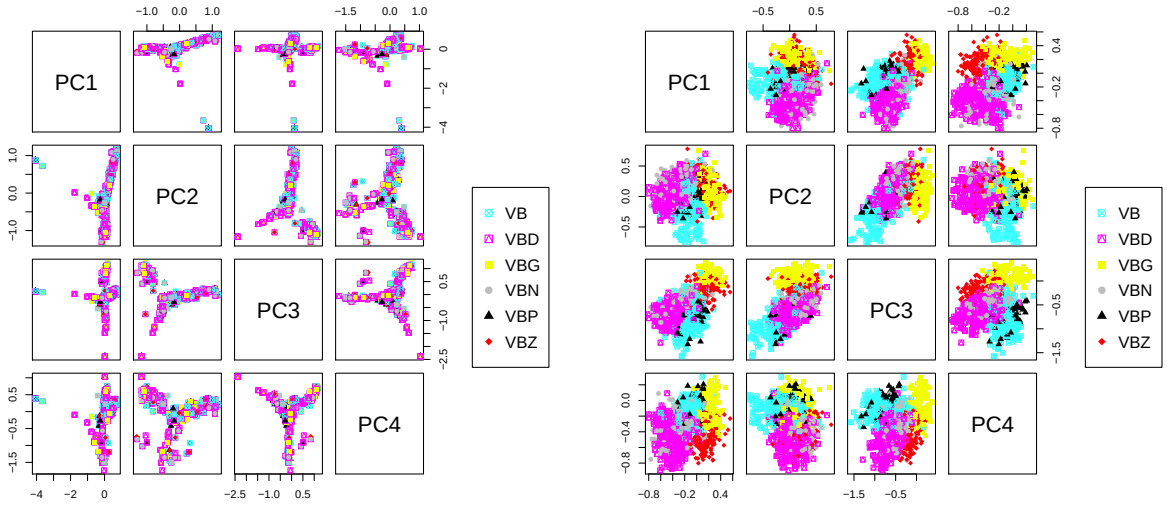
In Table 3, we compare the different settings of the neural networks with a single hidden layer [4] on the development set and test set from PTB. Neural networks with the MAXOUT and L_p (Nos. $3, 4, 5$) significantly outperform the best linear model (No. 1) [5], but the accuracy of the ReLUs (No. 2) was similar to that of the best linear model. According to these results, we argue that the activation function selection is important, although conventional research in NLP has used only a single activation function. It took roughly 7 times as long to learn the hybrid models than the linear model (No. 1). "$L_p(p=2)$ (w/o linear part)" (No. 6) shows the result for a $L_p(p=2)$ model which does not include the linear model f_{linear} for the binary features. Comparing the test results of No. 6 with that of No. 3, the proposed hybrid architecture of a linear model and a neural network enjoys the benefits of both models. Note that No. 6's accuracies of the unknown tokens are relatively competitive, and this may be because the continuous features for the neural network do not include word surfaces.

Since it shows the best accuracy for all tokens on the development set, we refer to $L_p(p=2)$ with 48 hidden variables and the group size of 8 (No. 3 in Table 3) as our representative tagger and denote it as $L_p(p=2)$ in the rest of discussion. In Table 4a, we compare our result with the previously reported results and we see that our tagger outperforms the current state-of-the-art systems on PTB for the accuracies of all tokens and unknown tokens.

In addition, since our tagger was trained using the dependency tree annotations as described in Section 3.3, we compare it with the results of Bohnet and Nivre (2012) which is also trained using both POS tag and dependency annotations. Although their focus is on the dependency pars-

[4] We leave the investigation of deeper neural networks as future work.

[5] For significance tests, we have used the *Wilcoxon matched-pairs signed-rank test* at the 95% confidence level dividing the data into 100 data pairs.



(a) PCA of the raw features

(b) PCA of the hidden activations of $L_p(p = 2)$

Figure 2: Scatter plots of verbs for all combinations between the first four principal components of the raw features and the activation of hidden variables.

ing, they report state-of-the art POS accuracies for many languages. Note that Bohnet and Nivre (2012) also used external resources. Table 4b gives the results for CoNLL2009 data set[6]. Our tagger outperform Bohnet and Nivre (2012), so we believe this is the highest POS accuracy ever reported for a tagger trained on this data set.

Finally, to visualize the learned representations, we applied principal components analysis (PCA) to the hidden activations $\boldsymbol{h}^L$ of the first $10,000$ tokens of the development set from PTB. We also performed PCA to the raw continuous inputs of the same data set. Figure 2 shows the data plots for all the combinations among the first four principal components. We plots only the verb tokens to make the plots easier to see. Figures 2a and 2b show the PCA results of the raw features and the hidden activations of $L_p(p = 2)$, respectively. Compared to Figure 2a, the tokens with the same POS tag are more clearly clustered in Figure 2b. This suggests the neural network learned the good representations for POS tagging and these hidden activations can be used as the input of the succeeding processes, such as parsing.

[6]The accuracies of our tagger on the development set of CoNLL2009 data are 97.76% for all tokens and 93.42% for unknown tokens.

6 Related Work

There is some old work on the POS tagging by neural networks. Nakamura et al. (1990) proposed a neural tagger that predicts the POS tag using a previous POS predictions. Schmid (1994) is most similar to our work. The inputs of his neural network are the POS tag distributions of a word and its suffix in a context window, and he reports a 2% improvement over a regular hidden Markov model. However, his tagger did not use the other kinds of corpus-wide information as we used.

Most of the recent studies on POS tagging use linear models (Suzuki and Isozaki, 2008; Spoustová et al., 2009) or other non-linear models, such as k-nearest neighbor (kNN) (Søgaard, 2011). One trend in these studies is model combinations. Suzuki and Isozaki (2008) combined generative and discriminative models, Spoustová et al. (2009) used the combination of three taggers to generate automatically annotated corpus, and Søgaard (2011) used the outputs of a supervised tagger and an unsupervised tagger as the feature space of the kNN. Our work also follows this trend since neural networks can be considered as non-linear integration of several linear classifiers.

Apart from POS tagging, some previous studies in parsing used the discretization method to handle the combination of continuous features. Bohnet and Nivre (2012) binned the difference of two con-

tinuous features in discrete steps of a predefined small interval. Bansal et al. (2014) used the conjunction of discretized features and studied two discretization methods: One is the binning of real values into discrete steps and the other is a hard clustering of continuous feature vectors. It is not easy to determine the optimal intervals for the binning method, and the clustering method is unsupervised so that the clusters are not guaranteed for good representations of the target tasks.

To capture rich syntactic information for Chinese POS tagging, Sun and Uszkoreit (2012) used the ensemble model of both a POS tagger and a constituency parser. Sun et al. (2013) improved the efficiency of Sun and Uszkoreit (2012) in which a single tagging model is trained using automatically annotated corpus generated by the ensemble tagger. Although the supertag distribution feature in Section 3.3 is a simple way to incorporate syntactic information, automatically parsed large corpora may make the estimate of the supertag distributions more accurate.

7 Conclusion and Future Work

We are studying a neural network approach to handle the non-linear interaction among corpus-wide statistics. For POS tagging, we used word embeddings, POS tag distributions, supertag distributions, and context word distributions in a context window. These features are beneficial, even for linear classifiers, but the neural networks leverage these features for improving tagging accuracies. Our tagger with Maxout networks (Goodfellow et al., 2013) or L_p-pooling (Zhang et al., 2014; Gulcehre et al., 2014) show the state-of-the-art results on two English benchmark sets.

Our empirical results suggest further opportunities to investigate continuous features not only for POS tagging but also for other NLP tasks. An obvious use case for continuous features is the N-best outputs with confidence values, which were predicted by the previous process in a NLP pipeline, such as the POS tags used for syntactic parsing. Another interesting extension is the use of on-the-fly features which reflect previous network states, although the neural networks in our current work do not refer to the prediction history. Recurrent neural networks (RNNs) may be a solution to represent the prediction history in a compact way, and Mesnil et al. (2013) reported that RNNs outperform conditional random fields (CRFs) on a sequential labeling task. They also show the superiority of bi-directional RNNs on their task, so the bi-directional RNNs may also be effective on the POS tagging, since bi-directional inferences were also used in earlier work (Tsuruoka and Tsujii, 2005).

It has a clear benefit over kernel methods in that the test-time computational cost of neural networks is independent from training data. However, although the test-time speed of original kernel methods is proportional to the number of training data, recent development of kernel approximation techniques achieve significant speed improvements (Le et al., 2013; Pham and Pagh, 2013). Since this work shows the non-linearity of continuous features should be exploited, those approximated kernel methods may also improve the tagging accuracies without sacrifice tagging speed.

Independent from our work, Ma et al. (2014) and Santos and Zadrozny (2014) also recently proposed neural network approaches for POS tagging. Ma et al. (2014)'s approach is similar to our approach, with a combination of a linear model and a neural network, although a direct comparison is not easy since their focus is the Web domain adaptation of POS tagging. Remarkably, they report n-gram embeddings are better than single word embeddings. Santos and Zadrozny (2014) proposed character-level embedding to capture the morphological and shape information for POS tagging. Although the reported accuracy (97.32%) on PTB data is lower than state of the art results, their approach is promising for morphologically rich languages. We may study the integration of these embeddings into our approach as future work.

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (ACL)*. The Association for Computer Linguistics.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Com-*

putational Natural Language Learning (EMNLP-CoNLL), pages 1455–1465.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. 2010. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 111–118.

Jinho D. Choi and Martha Palmer. 2012. Fast and robust part-of-speech tagging using dynamic model selection. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (ACL)*, pages 363–367.

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

Koby Crammer and Yoram Singer. 2001. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning Journal*, 75(3):297–325, June.

John C. Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 257–269.

Jesús Giménez and Lluís Màrquez. 2003. Fast and accurate part-of-speech tagging: The svm approach revisited. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*, pages 153–163.

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

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C. Courville, and Yoshua Bengio. 2013. Maxout networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1319–1327.

Ian J. Goodfellow, Mehdi Mirza, Xia Da, Aaron C. Courville, and Yoshua Bengio. 2014. An empirical investigation of catastrophic forgeting in gradient-based neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Caglar Gulcehre, Kyunghyun Cho, Razvan Pascanu, and Yoshua Bengio. 2014. Learned-norm pooling for deep feedforward and recurrent neural networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 1–18.

Fei Huang and Alexander Yates. 2009. Distributional representations for handling sparsity in supervised sequence-labeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL/IJCNLP)*, pages 495–503.

Quoc V. Le, Tamás Sarlós, and Alexander J. Smola. 2013. Fastfood - computing Hilbert space expansions in loglinear time. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 244–252.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL)*.

Ji Ma, Yue Zhang, Tong Xiao, and Jingbo Zhu. 2014. Tagging the Web: Building a robust web tagger with neural network. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (ACL)*. The Association for Computer Linguistics.

Christopher D. Manning. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Proceedings of Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, pages 171–189.

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

H. Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad click prediction: a view from the trenches. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1222–1230.

H. Brendan McMahan. 2011. Follow-the-regularized-leader and mirror descent: Equivalence theorems and L1 regularization. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 525–533.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proceedings of Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 3771–3775.

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

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 807–814.

Tetsuji Nakagawa and Yuji Matsumoto. 2006. Guessing parts-of-speech of unknown words using global information. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (ACL)*.

Masami Nakamura, Katsuteru Maruyama, Takeshi Kawabata, and Kiyohiro Shikano. 1990. Neural network approach to word category prediction for English texts. In *Proceedings of International Conference on Computational Linguistics (COLING)*, pages 213–218.

Hiroki Ouchi, Kevin Duh, and Yuji Matsumoto. 2014. Improving dependency parsers with supertags. In *Proceedings of Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 154–158.

Neal Parikh and Stephen P. Boyd. 2013. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: global vectors for word representation. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Ninh Pham and Rasmus Pagh. 2013. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 239–247.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635.

Cicero Dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1818–1826.

Helmut Schmid. 1994. Part-of-speech tagging with neural networks. In *Proceedings of International Conference on Computational Linguistics (COLING)*, pages 172–176.

Tobias Schnabel and Hinrich Schütze. 2014. FLORS: Fast and simple domain adaptation for part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 2:15–26, February.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 801–809.

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

Anders Søgaard. 2011. Semi-supervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (ACL)*, pages 48–52.

Drahomíra johanka Spoustová, Jan Hajič, Jan Raab, and Miroslav Spousta. 2009. Semi-supervised training for the averaged perceptron pos tagger. In *Proceedings of Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 763–771.

Weiwei Sun and Hans Uszkoreit. 2012. Capturing paradigmatic and syntagmatic lexical relations: Towards accurate Chinese part-of-speech tagging. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (ACL)*, pages 242–252.

Weiwei Sun, Xiaochang Peng, and Xiaojun Wan. 2013. Capturing long-distance dependencies in sequence models: A case study of Chinese part-of-speech tagging. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 180–188.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1139–1147.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using gigaword scale unlabeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (ACL)*, pages 665–673.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 252–259.

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 467–474.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (ACL)*, pages 384–394.

Mengqiu Wang and Christopher D. Manning. 2013. Effect of non-linear deep architecture in sequence labeling. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*.

Xiaohui Zhang, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. 2014. Improving deep neural network acoustic models using generalized maxout networks. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

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

System Combination for Grammatical Error Correction

Raymond Hendy Susanto **Peter Phandi** **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
13 Computing Drive, Singapore 117417
{raymondhs,peter-p,nght}@comp.nus.edu.sg

Abstract

Different approaches to high-quality grammatical error correction have been proposed recently, many of which have their own strengths and weaknesses. Most of these approaches are based on classification or statistical machine translation (SMT). In this paper, we propose to combine the output from a classification-based system and an SMT-based system to improve the correction quality. We adopt the system combination technique of Heafield and Lavie (2010). We achieve an $F_{0.5}$ score of 39.39% on the test set of the CoNLL-2014 shared task, outperforming the best system in the shared task.

1 Introduction

Grammatical error correction (GEC) refers to the task of detecting and correcting grammatical errors present in a text written by a second language learner. For example, a GEC system to correct English promises to benefit millions of learners around the world, since it functions as a learning aid by providing instantaneous feedback on ESL writing.

Research in this area has attracted much interest recently, with four shared tasks organized in the past several years: Helping Our Own (HOO) 2011 and 2012 (Dale and Kilgarriff, 2010; Dale et al., 2012), and the CoNLL 2013 and 2014 shared tasks (Ng et al., 2013; Ng et al., 2014). Each shared task comes with an annotated corpus of learner texts and a benchmark test set, facilitating further research in GEC.

Many approaches have been proposed to detect and correct grammatical errors. The most dominant approaches are based on *classification* (a set of classifier modules where each module addresses a specific error type) and *statistical machine translation (SMT)* (formulated as a translation task from "bad" to "good" English). Other approaches combine the classification and SMT approaches, and often have some rule-based components.

Each approach has its own strengths and weaknesses. Since the classification approach is able to focus on each individual error type using a separate classifier, it may perform better on an error type where it can build a custom-made classifier tailored to the error type, such as subject-verb agreement errors. The drawback of the classification approach is that one classifier must be built for each error type, so a comprehensive GEC system will need to build many classifiers which complicates its design. Furthermore, the classification approach does not address multiple error types that may interact.

The SMT approach, on the other hand, naturally takes care of interaction among words in a sentence as it attempts to find the best overall corrected sentence. It usually has a better coverage of different error types. The drawback of this approach is its reliance on error-annotated learner data, which is expensive to produce. It is not possible to build a competitive SMT system without a sufficiently large parallel training corpus, consisting of texts written by ESL learners and the corresponding corrected texts.

In this work, we aim to take advantage of both the classification and the SMT approaches. By combining the outputs of both systems, we hope that the strengths of one approach will offset the weaknesses of the other approach. We adopt the system combination technique of (Heafield and Lavie, 2010), which starts by creating word-level alignments among multiple outputs. By performing beam search over these alignments, it tries to find the best corrected sentence that combines parts of multiple system outputs.

The main contributions of this paper are as fol-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 951–962,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

lows:

- It is the first work that makes use of a system combination strategy to improve grammatical error correction;
- It gives a detailed description of methods and experimental setup for building component systems using two state-of-the-art approaches; and
- It provides a detailed analysis of how one approach can benefit from the other approach through system combination.

We evaluate our system combination approach on the CoNLL-2014 shared task. The approach achieves an $F_{0.5}$ score of 39.39%, outperforming the best participating team in the shared task.

The remainder of this paper is organized as follows. Section 2 gives the related work. Section 3 describes the individual systems. Section 4 explains the system combination method. Section 5 presents experimental setup and results. Section 6 provides a discussion and analysis of the results. Section 7 describes further experiments on system combination. Finally, Section 8 concludes the paper.

2 Related Work

2.1 Grammatical Error Correction

Early research in grammatical error correction focused on a single error type in isolation. For example, Knight and Chander (1994) built an article correction system for post-editing machine translation output.

The classification approach has been used to deal with the most common grammatical mistakes made by ESL learners, such as article and preposition errors (Han et al., 2006; Chodorow et al., 2007; Tetreault and Chodorow, 2008; Gamon, 2010; Dahlmeier and Ng, 2011; Rozovskaya and Roth, 2011; Wu and Ng, 2013), and more recently, verb errors (Rozovskaya et al., 2014b). Statistical classifiers are trained either from learner or non-learner texts. Features are extracted from the sentence context. Typically, these are shallow features, such as surrounding n-grams, part-of-speech (POS) tags, chunks, etc. Different sets of features are employed depending on the error type addressed.

The statistical machine translation (SMT) approach has gained more interest recently. Earlier work was done by Brockett et al. (2006), where they used SMT to correct mass noun errors. The major impediment in using the SMT approach for GEC is the lack of error-annotated learner ("parallel") corpora. Mizumoto et al. (2011) mined a learner corpus from the social learning platform Lang-8 and built an SMT system for correcting grammatical errors in Japanese. They further tried their method for English (Mizumoto et al., 2012).

Other approaches combine the advantages of classification and SMT (Dahlmeier and Ng, 2012a) and sometimes also include rule-based components. Note that in the hybrid approaches proposed previously, the output of each component system might be only *partially* corrected for some subset of error types. This is different from our system combination approach, where the output of each component system is a *complete* correction of the input sentence where all error types are dealt with.

State-of-the-art performance is achieved by both the classification (Dahlmeier et al., 2012; Rozovskaya et al., 2013; Rozovskaya et al., 2014a) and the SMT approach (Felice et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2014), which motivates us to attempt system output combination from both approaches.

2.2 System Combination

System combination is the task of combining the outputs of multiple systems to produce an output better than each of its individual component systems. In machine translation (MT), combining multiple MT outputs has been attempted in the Workshop on Statistical Machine Translation (Callison-Burch et al., 2009; Bojar et al., 2011).

One of the common approaches in system combination is the confusion network approach (Rosti et al., 2007b). In this approach, a confusion network is created by aligning the outputs of multiple systems. The combined output is generated by choosing the output of one single system as the "backbone", and aligning the outputs of all other systems to this backbone. The word order of the combined output will then follow the word order of the backbone. The alignment step is critical in system combination. If there is an alignment error, the resulting combined output sentence may be ungrammatical.

Rosti et al. (2007a) evaluated three system combination methods in their work:

- **Sentence level** This method looks at the combined N-best list of the systems and selects the best output.
- **Phrase level** This method creates new hypotheses using a new phrase translation table, built according to the phrase alignments of the systems.
- **Word level** This method creates a graph by aligning the hypotheses of the systems. The confidence score of each aligned word is then calculated according to the votes from the hypotheses.

Combining different component sub-systems was attempted by CUUI (Rozovskaya et al., 2014a) and CAMB (Felice et al., 2014) in the CoNLL-2014 shared task. The CUUI system employs different classifiers to correct various error types and then merges the results. The CAMB system uses a pipeline of systems to combine the outputs of their rule based system and their SMT system. The combination methods used in those systems are different from our approach, because they combine individual *sub*-system components, by piping the output from one sub-system to another, whereas we combine the outputs of *whole* systems. Moreover, our approach is able to combine the advantages of both the classification and SMT approaches. In the field of grammatical error correction, our work is novel as it is the first that uses system combination to improve grammatical error correction.

3 The Component Systems

We build four individual error correction systems. Two systems are pipeline systems based on the classification approach, whereas the other two are phrase-based SMT systems. In this section, we describe how we build each system.

3.1 Pipeline

We build two different pipeline systems. Each system consists of a sequence of classifier-based correction steps. We use two different sequences of correction steps as shown in Table 1. As shown by the table, the only difference between the two pipeline systems is that we swap the noun number and the article correction step. We do this because there is an interaction between noun number and article correction. Swapping them generates system outputs that are quite different.

Step	Pipeline 1 (*P1*)	Pipeline 2 (*P2*)
1	Spelling	Spelling
2	Noun number	Article
3	Preposition	Preposition
4	Punctuation	Punctuation
5	Article	Noun number
6	Verb form, SVA	Verb form, SVA

Table 1: The two pipeline systems.

We model each of the article, preposition, and noun number correction task as a multi-class classification problem. A separate multi-class confidence weighted classifier (Crammer et al., 2009) is used for correcting each of these error types. A correction is only made if the difference between the scores of the original class and the proposed class is larger than a threshold tuned on the development set. The features of the article and preposition classifiers follow the features used by the NUS system from HOO 2012 (Dahlmeier et al., 2012). For the noun number error type, we use lexical n-grams, ngram counts, dependency relations, noun lemma, and countability features.

For article correction, the classes are the articles *a*, *the*, and the *null article*. The article *an* is considered to be the same class as *a*. A subsequent post-processing step chooses between *a* and *an* based on the following word. For preposition correction, we choose 36 common English prepositions as used in (Dahlmeier et al., 2012). We only deal with preposition replacement but not preposition insertion or deletion. For noun number correction, the classes are *singular* and *plural*.

Punctuation, subject-verb agreement (SVA), and verb form errors are corrected using rule-based classifiers. For SVA errors, we assume that noun number errors have already been corrected by classifiers earlier in the pipeline. Hence, only the verb is corrected when an SVA error is detected. For verb form errors, we change a verb into its base form if it is preceded by a modal verb, and we change it into the past participle form if it is preceded by *has*, *have*, or *had*.

The spelling corrector uses Jazzy, an open source Java spell-checker[1]. We filter the suggestions given by Jazzy using a language model. We accept a suggestion from Jazzy only if the suggestion increases the language model score of the sentence.

[1] http://jazzy.sourceforge.net/

3.2 Statistical Machine Translation

The other two component systems are based on phrase-based statistical machine translation (Koehn et al., 2003). It follows the well-known log-linear model formulation (Och and Ney, 2002):

$$\begin{aligned}\hat{e} &= \arg\max_{e} P(e|f) \\ &= \arg\max_{e} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(e,f)\right) \quad (1)\end{aligned}$$

where f is the input sentence, e is the corrected output sentence, h_m is a feature function, and λ_m is its weight. The feature functions include a translation model learned from a sentence-aligned parallel corpus and a language model learned from a large English corpus. More feature functions can be integrated into the log-linear model. A decoder finds the best correction $\hat{e}$ that maximizes Equation 1 above.

The parallel corpora that we use to train the translation model come from two different sources. The first corpus is NUCLE (Dahlmeier et al., 2013), containing essays written by students at the National University of Singapore (NUS) which have been manually corrected by English instructors at NUS. The other corpus is collected from the language exchange social networking website Lang-8. We develop two versions of SMT systems: one with two phrase tables trained on NUCLE and Lang-8 separately (*S1*), and the other with a single phrase table trained on the concatenation of NUCLE and Lang-8 data (*S2*). Multiple phrase tables are used with alternative decoding paths (Birch et al., 2007). We add a word-level Levenshtein distance feature in the phrase table used by *S2*, similar to (Felice et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2014). This feature is not included in *S1*.

4 System Combination

We use MEMT (Heafield and Lavie, 2010) to combine the outputs of our systems. MEMT uses METEOR (Banerjee and Lavie, 2005) to perform alignment of each pair of outputs from the component systems. The METEOR matcher can identify exact matches, words with identical stems, synonyms, and unigram paraphrases.

MEMT uses an approach similar to the confusion network approach in SMT system combination. The difference is that it performs alignment on the outputs of every pair of component systems, so it does not need to choose a single backbone. As MEMT does not choose any single system output as its backbone, it can consider the output of each component system in a symmetrical manner. This increases word order flexibility, as choosing a single hypothesis as the backbone will limit the number of possible word order permutations.

After creating pairwise alignments using METEOR, the alignments form a confusion network. MEMT will then perform a beam search over this graph to find the one-best hypothesis. The search is carried out from left to right, one word at a time, creating a partial hypothesis. During beam search, it can freely switch among the component systems, combining the outputs together into a sentence. When it adds a word to its hypothesis, all the words aligned to it in the other systems are also marked as "used". If it switches to another input sentence, it has to use the first "unused" word in that sentence. This is done to make sure that every aligned word in the sentences is used. In some cases, a heuristic could be used to allow skipping over some words (Heafield et al., 2009).

During beam search, MEMT uses a few features to score the hypotheses (both partial hypotheses and full hypotheses):

- **Length** The number of tokens in a hypothesis. It is useful to normalize the impact of sentence length.
- **Language model** Log probability from a language model. It is especially useful in maintaining sentence fluency.
- **Backoff** The average n-gram length found in the language model.
- **Match** The number of n-gram matches between the outputs of the component systems and the hypothesis, counted for small order n-grams.

The weights of these features are tuned using Z-MERT (Zaidan, 2009) on a development set.

This system combination approach has a few advantages in grammatical error correction. METEOR not only can match words with exact matches, but also words with identical stems, synonyms, and unigram paraphrases. This means that it can deal with word form, noun number, and verb form corrections that share identical stems, as well

Data set	# sentences	# source tokens
NUCLE	57,151	1,161,567
Lang-8	1,114,139	12,945,666
CoNLL-2013	1,381	29,207
CoNLL-2014	1,312	30,144
English Wikipedia	86,992,889	1,778,849,655

Table 2: Statistics of the data sets.

as word choice corrections (with synonyms and unigram paraphrases). Also, MEMT uses a language model feature to maintain sentence fluency, favoring grammatical output sentences.

In this paper, we combine the pipeline system $P1$ (Table 1) with the SMT system $S1$, and also combine $P2$ with $S2$. The two component systems in each pair have comparable performance. For our final system, we also combine all four systems together.

5 Experiments

Our approach is evaluated in the context of the CoNLL-2014 shared task on grammatical error correction. Specific details of the shared task can be found in the overview paper (Ng et al., 2014), but we summarize the most important details relevant to our study here.

5.1 Data

We use NUCLE version 3.2 (Dahlmeier et al., 2013), the official training data of the CoNLL-2014 shared task, to train our component systems. The grammatical errors in this corpus are categorized into 28 different error types. We also use the "Lang-8 Corpus of Learner English v1.0"[2] (Tajiri et al., 2012) to obtain additional learner data. English Wikipedia[3] is used for language modeling and collecting n-gram counts. All systems are tuned on the CoNLL-2013 test data (which serves as the development data set) and tested on the CoNLL-2014 test data. The statistics of the data sets can be found in Table 2.

5.2 Evaluation

System performance is evaluated based on precision, recall, and $F_{0.5}$ (which weights precision twice as much as recall). Given a set of n sentences, where $\mathbf{g}_i$ is the set of gold-standard edits for sentence i, and $\mathbf{e}_i$ is the set of system edits for sentence i, precision, recall, and $F_{0.5}$ are defined as follows:

$$P = \frac{\sum_{i=1}^{n} |\mathbf{g}_i \cap \mathbf{e}_i|}{\sum_{i=1}^{n} |\mathbf{e}_i|} \quad (2)$$

$$R = \frac{\sum_{i=1}^{n} |\mathbf{g}_i \cap \mathbf{e}_i|}{\sum_{i=1}^{n} |\mathbf{g}_i|} \quad (3)$$

$$F_{0.5} = \frac{(1 + 0.5^2) \times R \times P}{R + 0.5^2 \times P} \quad (4)$$

where the intersection between $\mathbf{g}_i$ and $\mathbf{e}_i$ for sentence i is defined as

$$\mathbf{g}_i \cap \mathbf{e}_i = \{e \in \mathbf{e}_i | \exists g \in \mathbf{g}_i, match(g, e)\} \quad (5)$$

The official scorer for the shared task was the *MaxMatch* (M^2) scorer[4] (Dahlmeier and Ng, 2012b). The scorer computes the sequence of system edits between a source sentence and a system hypothesis that achieves the maximal overlap with the gold-standard edits. Like CoNLL-2014, $F_{0.5}$ is used instead of F_1 to emphasize precision. For statistical significance testing, we use the sign test with bootstrap re-sampling on 100 samples.

5.3 Pipeline System

We use ClearNLP[5] for POS tagging and dependency parsing, and OpenNLP for chunking[6]. We use the WordNet (Fellbaum, 1998) morphology software to generate singular and plural word surface forms.

The article, preposition, and noun number correctors use the classifier approach to correct errors. Each classifier is trained using multi-class confidence weighted learning on the NUCLE and Lang-8 corpora. The classifier threshold is tuned using a simple grid search on the development data set for each class of a classifier.

5.4 SMT System

The system is trained using Moses (Koehn et al., 2007), with Giza++ (Och and Ney, 2003) for word alignment. The translation table is trained using the "parallel" corpora of NUCLE and Lang-8. The table contains phrase pairs of maximum length seven. We include five standard parameters in the translation table: forward and reverse phrase translations, forward and reverse lexical translations,

[2] http://cl.naist.jp/nldata/lang-8/

[3] http://dumps.wikimedia.org/enwiki/20140102/enwiki-20140102-pages-articles.xml.bz2

[4] http://www.comp.nus.edu.sg/∼nlp/sw/m2scorer.tar.gz

[5] https://code.google.com/p/clearnlp/

[6] http://opennlp.apache.org/

and phrase penalty. We further add a word-level Levenshtein distance feature for *S2*.

We do not use any reordering model in our system. The intuition is that most error types do not involve long-range reordering and local reordering can be easily captured in the phrase translation table. The distortion limit is set to 0 to prohibit reordering during hypothesis generation.

We build two 5-gram language models using the corrected side of NUCLE and English Wikipedia. The language models are estimated using the KenLM toolkit (Heafield et al., 2013) with modified Kneser-Ney smoothing. These two language models are used as separate feature functions in the log-linear model. Finally, they are binarized into a probing data structure (Heafield, 2011). Tuning is done on the development data set with MERT (Och, 2003). We use BLEU (Papineni et al., 2002) as the tuning metric, which turns out to work well in our experiment.

5.5 Combined System

We use an open source MEMT implementation by Heafield and Lavie (2010) to combine the outputs of our systems. Parameters are set to the values recommended by (Heafield and Lavie, 2010): a beam size of 500, word skipping using length heuristic with radius 5, and with the length normalization option turned off. We use five matching features for each system: the number of exact unigram and bigram matches between hypotheses and the number of matches in terms of stems, synonyms, or paraphrases for unigrams, bigrams, and trigrams. We use the Wikipedia 5-gram language model in this experiment.

We tune the combined system on the development data set. The test data is input into both the pipeline and SMT system respectively and the output from each system is then matched using METEOR (Banerjee and Lavie, 2005). Feature weights, based on BLEU, are then tuned using Z-MERT (Zaidan, 2009). We repeat this process five times and use the weights that achieve the best score on the development data set in our final combined system.

5.6 Results

Our experimental results using the CoNLL-2014 test data as the test set are shown in Table 3. Each system is evaluated against the same gold standard human annotations. As recommended in Ng et al. (2014), we do not use the revised gold standard to ensure a fairer evaluation (i.e., without using alternative answers).

System	P	R	$F_{0.5}$
Pipeline			
P1	40.24	23.99	35.44
P2	39.93	22.77	34.70
SMT			
S1	57.90	14.16	35.80
S2	62.11	12.54	34.69
Combined			
P1+S1	53.85	17.65	38.19
P2+S2	56.92	16.22	37.90
P1+P2+S1+S2	53.55	19.14	39.39
Top 4 Systems in CoNLL-2014			
CAMB	39.71	30.10	37.33
CUUI	41.78	24.88	36.79
AMU	41.62	21.40	35.01
POST	34.51	21.73	30.88

Table 3: Performance of the pipeline, SMT, and combined systems on the CoNLL-2014 test set. All improvements of combined systems over their component systems are statistically significant ($p < 0.01$). The differences between P1 and S1 and between P2 and S2 are not statistically significant.

First, we can see that both the pipeline and SMT systems individually achieve relatively good results that are comparable with the third highest ranking participant in the CoNLL-2014 shared task. It is worth noting that the pipeline systems only target the seven most common error types, yet still perform well in an all-error-type setting. In general, the pipeline systems have higher recall but lower precision than the SMT systems.

The pipeline system is also sensitive to the order in which corrections are applied; for example applying noun number corrections before article corrections results in a better score. This means that there is definitely some interaction between grammatical errors and, for instance, the phrase *a houses* can be corrected to *a house* or *houses* depending on the order of correction.

We noticed that the performance of the SMT system could be improved by using multiple translation models. This is most likely due to domain differences between the NUCLE and Lang-8 corpus, e.g., text genres, writing style, topics, etc. Note also that the Lang-8 corpus is more than 10 times larger than the NUCLE corpus, so there

is some benefit from training and weighting two translation tables separately.

The performance of the pipeline system *P1* is comparable to that of the SMT system *S1*, and likewise the performance of *P2* is comparable to that of *S2*. The differences between them are not statistically significant, making it appropriate to combine their respective outputs.

Every combined system achieves a better result than its component systems. In every combination, there is some improvement in precision over the pipeline systems, and some improvement in recall over the SMT systems. The combination of the better component systems (*P1*+*S1*) is also statistically significantly better than the combination of the other component systems (*P2*+*S2*). Combining all four component systems yields an even better result of 39.39% $F_{0.5}$, which is even better than the CoNLL-2014 shared task winner. This is significant because the individual component systems barely reached the score of the third highest ranking participant before they were combined.

6 Discussion

In this section, we discuss the strengths and weaknesses of the pipeline and SMT systems, and show how system output combination improves performance. Specifically, we compare *P1*, *S1*, and *P1*+*S1*, although the discussion also applies to *P2*, *S2*, and *P2*+*S2*.

Type performance. We start by computing the recall for each of the 28 error types achieved by each system. This computation is straightforward as each gold standard edit is also annotated with error type. On the other hand, precision, as mentioned in the overview paper (Ng et al., 2014), is much harder to compute because systems typically do not categorize their corrections by error type. Although it may be possible to compute the precision for each error type in the pipeline system (since we know which correction was proposed by which classifier), this is more difficult to do in the SMT and combined system, where we would need to rely on heuristics which are more prone to errors. As a result, we decided to analyze a sample of 200 sentences by hand for a comparatively more robust comparison. The results can be seen in Table 4.

We observe that the pipeline system has a higher recall than the SMT system for the following error types: *ArtOrDet*, *Mec*, *Nn*, *Prep*, *SVA*, *Vform*, and *Vt*. Conversely, the SMT system generally has a higher precision than the pipeline system. The combined system usually has slightly lower precision than the SMT system, but higher than the pipeline system, and slightly higher recall than the SMT system but lower than the pipeline system. In some cases however, like for *Vform* correction, both precision and recall increase.

The combined system can also make use of corrections which are only corrected in one of the systems. For example, it corrects both Wform and Pform errors, which are only corrected by the SMT system, and SVA errors, which are only corrected by the pipeline system.

Error analysis. For illustration on how system combination helps, we provide example output from the pipeline system *P1*, SMT system *S1*, and the combined system *P1*+*S1* in Table 5. We illustrate three common scenarios where system combination helps: the first is when *P1* performs better than *S1*, and the combined system chooses the corrections made by *P1*, the second is the opposite where *S1* performs better than *P1* and the combined system chooses *S1*, and the last is when the combined system combines the corrections made by *P1* and *S1* to produce output better than both *P1* and *S1*.

7 Additional System Combination Experiments

We further evaluate our system combination approach by making use of the corrected system outputs of 12 participating teams in the CoNLL-2014 shared task, which are publicly available on the shared task website.[7] Specifically, we combined the system outputs of the top $2, 3, \ldots, 12$ CoNLL-2014 shared task teams and computed the results.

In our earlier experiments, the CoNLL-2013 test data was used as the development set. However, the participants' outputs for this 2013 data are not available. Therefore, we split the CoNLL-2014 test data into two parts: the first 500 sentences for the development set and the remaining 812 sentences for the test set. We then tried combining the n best performing systems, for $n = 2, 3, \ldots, 12$. Other than the data, the experimental setup is the same as that described in Section 5.5. Table 6 shows the ranking of the participants on the 812 test sentences (without alter-

[7] http://www.comp.nus.edu.sg/∼nlp/conll14st/official_submissions.tar.gz

Type	Pipeline						SMT						Combined					
	TP	FN	FP	P	R	$F_{0.5}$	TP	FN	FP	P	R	$F_{0.5}$	TP	FN	FP	P	R	$F_{0.5}$
ArtOrDet	13	38	54	19.40	25.49	20.38	11	35	7	61.11	23.91	46.61	16	30	21	43.24	34.78	41.24
Cit	0	0	0	0.00	0.00	0.00	0	0	0	0.00	0.00	0.00	0	0	0	0.00	0.00	0.00
Mec	27	35	43	38.57	43.55	39.47	18	47	8	69.23	27.69	53.25	20	47	10	66.67	29.85	53.48
Nn	27	15	41	39.71	64.29	42.99	5	23	3	62.50	17.86	41.67	11	21	7	61.11	34.38	52.88
Npos	0	10	0	0.00	0.00	0.00	0	9	0	0.00	0.00	0.00	0	9	0	0.00	0.00	0.00
Others	0	1	0	0.00	0.00	0.00	0	3	0	0.00	0.00	0.00	0	3	0	0.00	0.00	0.00
Pform	0	7	0	0.00	0.00	0.00	1	5	0	100.00	16.67	50.00	1	5	0	100.00	16.67	50.00
Pref	1	10	11	8.33	9.09	8.47	0	9	0	0.00	0.00	0.00	0	9	0	0.00	0.00	0.00
Prep	12	25	32	27.27	32.43	28.17	4	26	1	80.00	13.33	40.00	4	27	3	57.14	12.90	33.90
Rloc−	4	16	1	80.00	20.00	50.00	0	16	0	0.00	0.00	0.00	0	16	0	0.00	0.00	0.00
Sfrag	0	1	0	0.00	0.00	0.00	0	0	0	0.00	0.00	0.00	0	0	0	0.00	0.00	0.00
Smod	0	0	0	0.00	0.00	0.00	0	0	0	0.00	0.00	0.00	0	0	0	0.00	0.00	0.00
Spar	0	1	0	0.00	0.00	0.00	0	3	0	0.00	0.00	0.00	0	2	0	0.00	0.00	0.00
Srun	0	2	0	0.00	0.00	0.00	0	1	0	0.00	0.00	0.00	0	1	0	0.00	0.00	0.00
Ssub	0	12	0	0.00	0.00	0.00	1	12	1	50.00	7.69	23.81	1	12	1	50.00	7.69	23.81
SVA	4	11	6	40.00	26.67	36.36	0	14	0	0.00	0.00	0.00	1	14	0	100.00	6.67	26.32
Trans	1	15	0	100.00	6.25	25.00	0	15	0	0.00	0.00	0.00	0	15	0	0.00	0.00	0.00
Um	1	4	0	100.00	20.00	55.56	0	5	0	0.00	0.00	0.00	0	5	0	0.00	0.00	0.00
V0	0	3	0	0.00	0.00	0.00	0	3	3	0.00	0.00	0.00	0	3	3	0.00	0.00	0.00
Vform	4	12	4	50.00	25.00	41.67	3	13	2	60.00	18.75	41.67	5	12	2	71.43	29.41	55.56
Vm	0	2	0	0.00	0.00	0.00	0	5	0	0.00	0.00	0.00	0	6	0	0.00	0.00	0.00
Vt	2	16	1	66.67	11.11	33.33	0	17	0	0.00	0.00	0.00	0	17	0	0.00	0.00	0.00
Wa	0	0	0	0.00	0.00	0.00	0	0	0	0.00	0.00	0.00	0	0	0	0.00	0.00	0.00
Wci	1	60	0	100.00	1.64	7.69	3	52	1	75.00	5.45	21.13	3	55	1	75.00	5.17	20.27
Wform	0	11	0	0.00	0.00	0.00	2	10	2	50.00	16.67	35.71	2	10	2	50.00	16.67	35.71
WOadv	0	0	0	0.00	0.00	0.00	1	1	0	100.00	50.00	83.33	0	1	0	0.00	0.00	0.00
WOinc	0	5	0	0.00	0.00	0.00	0	4	1	0.00	0.00	0.00	0	4	0	0.00	0.00	0.00
Wtone	0	6	0	0.00	0.00	0.00	0	2	0	0.00	0.00	0.00	0	2	0	0.00	0.00	0.00

Table 4: True positives (TP), false negatives (FN), false positives (FP), precision (P), recall (R), and $F_{0.5}$ (in %) for each error type *without* alternative answers, indicating how well each system performs against a particular error type.

System	Example sentence
Source	Nowadays , the use of **the sociall** media platforms is a commonplace in our lives .
P1	Nowadays , the use of **social** media platforms is a commonplace in our lives .
S1	Nowadays , the use of **the sociall** media platforms is a commonplace in our lives .
P1+S1	Nowadays , the use of **social** media platforms is a commonplace in our lives .
Gold	Nowadays , the use of **social** media platforms is commonplace in our lives .
Source	**Human has** their own rights and privacy .
P1	**Human has** their own rights and privacy .
S1	**Humans have** their own rights and privacy .
P1+S1	**Humans have** their own rights and privacy .
Gold	**Humans have** their own rights and privacy .
Source	People **that living** in the modern world really can not live without **the social** media sites .
P1	People **that living** in the modern world really can not live without **social** media sites .
S1	People **living** in the modern world really can not live without **the social** media sites .
P1+S1	People **living** in the modern world really can not live without **social** media sites .
Gold	People **living** in the modern world really can not live without **social** media sites .

Table 5: Example output from three systems.

System	P	R	$F_{0.5}$
CUUI	44.62	27.54	39.69
CAMB	39.93	31.02	37.76
AMU	40.77	21.31	34.47
POST	38.88	23.06	34.19
NTHU	36.30	20.50	31.45
RAC	32.38	13.62	25.39
PKU	30.14	13.12	23.93
UMC	29.03	12.88	23.21
SJTU	32.04	5.43	16.18
UFC	76.92	2.49	11.04
IPN	11.99	2.88	7.34
IITB	28.12	1.53	6.28

Table 6: Performance of each participant when evaluated on 812 sentences from CoNLL-2014 test data.

# systems	P	R	$F_{0.5}$
2	44.72	29.78	40.64
3	56.24	25.04	45.02
4	59.16	23.63	45.48
5	63.41	24.09	47.80
6	65.02	19.54	44.37
7	64.95	18.13	42.83
8	66.09	14.70	38.90
9	70.22	14.81	40.16
10	69.72	13.67	38.31
11	70.23	14.23	39.30
12	69.72	11.82	35.22

Table 7: Performance with different numbers of combined top systems.

native answers). Note that since we use a subset of the original CoNLL-2014 test data for testing, the ranking is different from the official CoNLL-2014 ranking.

Table 7 shows the results of system combination in terms of increasing numbers of top systems. We observe consistent improvements in $F_{0.5}$ when we combine more system outputs, up to 5 best performing systems. When combining 6 or more systems, the performance starts to fluctuate and degrade. An important observation is that when we perform system combination, it is more effective, in terms of $F_{0.5}$, to combine a handful of high-quality system outputs than many outputs of variable quality. Precision tends to increase as more systems are combined although recall tends to decrease. This indicates that combining multiple systems can produce a grammatical error correction system with high precision, which is useful in a practical application setting where high precision is desirable. Figure 1 shows how the performance varies as the number of combined systems increases.

8 Conclusion

We have presented a system combination approach for grammatical error correction using MEMT. Our approach combines the outputs from two of the most common paradigms in GEC: the pipeline and statistical machine translation ap-

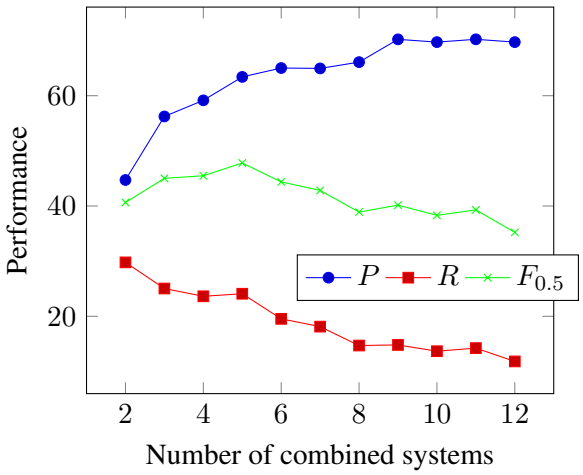


Figure 1: Performance in terms of precision (P), recall (R), and $F_{0.5}$ versus the number of combined top systems.

proach. We created two variants of the pipeline and statistical machine translation approaches and showed that system combination can be used to combine their outputs together to yield a superior system.

Our best combined system achieves an $F_{0.5}$ score of 39.39% on the official CoNLL 2014 test set without alternative answers, higher than the top participating team in CoNLL 2014 on this data set. We achieved this by using component systems which were individually weaker than the top three systems that participated in the shared task.

Acknowledgments

This research is supported by Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2013-T2-1-150. We would like to thank Christopher Bryant for his comments on this paper.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

Alexandra Birch, Miles Osborne, and Philipp Koehn. 2007. CCG supertags in factored statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 9–16.

Ondřej Bojar, Miloš Ercegovčević, Martin Popel, and Omar Zaidan. 2011. A grain of salt for the WMT manual evaluation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 1–11.

Chris Brockett, William B Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 249–256.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28.

Martin Chodorow, Joel R Tetreault, and Na-Rae Han. 2007. Detection of grammatical errors involving prepositions. In *Proceedings of the Fourth ACL-SIGSEM Workshop on Prepositions*, pages 25–30.

Koby Crammer, Mark Dredze, and Alex Kulesza. 2009. Multi-class confidence weighted algorithms. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 496–504.

Daniel Dahlmeier and Hwee Tou Ng. 2011. Grammatical error correction with alternating structure optimization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 915–923.

Daniel Dahlmeier and Hwee Tou Ng. 2012a. A beam-search decoder for grammatical error correction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 568–578.

Daniel Dahlmeier and Hwee Tou Ng. 2012b. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 568–572.

Daniel Dahlmeier, Hwee Tou Ng, and Eric Jun Feng Ng. 2012. NUS at the HOO 2012 shared task. In *Proceedings of the Seventh Workshop on the Innovative Use of NLP for Building Educational Applications*, pages 216–224.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS Corpus of Learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31.

Robert Dale and Adam Kilgarriff. 2010. Helping Our Own: Text massaging for computational linguistics as a new shared task. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 263–267.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on the Innovative Use of NLP for Building Educational Applications*, pages 54–62.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 15–24.

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

Michael Gamon. 2010. Using mostly native data to correct errors in learners' writing: A meta-classifier approach. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 163–171.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Natural Language Engineering*, 12(2):115–129.

Kenneth Heafield and Alon Lavie. 2010. Combining machine translation output with open source: The Carnegie Mellon multi-engine machine translation scheme. *The Prague Bulletin of Mathematical Linguistics*, 93:27–36.

Kenneth Heafield, Greg Hanneman, and Alon Lavie. 2009. Machine translation system combination with flexible word ordering. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 56–60.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The AMU system in the CoNLL-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 25–33.

Kevin Knight and Ishwar Chander. 1994. Automated postediting of documents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 779–784.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 48–54.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 177–180.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In *Proceedings of the Fifth International Joint Conference on Natural Language Processing*, pages 147–155.

Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2012. The effect of learner corpus size in grammatical error correction of ESL writings. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 863–872.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Antti-Veikko I. Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie J. Dorr. 2007a. Combining outputs from multiple machine translation systems. In *Proceedings of the 2007 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 228–235.

Antti-Veikko I. Rosti, Spyros Matsoukas, and Richard Schwartz. 2007b. Improved word-level system combination for machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 312–319.

Alla Rozovskaya and Dan Roth. 2011. Algorithm selection and model adaptation for ESL correction tasks. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 924–933.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, and Dan Roth. 2013. The University of Illinois system in the CoNLL-2013 shared task. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 13–19.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, Dan Roth, and Nizar Habash. 2014a. The Illinois-Columbia system in the CoNLL-2014 shared task. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 34–42.

Alla Rozovskaya, Dan Roth, and Vivek Srikumar. 2014b. Correcting grammatical verb errors. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 358–367.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers*, pages 198–202.

Joel R Tetreault and Martin Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 865–872.

Yuanbin Wu and Hwee Tou Ng. 2013. Grammatical error correction using integer linear programming. In *Proceedings of the 51tst Annual Meeting of the Association for Computational Linguistics*, pages 1456–1465.

Omar Zaidan. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

Dependency parsing with latent refinements of part-of-speech tags

Thomas Müller[†], Richard Farkas[§], Alex Judea[‡], Helmut Schmid[†], and Hinrich Schütze[†]

[†]Center for Information and Language Processing, University of Munich, Germany
[§]Department of Informatics, University of Szeged, Hungary
[‡]Heidelberg Institute for Theoretical Studies, Heidelberg, Germany
muellets@cis.lmu.de

Abstract

In this paper we propose a method to increase dependency parser performance without using additional labeled or unlabeled data by refining the layer of predicted part-of-speech (POS) tags. We perform experiments on English and German and show significant improvements for both languages. The refinement is based on generative split-merge training for Hidden Markov models (HMMs).

1 Introduction

Probabilistic Context-free Grammars with latent annotations (PCFG-LA) have been shown (Petrov et al., 2006) to yield phrase structure parsers with state-of-the-art accuracy. While Hidden Markov Models with latent annotations (HMM-LA) (Huang et al., 2009), stay somewhat behind the performance of state-of-the-art discriminative taggers (Eidelman et al., 2010). In this paper we address the question of whether the resulting latent POS tags are linguistically meaningful and useful for upstream tasks such as syntactic parsing. We find that this is indeed the case, leading to a procedure that significantly increases the performance of dependency parsers. The procedure is attractive because the refinement of predicted part-of-speech sequences using a coarse-to-fine strategy (Petrov and Klein, 2007) is fast and efficient. More precisely, we show that incorporating the induced POS into a state-of-the-art dependency parser (Bohnet, 2010) gives increases in Labeled Attachment Score (LAS): from 90.34 to 90.57 for English and from 87.92 to 88.24 (resp. 88.35 to 88.51) for German without using (resp. with using) morphological features.

2 Related Work

Petrov et al. (2006) introduce generative split-merge training for PCFGs and provide a fully automatic method for training state-of-the-art phrase structure parsers. They argue that the resulting latent annotations are linguistically meaningful. Sun et al. (2008) induce latent sub-states into CRFs and show that noun phrase (NP) recognition can be improved, especially if no part-of-speech features are available. Huang et al. (2009) apply split-merge training to create HMMs with latent annotations (HMM-LA) for Chinese POS tagging. They report that the method outperforms standard generative bigram and trigram tagging, but do not compare to discriminative methods. Eidelman et al. (2010) show that a bidirectional variant of latent HMMs with incorporation of prosodic information can yield state-of-the-art results in POS tagging of conversational speech.

3 Split-Merge Training for HMMs

Split-merge training for HMMs (Huang et al., 2009) iteratively splits every tag into two subtags. Word emission and tag transition probabilities of subtags are then initialized close to the values of the parent tags but with some randomness to break symmetry. Using expectation–maximization (EM) training the parameters can then be set to a local maximum of the training data likelihood. After this split phase, the merge phase reverts splits that only lead to small improvements in the likelihood function in order to increase the robustness of the model. This approach requires an approximation of the gain in likelihood of every split analogous to Petrov et al. (2006) as an exact computation is not feasible.

We have observed that this procedure is not

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 963–967,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

	Universal Tag	Feature	Tag_0	Tag_1
English	Adjectives	$p(w\|t)$	more (0.05) many (0.03) last (0.03)	new (0.03) other (0.03) first (0.02)
	(ADJ)	$p(u\|t)$	**VERB** (0.32) ADV (0.27) NOUN (0.14)	**DET** (0.39) ADP (0.17) ADJ (0.10)
	Particles	$p(w\|t)$	's (0.93) ' (0.07)	to (0.89) up (0.04) out (0.02) off (0.01)
	(PRT)	$p(b\|t)$	**POS** (1.00)	**TO** (0.89) **RP** (0.10)
	Prepositions	$p(w\|t)$	that (0.11) in (0.10) by (0.09)	of (0.43) in (0.19) for (0.11)
	(ADP)	$p(u\|t)$	**VERB** (0.46) NOUN (0.15) . (0.13)	**NOUN** (0.84) NUM (0.06) ADJ (0.03)
	Pronouns	$p(w\|t)$	its (0.30) their (0.15) his (0.14)	it (0.21) he (0.16) they (0.12)
	(PRON)	$p(b\|t)$	**PRP$** (0.68) PRP (0.26) WP (0.05)	**PRP** (0.87) WP (0.11) PRP$ (0.02)
	Verbs	$p(w\|t)$	be (0.06) been (0.02) have (0.02)	is (0.10) said (0.08) was (0.05)
	(VERB)	$p(u\|t)$	**VERB** (0.38) PRT (0.22) ADV (0.11)	**NOUN** (0.52) **PRON** (0.20) . (0.12)
German	Conjunctions	$p(w\|t)$	daß (0.26) wenn (0.08) um (0.06)	und (0.76) oder (0.07) als (0.06)
	(CONJ)	$p(b\|t)$	**KOUS** (0.58) KON (0.30) KOUI (0.06)	**KON** (0.88) KOKOM (0.10) APPR (0.02)
	Particles	$p(w\|t)$	an (0.13) aus (0.10) ab (0.09)	**nicht** (0.49) **zu** (0.46) Nicht (0.01)
	(PRT)	$p(b\|t)$	PTKVZ (0.92) ADV (0.04) ADJD (0.01)	**PTKNEG** (0.52) **PTKZU** (0.44) PTKA (0.02)
	Pronouns	$p(w\|t)$	sich (0.13) die (0.08) es (0.07)	ihre (0.06) seine (0.05) seiner (0.05)
	(PRON)	$p(b\|t)$	**PPER** (0.33) PRF (0.14) PRELS (0.14)	**PPOSAT** (0.40) PIAT (0.34) PDAT (0.16)
	Verbs	$p(w\|t)$	werden (0.04) worden (0.02) ist (0.02)	ist (0.07) hat (0.04) sind (0.03)
	(VERB)	$p(u\|t)$	NOUN (0.46) VERB (0.22) PRT (0.10)	NOUN (0.49) **.** (0.19) PRON (0.16)

Table 1: Induced sub-tags and their statistics, word forms ($p(w|t)$), treebank tag ($p(b|t)$) and preceding Universal tag probability ($p(u|t)$). Bold: linguistically interesting differences.

only a way to increase HMM tagger performance but also yields annotations that are to a considerable extent linguistically interpretable. As an example we discuss some splits that occurred after a particular split-merge step for English and German. For the sake of comparability we applied the split to the Universal Tagset (Petrov et al., 2011). Table 1 shows the statistics used for this analysis. The Universal POS tag set puts the three Penn-Treebank tags RP (particle), POS (possessive marker) and TO into one particle tag (see "PRT" in English part of the table). The training essentially reverses this by splitting particles first into possessive and non-possessive markers and in a subsequent split the non-possessives into TO and particles. For German we have a similar split into verb particles, negation particles like *nicht* 'not' and the infinitive marker *zu* 'to' ("PRT") in the German part of the table). English prepositions get split by proximity to verbs or nouns ("ADP"). Subordinate conjunctions like *that*, which in the Penn-Treebank annotation are part of the preposition tag IN, get assigned to the sub-class next to verbs. For German we also see a separation of "CONJ" into predominantly subordinate conjunctions (Tag 0) and predominantly coordinating conjunctions (Tag 1). For both languages adjectives get split by predicative and attributive use. For English the predicative sub-class also seems to hold rather atypical adjectives like "such" and "last." For English, verbs ("VERB") get split into a predominantly infinite tag (Tag 0) and a predominantly finite tag (Tag 1) while for German we get a separation by verb position. In German we get a separation of pronouns ("PRON") into possessive and non-possessive; in English, pronouns get split by predominant usage in subject position (Tag 0) and as possessives (Tag 1).

Our implementation of HMM-LA has been released under an open-source licence.[1]

In the next section we evaluate the utility of these annotations for dependency parsing.

4 Dependency Parsing

In this section we investigate the utility of induced POS as features for dependency parsing. We run our experiments on the CoNLL-2009 data sets (Hajič et al., 2009) for English and German. As a baseline system we use the latest version of the mate-tools parser (Bohnet, 2010).[3] It was the highest scoring syntactic parser for German and English in the CoNLL 2009 shared task evaluation. The parser gets automatically annotated lemmas, POS and morphological features as input which are part of the CoNLL-2009 data sets.

In this experiment we want to examine the benefits of tag refinements isolated from the improvements caused by using two taggers in parallel, thus we train the HMM-LA on the automatically tagged POS sequences of the training set and use it to add an additional layer of refined POS to the input data of the parser. We do this by calculating the forward-backward charts that are also used in the E-steps during training — in these charts base

[1] https://code.google.com/p/cistern/

[1] Unlabeled Attachment Score

[3] We use v3.3 of Bohnet's graph-based parser.

	#Tags	μ_{LAS}	$\max_{\text{LAS}}$	σ_{LAS}	μ_{UAS}	$\max_{\text{UAS}}$	σ_{UAS}
English	Baseline	88.43			91.46		
	58	88.52	(88.59)	0.06	91.52	(91.61)	0.08
	73	88.55	(88.61)	0.05	91.54	(91.59)	0.04
	92	88.60	(88.71)	0.08	91.60	(91.72)	0.08
	115	88.62	(88.73)	0.07	91.58	(91.71)	0.08
	144	88.60	(88.70)	0.07	91.60	(91.71)	0.07
German (no feat.)	Baseline	87.06			89.54		
	85	87.09	(87.18)	0.06	89.61	(89.67)	0.04
	107	87.23	(87.36)	0.09	89.74	(89.83)	0.08
	134	87.22	(87.31)	0.09	89.75	(89.86)	0.09
German (feat.)	Baseline	87.35			89.75		
	85	87.33	(87.47)	0.11	89.76	(89.88)	0.09
	107	87.43	(87.73)	0.16	89.81	(90.14)	0.17
	134	87.38	(87.53)	0.08	89.75	(89.89)	0.08

Table 2: LAS and UAS[1] mean (μ), best value ($\max$) and std. deviation (σ) for the development set for English and German dependency parsing with (feat.) and without morphological features (no feat.).

tags of the refined tags are constrained to be identical to the automatically predicted tags.

We use 100 EM iterations after each split and merge phase. The percentage of splits reverted in each merge phase is set to .75.

We integrate the tags by adding one additional feature for every edge: the conjunction of latent tags of the two words connected by the edge.

Table 2 shows results of our experiments. All numbers are averages of five independent runs. For English the smaller models with 58 and 73 tags achieve improvements of ≈.1. The improvements for the larger tag sets are ≈.2. The best individual model improves LAS by .3. For the German experiments without morphological features we get only marginal average improvements for the smallest tag set and improvements of ≈.15 for the bigger tag sets. The average ULA scores for 107 and 134 tags are at the same level as the ULA scores of the baseline with morph. features. The best model improves LAS by .3. For German with morphological features the absolute differences are smaller: The smallest tag set does not improve the parser on average. For the tag set of 107 tags the average improvement is .08. The best model improves LAS by .38. In all experiments we see the highest improvements for tag set sizes of roughly the same size (115 for English, 107 for German). While average improvements are low (esp. for German with morphological features), peak improvements are substantial.

Running the best English system on the test set gives an improvement in LAS from 90.34 to 90.57; this improvement is significant[4] ($p < .02$). For German we get an improvement from 87.92 to 88.24 without and from 88.35 to 88.51 with morphological features. The difference between the values without morphological features is significant ($p < .05$), but the difference between models with morphological features is not ($p = .26$). However, the difference between the baseline system with morphological features and the best system without morphological features is also not significant ($p = .49$).

We can conclude that HMM-LA tags can significantly improve parsing results. For German we see that HMM-LA tags can substitute morphological features up to an insignificant difference. We also see that morphological features and HMM-LA seem to be correlated as combining the two gives only insignificant improvements.

5 Contribution Analysis

In this section we try to find statistical evidence for why a parser using a fine-grained tag set might outperform a parser based on treebank tags only.

The results indicate that an induced latent tag set as a whole increases parsing performance. However, not every split made by the HMM-LA seems to be useful for the parser. The scatter plots in Figure 1 show that there is no strict correlation between tagging accuracy of a model and the resulting LAS. This is expected as the latent induction optimizes a tagging objective function, which does not directly translate into better parsing performance. An example is lexicalization. Most latent models for English create a subtag for the preposition "of". This is useful for a HMM as "of" is frequent and has a very specific context. A lexicalized syntactic parser, however, does not benefit from such a tag.

[4] Approx. randomization test (Yeh, 2000) on LAS scores

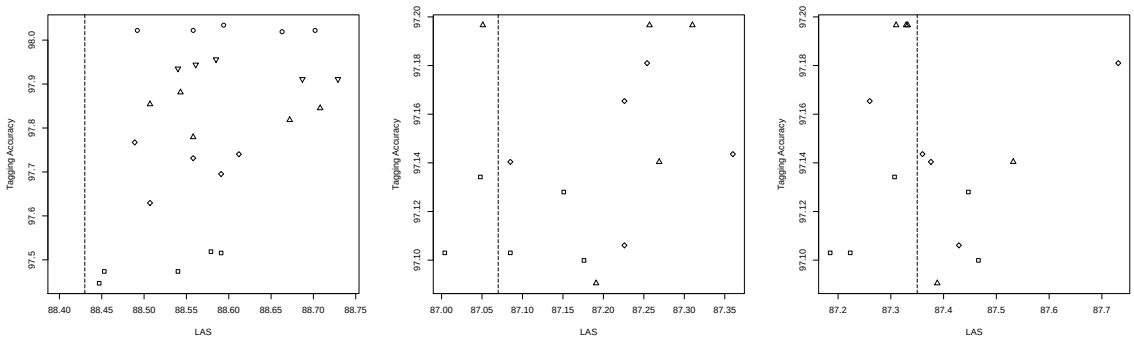


Figure 1: Scatter plots of LAS vs tagging accuracy for English (left) and German without (middle) and with (right) morphological features. English tag set sizes are 58 (squares), 73 (diamonds), 92 (triangles), 115 (triangles pointing downwards) and 144 (circles). German tag set sizes are 85 (squares), 107 (diamonds) and 134 (triangles). The dashed lines indicate the baselines.

We base the remainder of our analysis on the results of the baseline parser on the English development set and the results of the best performing latent model. The best performing model has a LAS score of 88.73 vs 88.43 for the baseline, a difference of .3. If we just look at the LAS of words with incorrectly predicted POS we see a difference of 1.49. A look at the data shows that the latent model helps the parser to identify words that might have been annotated incorrectly. As an example consider plural nouns (NNS) and two of their latent subtags NNS_1 and NNS_2 and how often they get classified correctly and misclassified as proper nouns (NNPS):

	NNS	NNPS
NNS	**2019**	**104**
NNS_1	90	72
NNS_2	**1100**	13
...	...	...

We see that NNS_1 is roughly equally likely to be a NNPS or NNS while NNS_2 gives much more confidence of the actual POS being NNS. So one benefit of HMM-LA POS tag sets are tags of different levels of confidence.

Another positive effect is that latent POS tags have a higher correlation with certain dependency relations. Consider proper nouns (NNP):

	NAME	NMOD	SBJ
NNP	**962**	**662**	**468**
NNP_1	10	27	**206**
NNP_2	24	50	**137**
...	...	...	...

We see that NNP_1 and NNP_2 are more likely to appear in subject relations. NNP_1 contains surnames; the most frequent word forms are *Keating*, *Papandreou* and *Kaye*. In contrast, NNP_2 contains company names such as *Sony*, *NBC* and *Keystone*. This explains why the difference in LAS is twice as high for NNPs as on average.

For German we see similar effects and the anticipated correlation with morphology. The 5 determiner subtags, for example, strongly correlate with grammatical case:

	Nom	Gen	Dat	Acc
ART	**1185**	**636**	**756**	**961**
ART_1	**367**		7	38
ART_2	11	28	**682**	21
ART_3	6	**602**	7	3
ART_4	39		43	**429**
ART_5	**762**	6	17	**470**

6 Conclusion and Future Work

We have shown that HMMs with latent annotations (HMMLA) can generate latent part-of-speech tagsets are linguistically interpretable and can be used to improve dependency parsing. Our best systems improve an English parser from a LAS of 90.34 to 90.57 and a German parser from 87.92 to 88.24 when not using morphological features and from 88.35 to 88.51 when using morphological features . Our analysis of the parsing results shows that the major reasons for the improvements are: the separation of POS tags into more and less trustworthy subtags, the creation of POS subtags with higher correlation to certain dependency labels and for German a correlation of tags and morphological features such as case.

7 Future Work

The procedure works well in general. However, not every split is useful for the parser; e.g., as

discussed above lexicalization increases HMM accuracy, but does not help an already lexicalized parser. We would like to use additional information (e.g., from the dependency trees) to identify useless splits. The different granularities of the hierarchy induced by split-merge training are potentially useful. However, the levels of the hierarchy are incomparable: a child tag is in general not a subtag of a parent tag. We think that coupling parents and children in the tag hierarchy might be one way to force a consistent hierarchy.

Acknowledgments

We would like to thank the anonymous reviewers for their comments. The first author is a recipient of the Google Europe Fellowship in Natural Language Processing, and this research is supported in part by this Google Fellowship and by DFG (grant SFB 732). Most of this work was conducted while the authors worked at the Institute for Natural Language Processing of the University of Stuttgart.

References

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of COLING*.

Vladimir Eidelman, Zhongqiang Huang, and Mary Harper. 2010. Lessons learned in part-of-speech tagging of conversational speech. In *Proceedings of EMNLP*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL*.

Zhongqiang Huang, Vladimir Eidelman, and Mary Harper. 2009. Improving a simple bigram hmm part-of-speech tagger by latent annotation and self-training. In *Proceedings of NAACL*.

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

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of ACL*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *ArXiv:1104.2086v1*.

Xu Sun, Louis-Philippe Morency, Daisuke Okanohara, and Jun'ichi Tsujii. 2008. Modeling latent-dynamic in shallow parsing: a latent conditional model with improved inference. In *Proceedings of COLING*.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of COLING*.

Importance weighting and unsupervised domain adaptation of POS taggers: a negative result

Barbara Plank, Anders Johannsen and Anders Søgaard
Center for Language Technology
University of Copenhagen, Denmark
Njalsgade 140, DK-2300 Copenhagen S
bplank@cst.dk,ajohannsen@hum.ku.dk,soegaard@hum.ku.dk

Abstract

Importance weighting is a generalization of various statistical bias correction techniques. While our labeled data in NLP is heavily biased, importance weighting has seen only few applications in NLP, most of them relying on a small amount of labeled target data. The publication bias toward reporting positive results makes it hard to say whether researchers have tried. This paper presents a negative result on unsupervised domain adaptation for POS tagging. In this setup, we only have *unlabeled* data and thus only indirect access to the bias in emission and transition probabilities. Moreover, most errors in POS tagging are due to unseen words, and there, importance weighting cannot help. We present experiments with a wide variety of weight functions, quantilizations, as well as with randomly generated weights, to support these claims.

1 Introduction

Many NLP tasks rely on the availability of annotated data. The majority of annotated data, however, is sampled from newswire corpora. The performance of NLP systems, e.g., part-of-speech (POS) tagger, parsers, relation extraction systems, etc., drops significantly when they are applied to data that departs from newswire conventions. So while we can extract information, translate and summarize newswire in major languages with some success, we are much less successful processing microblogs, chat, weblogs, answers, emails or literature in a robust way. The main reasons for the drops in accuracy have been attributed to factors such as previously unseen words and bigrams, missing punctuation and capitalization, as well as differences in the marginal distribution of data (Blitzer et al., 2006; McClosky et al., 2008; Søgaard and Haulrich, 2011).

The move from one domain to another (from a *source* to a new *target* domain), say from newspaper articles to weblogs, results in a sample selection bias. Our training data is now biased, since it is sampled from a related, but nevertheless different distribution. The problem of automatically adjusting the model induced from source to a different target is referred to as *domain adaptation*.

Some researchers have studied domain adaptation scenarios, where small samples of labeled data have been assumed to be available for the target domains. This is usually an unrealistic assumption, since even for major languages, small samples are only available from a limited number of domains, and in this work we focus on unsupervised domain adaptation, assuming only unlabeled target data is available.

Jiang and Zhai (2007), Foster et al. (2010; Plank and Moschitti (2013) and Søgaard and Haulrich (2011) have previously tried to use importance weighting to correct sample bias in NLP. Importance weighting means assigning a weight to each training instance, reflecting its importance for modeling the target distribution. Importance weighting is a generalization over post-stratification (Smith, 1991) and importance sampling (Smith et al., 1997) and can be used to correct bias in the labeled data.

Out of the four papers mentioned, only Søgaard and Haulrich (2011) and Plank and Moschitti (2013) considered an unsupervised domain adaptation scenario, obtaining mixed results. These two papers assume *covariate shift* (Shimodaira, 2000), i.e., that there is only a bias in the marginal distribution of the training data. Under this assumption, we can correct the bias by applying a weight function $\frac{P_t(\mathbf{x})}{P_s(\mathbf{x})}$ to our training data points (labeled sentences) and learn from the weighted data. Of course this weight function cannot be

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 968–973,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

computed in general, but we can approximate it in different ways.

In POS tagging, we typically factorize sequences into emission and transition probabilities. Importance weighting can change emission probabilities and transition probabilities by assigning weights to sentences. For instance, if our corpus consisted of three sequences: 1) *a/A b/A*, 2) *a/A b/B*, and 3) *a/A b/B*, then $P(B|A) = 2/3$. If sequences two and three were down-weighted to 0.5, then $P(B|A) = 1/2$.

However, this paper argues that importance weighting cannot help adapting POS taggers to new domains using only unlabeled target data. We present three sources of evidence: (a) negative results with the most obvious weight functions across various English datasets, (b) negative results with randomly sampled weights, as well as (c) an analysis of annotated data indicating that there is little variation in emission and transition probabilities across the various domains.

2 Related work

Most prior work on importance weighting use a *domain classifier*, i.e., train a classifier to discriminate between source and target instances (Søgaard and Haulrich, 2011; Plank and Moschitti, 2013) ($y \in \{s, t\}$). For instance, Søgaard and Haulrich (2011) train a n-gram text classifier and Plank and Moschitti (2013) a tree-kernel based classifier on relation extraction instances. In these studies, $\hat{P}(t|\mathbf{x})$ is used as an approximation of $\frac{P_t(\mathbf{x})}{P_s(\mathbf{x})}$, following Zadrozny (2004). In §3, we follow the approach of Søgaard and Haulrich (2011), but consider a wider range of weight functions. Others have proposed to use kernel mean matching (Huang et al., 2007) or minimizing KL-divergence (Sugiyama et al., 2007).

Jiang and Zhai (2007) use importance weighting to select a subsample of the source data by subsequently setting the weight of all selected data points to 1, and 0 otherwise. However, they do so by relying on a sequential model trained on labeled target data. Our results indicate that the covariate shift assumption fails to hold for cross-domain POS tagging. While the marginal distributions obviously *do* differ (since we can tell domains apart without POS analysis), this is most likely not the only difference. This might explain the positive results obtained by Jiang and Zhai (2007). We will come back to this in §4.

Cortes et al. (2010) show that importance weighting potentially leads to over-fitting, but propose to use quantiles to obtain more robust weight functions. The idea is to rank all weights and obtain q quantiles. If a data point $\mathbf{x}$ is weighted by w, and w lies in the ith quantile of the ranking ($i \leq q$), $\mathbf{x}$ is weighted by the average weight of data points in the ith quantile.

The weighted structured perceptron (§3) used in the experiments below was recently used for a different problem, namely for correcting for bias in annotations (Plank et al., 2014).

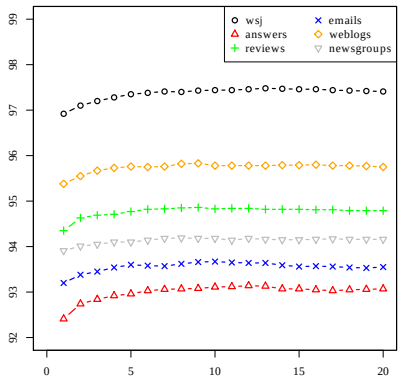


Figure 1: Training epochs vs tagging accuracy for the baseline model on the dev data.

3 Experiments

3.1 Data

We use the data made available in the SANCL 2012 Shared Task (Petrov and McDonald, 2012). The training data is the OntoNotes 4.0 release of the Wall Street Journal section of the Penn Treebank, while the target domain evaluation data comes from various sources, incl. Yahoo Answers, user reviews, emails, weblogs and newsgroups. For each target domain, we have both development and test data.

3.2 Model

In the weighted perceptron (Cavallanti et al., 2006), we make the learning rate dependent on the current instance $\mathbf{x}_n$, using the following update:

$$\mathbf{w}^{i+1} \leftarrow \mathbf{w}^i + \beta_n \alpha(y_n - \text{sign}(\mathbf{w}^i \cdot \mathbf{x}_n))\mathbf{x}_n \quad (1)$$

where β_n is the weight associated with $\mathbf{x}_n$. See Huang et al. (2007) for similar notation.

We extend this idea straightforwardly to the structured perceptron (Collins, 2002), for which

System	Answers	Newsgroups	Reviews	Avg	Emails	Weblogs	WSJ
Our system	91.08	91.57	91.59	91.41	87.97	92.19	97.32
SANCL12-2nd	90.99	92.32	90.65	91.32	–	–	97.76
SANCL12-best	91.79	93.81	93.11	92.90	–	–	97.29
SANCL12-last	88.24	89.70	88.15	88.70	–	–	95.14
FLORS basic	91.17	92.41	92.25	88.67	91.37	97.11	91.94

Table 1: Tagging accuracies and comparison to prior work on the SANCL test sets (fine-grained POS).

we use an in-house implementation. We use commonly used features, i.e., w,w_{-1}, w_{-2}, w_{+1}, w_{+2}, digit, hyphen, capitalization, pre-/suffix features, and Brown word clusters. The model seems robust with respect to number of training epochs, cf. Figure 1. Therefore we fix the number of epochs to five and use this setting in all our experiments. Our code is available at: `https://bitbucket.org/bplank/importance-weighting-exp`.

3.3 Importance weighting

In our first set of experiments, we follow Søgaard and Haulrich (2011) in using document classifiers to obtain weights for the source instances. We train a text classifier that discriminates the two domains (source and target). For each sentence in the source and target domain (the unlabeled text that comes with the SANCL data), we mark whether it comes from the source or target domain and train a binary classifier (logistic regression) to discriminate between the two. For every sentence in the source we obtain its probability for the target domain by doing 5-fold cross-validation. While Søgaard and Haulrich (2011) use only token-based features (word n-grams $\leq$ 3), we here exploit a variety of features: word token n-grams, and two generalizations: using Brown clusters (estimated from the union of the 5 target domains), and Wiktionary tags (if a word has multiple tags, we assign it the union of tags as single tag; OOV words are marked as such).

The distributions of weights can be seen in the upper half of Figure 2.

3.3.1 Results

Table 1 shows that our baseline model achieves state-of-the-art performance compared to SANCL (Petrov and McDonald, 2012)[1] and FLORS (Schnabel and Schütze, 2014). Our results align well with the second best POS tagger in the SANCL 2012 Shared Task. Note

[1] `https://sites.google.com/site/sancl2012/home/shared-task/results`

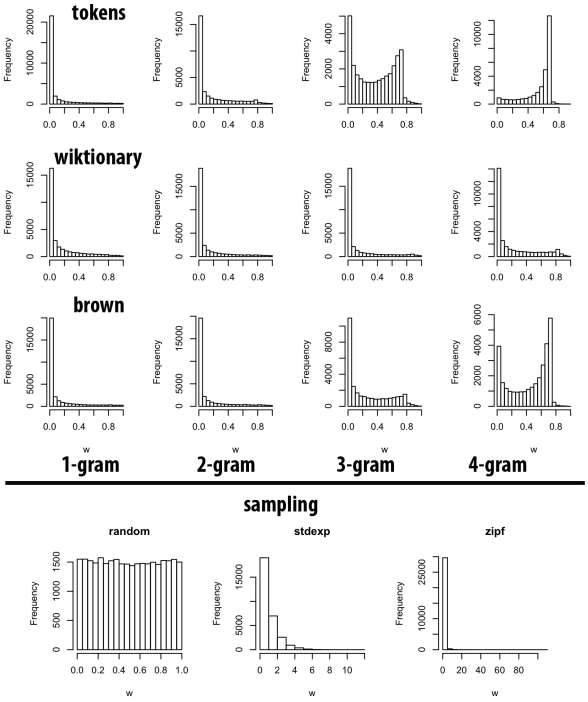


Figure 2: Histogram of different weight functions.

that the best tagger in the shared task explicitly used normalization and various other heuristics to achieve better performance. In the rest of the paper, we use the universal tag set part of the SANCL data (Petrov et al., 2012).

Figure 3 presents our results on development data for different importance weighting setups. None of the above weight functions lead to significant improvements on *any* of the datasets. We also tried scaling and binning the weights, as suggested by Cortes et al. (2010), but results kept fluctuating around baseline performance, with no significant improvements.

3.4 Random weighting

Obviously, weight functions based on document classifiers may simply not characterize the relevant properties of the instances and hence lead to bad re-weighting of the data. We consider three random sampling strategies, namely sampling random uniforms, random exponentials, and random

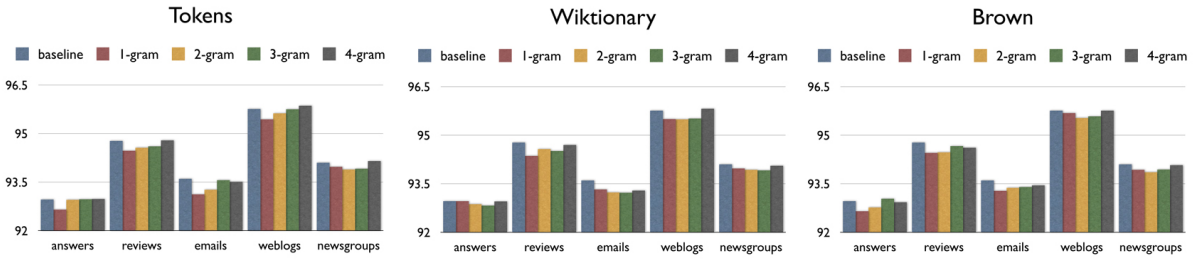


Figure 3: Results on development data for different weight functions, i.e., document classifiers trained on a) raw tokens; b) tokens replaced by Wiktionary tags; c) tokens replaced by Brown cluster ids. The weight was the raw $p_t(y|x)$ value, no scaling, no quantiles. Replacing only open-class tokens for b) and c) gave similar or lower performance.

Zipfians and ran 500 samples for each. For these experiments, we estimate significance cut-off levels of tagging accuracies using the approximate randomization test. To find the cut-off levels, we randomly replace labels with gold labels until the achieved accuracy significantly improves over the baseline for more than 50% of the samples. For each accuracy level, 50 random samples were taken.

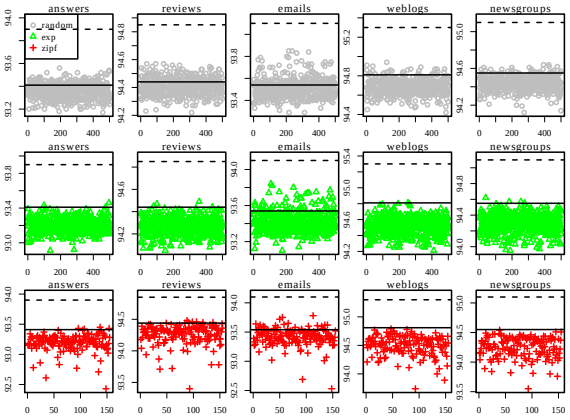


Figure 4: Random weight functions (500 runs each) on test sets. Solid line is the baseline performance, while the dashed line is the p-value cut-off. From top: random, exponential and Zipfian weighting. All runs fall below the cut-off.

3.4.1 Results

The dashed lines in Figure 4 show the p-value cut-offs for positive results. We see that most random weightings of data lead to slight drops in performance or are around baseline performance, and no weightings lead to significant improvements. Random uniforms seem slightly better than exponentials and Zipfians.

domain (tokens)	**avg tag ambiguity**		**OOV**	**KL**	ρ
	type	token			
wsj (train/test: 731k/39k)	1.09	1.41	11.5	0.0006	0.99
answers (28k)	1.09	1.22	27.7	0.048	0.77
reviews (28k)	1.07	1.19	29.5	0.040	0.82
emails (28k)	1.07	1.19	29.9	0.027	0.92
weblogs (20k)	1.05	1.11	22.1	0.010	0.96
newsgroups (20k)	1.05	1.14	23.1	0.011	0.96

Table 2: Relevant statistics for our analysis (§4) on the test sets: average tag ambiguity, out-of-vocabulary rate, and KL-divergence and Pearson correlation coefficient (ρ) on POS bigrams.

4 Analysis

Some differences between the gold-annotated source domain data and the gold-annotated target data used for evaluation are presented in Table 2. One important observation is the low ambiguity of word forms in the data. This makes the room for improvement with importance weighting smaller. Moreover, the KL divergencies over POS bigrams are also very low. This tells us that transition probabilities are also relatively constant across domains, again suggesting limited room for improvement for importance weighting.

Compared to this, we see much bigger differences in OOV rates. OOV rates do seem to explain most of the performance drop across domains. In order to verify this, we implemented a version of our structured perceptron tagger with type-constrained inference (Täckström et al., 2013). This technique only improves performance on unseen words, but nevertheless we saw significant improvements across all five domains (cf. Table 3). This suggests that unseen words are a more important problem than the marginal distribution of data for unsupervised domain adaptation of POS taggers.

	ans	rev	email	webl	newsg
base	93.41	94.44	93.54	94.81	94.55
+type constr.	94.09†	94.85†	94.31†	95.99†	95.97†
p-val cut-off	93.90	94.85	94.10	95.3	95.10

Table 3: Results on the test sets by adding Wiktionary type constraints. †=p-value < 0.001.

We also tried Jiang and Zhai's subset selection technique (§3.1 in Jiang and Zhai (2007)), which assumes labeled training material for the target domain. However, we did not see any improvements. A possible explanation for these different findings might be the following. Jiang and Zhai (2007) use labeled target data to learn their weighting model, i.e., in a supervised domain adaptation scenario. This potentially leads to very different weight functions. For example, let the source domain be 100 instances of $a/A\ b/B$ and 100 instances of $b/B\ b/B$, and the target domain be 100 instances of $a/B\ a/B$. Note that a domain classifier would favor the first 100 sentences, but in an HMM model induced from the labeled target data, things look very different. If we apply Laplace smoothing, the probability of $a/A\ b/B$ according to the target domain HMM model would be $\sim 8.9e^{-7}$, and the probability of $b/B\ b/B$ would be $\sim 9e^{-5}$. Note also that this set-up does not assume covariate shift.

5 Conclusions and Future Work

Importance weighting, a generalization of various statistical bias correction techniques, can potentially correct bias in our labeled training data, but this paper presented a negative result about importance weighting for unsupervised domain adaptation of POS taggers. We first presented experiments with a wide variety of weight functions, quantilizations, as well as with randomly generated weights, none of which lead to significant improvements. Our analysis indicates that most errors in POS tagging are due to unseen words, and what remains seem to not be captured adequately by unsupervised weight functions.

For future work we plan to extend this work to further weight functions, data sets and NLP tasks.

Acknowledgements

This research is funded by the ERC Starting Grant LOWLANDS No. 313695.

References

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. 2006. Tracking the best hyperplane with a simple budget perceptron. In *COLT*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

Corinna Cortes, Yishay Mansour, and Mehryar Mohri. 2010. Learning bounds for importance weighting. In *NIPS*.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *EMNLP*.

Jiayuan Huang, Alexander Smola, Arthur Gretton, Karsten Borgwardt, and Bernhard Schölkopf. 2007. Correcting sample bias by unlabeled data. In *NIPS*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *ACL*.

David McClosky, Eugene Charniak, and Mark Johnson. 2008. When is self-training effective for parsing? In *COLING*.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 Shared Task on Parsing the Web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *LREC*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *ACL*.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014. Learning part-of-speech taggers with inter-annotator agreement loss. In *EACL*.

Tobias Schnabel and Hinrich Schütze. 2014. Flors: Fast and simple domain adaptation for part-of-speech tagging. *TACL*, 2:15–16.

Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244.

Peter Smith, Mansoor Shafi, and Hongsheng Gao. 1997. Quick simulation: A review of importance sampling techniques in communications systems. *IEEE Journal on Selected Areas in Communications*, 15(4):597–613.

T.M.F. Smith. 1991. Post-stratification. *The Statistician*, 40:315–323.

Anders Søgaard and Martin Haulrich. 2011. Sentence-level instance-weighting for graph-based and transition-based dependency parsing. In *IWPT*.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. 2007. Direct importance estimation with model selection and its application to covariate shift adaptation. In *NIPS*.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *TACL*, 1:1–12.

Bianca Zadrozny. 2004. Learning and evaluating classifiers under sample selection bias. In *ICML*.

POS Tagging of English-Hindi Code-Mixed Social Media Content

Yogarshi Vyas*
University of Maryland
yogarshi@cs.umd.edu

Spandana Gella*
Xerox Research Centre Europe
spandanagella@gmail.com

Jatin Sharma **Kalika Bali** **Monojit Choudhury**
Microsoft Research India
{jatin.sharma,kalikab,monojitc}@microsoft.com

Abstract

Code-mixing is frequently observed in user generated content on social media, especially from multilingual users. The linguistic complexity of such content is compounded by presence of spelling variations, transliteration and non-adherance to formal grammar. We describe our initial efforts to create a multi-level annotated corpus of Hindi-English code-mixed text collated from Facebook forums, and explore language identification, back-transliteration, normalization and POS tagging of this data. Our results show that language identification and transliteration for Hindi are two major challenges that impact POS tagging accuracy.

1 Introduction

Code-Switching and *Code-Mixing* are typical and well-studied phenomena of multilingual societies (Gumperz, 1964; Auer, 1984; Myers-Scotton, 1993; Danet and Herring, 2007; Cardenas-Claros and Isharyanti, 2009). Linguists differentiate between the two, where Code-Switching is juxtaposition within the same speech exchange of passages of speech belonging to two different grammatical systems or sub-systems (Gumperz, 1982), and Code-Mixing (CM) refers to the embedding of linguistic units such as phrases, words and morphemes of one language into an utterance of another language (Myers-Scotton, 1993). The first example in Fig. 1 features CM where English words are embedded in a Hindi sentence, whereas the second example shows codeswitching. Here, we will use CM to imply both. Work on computational models of CM have been few and far between (Solorio and Liu, 2008a; Solorio and Liu, 2008b; Nguyen and Dogruoz, 2013), primarily due to the paucity of CM data in conventional text-corpora which makes data-intensive methods hard to apply. Solorio and Liu (2008a) in their work on English-Spanish CM use models built on smaller datasets to predict valid switching points to synthetically generate data from monolingual corpora, and in another work (2008b) describe parts-of-speech (POS) tagging of CM text.

CM though typically observed in spoken language is now increasingly more common in text, thanks to the proliferation of the Computer Mediated Communication channels, especially social media like Twitter and Facebook (Crystal, 2001; Herring, 2003; Danet and Herring, 2007; Cardenas-Claros and Isharyanti, 2009). Social media content is tremendously important for studying trends, reviews, events, human-behaviour as well as linguistic analysis, and therefore in recent times has spurred a lot of interest in automatic processing of such data. Nevertheless, CM on social media has not been studied from a computational aspect. Moreover, social media content presents additional challenges due to contractions, non-standard spellings and non-grammatical constructions. Furthermore, for languages written in scripts other than Roman, like Hindi, Bangla, Japanese, Chinese and Arabic, Roman transliterations are typically used for representing the words (Sowmya et al., 2010). This can prove a challenge for language identification and segregation of the two languages.

In this paper, we describe our initial efforts to POS tag social media content from English-Hindi (henceforth **En-Hi**) bilinguals while trying to address the challenges of CM, transliteration and non-standard spelling, as well as lack of annotated data. POS tagging is one of the fundamental pre-processing steps for NLP, and while there

*This work was done during authors' internship at Microsoft Research India.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 974–979,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

have been works on POS tagging of social media data (Gimpel et al., 2011; Owoputi et al., 2013) and of CM (Solorio and Liu, 2008b), but we do not know of any work on POS tagging of CM text from social media that involves transliteration. The salient contributions of this work are in formalizing the problem and related challenges for processing of **En-Hi** social media data, creation of an annotated dataset and some initial experiments for language identification, transliteration, normalization and POS tagging of this data.

2 Corpus Creation

For this study, we collected data from Facebook public pages of three celebrities: Amitabh Bachchan, Shahrukh Khan, Narendra Modi, and the BBC Hindi news page. All these pages are very popular with 1.8 to 15.5 million "likes". A total of 40 posts were manually selected from these pages, which were published between 22nd – 28th October 2013. The posts having a long thread of comments (50+) were preferred, because CM and non-standard usage of language is more common in the comments. We shall use the term *post* to refer to either a post or a comment. The corpus thus created has 6,983 posts and 113,578 words. The data was semi-automatically cleaned and formatted. The user names were removed for anonymity, but the names appearing in comments, which are mostly of celebrities, were retained.

2.1 Annotation

There are various interesting linguistic as well as socio-pragmatic features (e.g., user demographics, presence of sarcasm or humor, polarity) for which this corpus could be annotated because CM is influenced by both linguistic as well as extra-linguistic features. However, initial attempts at such detailed and layered annotation soon revealed the resource-intensiveness of the task. We, thus, scaled down the annotation to the following four layers:

Matrix: The posts are split into contiguous fragments of words such that each fragment has a unique *matrix language* (either **En** or **Hi**). The matrix language is defined as the language which governs the grammatical relation between the constituents of the utterance. Any other language words that are nested into the matrix constitute the *embedded* language(s). Usually, matrix language can be assigned to clauses or sentences.

Word origin: Every word is marked for its origin or source language, **En** or **Hi**, depending on whether it is an English or Hindi word. Words that are of neither Hindi nor English origin are marked as **Ot** or Other. Here, we assume that code-mixing does not happen at sublexical levels, as it is uncommon in this data; **Hi** and **En** have a simpler inflectional morphology and thus, sub-lexical mixing though present (e.g., `computeron` has a **En** root - `computer` and a **Hi** plural marker `on`) is relatively less common. In languages with richer morphology and agglutination, like Bangla and most Dravidian languages, more frequent sublexical mixing may be observed. Also note that words are borrowed extensively between **Hi** and **En** such that certain English words (e.g., bus, party, vote etc) are no longer perceived as English words by the Hindi speakers. However, here we will not distinguish between CM and borrowing, and such borrowed English words have also been labeled as **En** words.

Normalization/Transliteration: Whenever the word is in a transliterated form, which is often the case for the **Hi** words, it is labeled with the intended word in the native script (e.g., Devanagari for **Hi**). If the word is in native script, but uses a non-standard spelling, it is labeled with the correct standard spelling. We call this the spelling normalization layer.

Parts-of-Speech (POS): Finally, each word is also labeled with its POS. We use the Universal POS tagset proposed by Petrov et al. (2011) which has 12 POS tags that are applicable to both **En** and **Hi**. The POS labels are decided based on the function of a word in the context, rather than a decontextualized lexical category. This is an important notion, especially for CM text, because often the original lexical category of an embedded word is lost in the context of the matrix language, and it plays the role of a different lexical category. Though the Universal POS tagset does not prescribe a separate tag for Named Entities, we felt the necessity of marking three different kinds of NEs - people, location and organization, because almost every comment has one or more NEs and strictly speaking word origin does not make sense for these words.

Annotation Scheme: Fig. 1 illustrates the annotation scheme through two examples. Each post is enclosed within <s></s> tags. The matrices within a post are separated by the <matrix></matrix> tags which take the matrix language as an argument. Each word is anno-

```
<s>
      <matrix name="Hindi">
      love_NOUN/E affection_NOUN/E lekar_VERB=”ले कर” salose_NOUN=सालों
se_ADP=से sunday_NOUN/e ke_ADP=के din_NOUN=दिन chali_VERB=चली aarahi_VERB=”आ
रही” divine_ADJ/e parampara_NOUN=परंपरा ko_ADP=को age_NOUN=आगे badhha_VERB=बढा
rahe_VERB=रहे ho_VERB=हो
      </matrix>
</s>
 <s>
       <matrix name="Hindi">
       jindagi_NOUN=ज़िंदगी kaise_PRON=कैसी h_VERB=है paheli_NOUN=पहेली
 haye_PRT=हाये
       </matrix>
       <matrix name="English">
       may_ADP his_PRON sol_NOUN=soul rest_VERB in_ADP peace_NOUN
       </matrix>
 </s>
```

Figure 1: Two example annotations.

tated for POS, and the language (/E or /H for **En** or **Hi** respectively) only if it is different from the language of the matrix. In case of non-standard spelling in English, the correct spelling is appended as “sol_NOUN=soul”, while for the Hindi words, the correct Devanagari transliteration is appended. The NEs are marked with the tags P (person), L (location) or O (organization) and multiword NEs are enclosed within square brackets “[]”.

A random subsample of 1062 posts consisting of 10171 words were annotated by a linguist who is a native speaker of **Hi** and proficient in **En**. The annotations were reviewed and corrected by two experts linguists. During this phase, it was also observed that a large number of comments were very short, typically an eulogism of their favorite celebrity and hence were not interesting from a linguistic point of view. For our experiments, we removed all posts that had fewer than 5 words. The resulting corpus had 381 comments/posts and 4135 words.

2.2 CM Distribution

Most of the posts (93.17%) are in Roman script, and only 2.93% were in Devanagari. Around 3.5% of the posts contain words in both the scripts (typically a post in Devanagari with hashtags or urls in Roman script), and a very small fraction of the text (0.4% of comments/posts and 0.6% words) was in some other script. The fraction of words present in Roman and Devanagri scripts are 80.76% and 15.32% respectively, which shows that the Devanagari posts are relatively longer than the Roman posts. Due to their relative rarity, the posts containing words in Devanagari or any other script were not considered for annotation.

In the annotated data, 1102 sentences are in a single matrix (398 **Hi**, 698 **En** and 6 **Ot**) and in 45 posts there is at least one switch of matrix (mostly between **Hi** and **En**. Thus, 4.2% of the data shows *code-switching*. This is a strict definition of code-switching; if we consider a change in matrix within a conversation thread as a code-switch, then in this data all the threads exhibit code-switching. However, out of the 398 comments in **Hi**-matrix, 23.37% feature CM (i.e., they have at least one or more non-**Hi** (or rather, almost always **En**) words embedded. On the other hand, only 7.34% **En**-matrix comments feature CM (again almost always with **Hi**). Thus, a total of 17.2% comments/posts, which contains a quarter of all the words in the annotated corpus, feature either CM or code-switching or both. We also note that more than 40% words in the corpus are in **Hi** or other Indian languages, but written in Roman script; hence, they are in transliterated form. See (Bali et al., 2014) for an in-depth discussion on the characteristics of the CM data.

This analysis demonstrates the necessity of CM and transliterated text processing in the context of Indian user-generated social media content. Perhaps, the numbers are not too different for such content generated by the users of any other bilingual and multilingual societies.

3 Models and Experiments

POS tagging of **En-Hi** code-mixed data requires language identification at both word and matrix level as well back-transliteration of the text into

Actual Label	Predicted Label Hi	En	Recall
Hi	1057	515	0.672
En	45	2023	0.978
Precision	0.959	0.797	

Table 1: Confusion matrix, precision and recall of the language identification module.

the native script. Additionaly, since we are working with content from social media, the usage of non-standard spelling is rampant and thus, normalization of text into some standard form is required. Ideally, these tasks should be performed jointly since they are interdependent. However, due to lack of resources, we implement a pipelined approach in which the tasks - language identification, text normalization and POS tagging - are performed sequentially, in that order. This pipelined approach also allows us to use various off-the-shelf tools for solving these subtasks and quickly create a baseline system. The baseline results can also provide useful insight into the inherent hardness of POS tagging of code-mixed social media text. In this section, we first describe our approach to solve these three tasks, and then discuss the experiments and results.

3.1 Language identification

Langauge identification is a well studied problem (King and Abney, 2013; Carter et al., 2013; Goldszmidt et al., 2013; Nguyen and Dogruoz, 2013), though for CM text, especially those involving transliterations and orthographic variation, this is far from a solved problem (Nguyen and Dogruoz, 2013). There was a shared task in FIRE 2013 (Saha Roy et al., 2013) on language identification and back transliteration for **En** mixed with **Hi**, Bangla and Gujarati. Along the lines of Gella et al (Gella et al., 2013), which was the best performing system in this shared task, we used the word-level logistic regression classifier built by King and Abney (2013). This system provides a source language with a confidence probability for each word in the test set. We trained the classifier on 3201 English words extracted from the SMS corpus developed by Choudhury et al (2007), while the Hindi data was obtained by sampling 3218 Hindi transliterations out of the **En-Hi** transliteration pairs developed by Sowmya et al. (Sowmya et al., 2010). Ideally, the context of a token is important for identifying the language. Again, following (Gella et al., 2013) we incorporate context information through a code-switching probability, P_s. A higher value of P_s implies a lower probability of code-switching, i.e., adjacent words are more likely to be in the same language.

Table 1 shows the token (word) level confusion matrix for the language identification task on our dataset. The language labels of 84.6% of the tokens were correctly predicted by the system. As can be seen from the Table, the precision for predicting **Hi** is high, whereas that for **En** is low. This is mainly due to the presence of a large number of contracted and distorted **Hi** words in the dataset, e.g. `h` for `hai` (Fig. 1), which were tagged as **En** by our system because the training examples had no contracted **Hi** words, but short and non-conventional spellings were in plenty in the **En** training examples as those were extracted from the SMS corpus.

3.2 Normalization

In our dataset, if a word is identified as **Hi**, then it must be back-transliterated to Devanagari script so that any off-the-shelf Hindi POS tagger can be used. We used the system by Gella et al. (Gella et al., 2013) for this task, which is part rule-based and part statistical. The system was trained on the 35000 unique transliteration pairs extracted from Hindi song lyrics (Gupta et al., 2012). This corpus has a reasonably wide coverage of Hindi words, and past researchers have also shown that transliteration does not require a very large amount of training data. Normalization of the **En** text was not needed because the POS tagger (Owoputi et al., 2013) could handle unnormalized text.

3.3 POS tagging

Solorio and Liu (2008b) describes a few approaches to POS-tagging of code-switched Spanglish text, all of which primarily relies on two monolingual taggers and certain heuristics to combine the output from the two. One of the simpler heuristics is based on language identification, where the POS tag of a word is the output of the monolingual tagger of the language in which the word is. In this initial study, we apply this basic idea for POS tagging of CM data. We divide the text (which is already sentence-separated) into contiguous maximal chunks of words which are in the same language. Then we apply a **Hi** POS tagger to the **Hi** chunks, and an **En** POS tagger to the **En** chunks.

Model	LI	HN	Tagger	Hi Acc.	En Acc.	Total Acc.	Hi CA	En CA	Total CA
1a	K	K	Standard	75.14	81.91	79.02	27.34	39.67	34.05
1b	K	K	Twitter	75.14	82.66	79.02	27.34	35.74	31.91
2	K	NK	Twitter	65.61	81.73	74.87	17.58	33.77	26.38
3	NK	NK	Twitter	44.74	80.68	65.39	40.00	13.17	25.00

Table 2: POS Tagging accuracies for the different models. K=Known, NK = Not Known. LI = Language labels, HN = Hindi normalized forms, Acc. = Token level accuracy, CA = Chunk level accuracy.

We use a CRF++ based POS tagger for **Hi**, which is freely available from `http://nltr.org/snltr-software/`. For **En**, we use the Twitter POS tagger (Owoputi et al., 2013). It also has an inbuilt tokenizer and can work directly on unnormalized text. This tagger has been chosen because Facebook posts and comments are more Twitter-like. We also use the Stanford POS Tagger (Toutanova et al., 2003) which, unlike the Twitter POS Tagger, has not been tuned for Twitter-like text. These taggers use different tagsets - the ILPOST for **Hi** (Sankaran et al., 2008) and Penn-TreeBank for **En** (Marcus et al., 1993). The output tags are appropriately mapped to the smaller Universal tagset (Petrov et al., 2011).

3.4 Experiments and Results

We conducted three different experiments as follows. In the first experiment, we assume that we know the language identities and normalized/transliterated forms of the words, and only do the POS tagging. This experiment gives us an idea of the accuracy of POS tagging task, if normalization, transliteration and language identification could be done perfectly. We conduct this experiments with two different **En** POS taggers: the Stanford POS tagger which is trained on formal English text (Model 1a) and the Twitter POS tagger (Model 1b). In the next experiment (Model 2), we assume that only the language identity of the words are known, but for Hindi we apply our model to generate the back transliterations. For English, we apply the Twitter POS tagger directly because it can handle unnormalized social media text. The third experiment (Model 3) assumes that nothing is known. So language identifier is first applied, and based on the language detected, we apply the **Hi** translitertaion module, and **Hi** POS tagger, or the **En** tagger. This is the most challenging and realistic setting. Note that the matrix information is not used in any of our experiments, though it could be potentially useful for POS tagging and could be explored in future.

Table 2 gives a summary of the four models along with the POS tagging accuracies (in %). It shows token level as well as chunk leve accuracies (**CA**), i.e., what percentage of chunks have been correctly POS tagged. As can be seen, **Hi** POS tagging has relatively low accuracies than **En** POS tagging at word level for all cases. This is primarily due to the errors of the transliteration module, which in turn, is because the transliteration does not address spelling contractions. This is also reflected in the drop in the accuracies for the case where LI is unknown. The very low **CA** for **En** for model 3 is primarily because some of the **Hi** chunks are incorrectly identified as **En** by the language identification module (see Table 1). However, the gradual drop of token and chunk level accuracies from model 1 to model 3 clearly shows the effect of gradual error accumulation from each of the modules. We observe that Nouns were usually confused most with Verbs and vice versa, while the Adj were mostly confused with Nouns, Pronouns with Determiners, and Adpositions with Conjunctions.

4 Conclusion

This is a work in progress. We have identified normalization and transliteration as two very challenging problems for **En-Hi** CM text. Joint modelling of language identification, normalization, transliteration as well as POS tagging is expected to yield better results. We plan to continue our work in that direction, specifically for conversational text in social media in a multilingual context. CM is a common phenomenon found in all bilingual and multilingual societies. The issue of transliteration exist for most of the South Asian languages as well as many other languages such as Arabic and Greek, which use a non-Roman based script (Gupta et al., 2014). The challenges and issues identified in this study are likely to hold for many other languages as well, which makes this a very important and globally prevalent problem.

References

Peter Auer. 1984. *The Pragmatics of Code-Switching: A Sequential Approach*. Cambridge University Press.

Kalika Bali, Yogarshi Vyas, Jatin Sharma, and Monojit Choudhury. 2014. ''i am borrowing ya mixing?" an analysis of English-Hindi code mixing in Facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching, EMNLP*.

Mónica Stella Cardenas-Claros and Neny Isharyanti. 2009. Code-switching and code-mixing in internet chatting: Between yes, ya, and si a case study. In *The JALT CALL Journal, 5*.

Simon Carter, Wouter Weerkamp, and Manos Tsagkias. 2013. Microblog language identification: Overcoming the limitations of short, unedited and idiomatic text. *Language Resources and Evaluation Journal*, 47:195–215.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *IJDAR*, 10(3-4):157–174.

David Crystal. 2001. *Language and the Internet*. Cambridge University Press.

Brenda Danet and Susan Herring. 2007. *The Multilingual Internet: Language, Culture, and Communication Online*. Oxford University Press., New York.

Spandana Gella, Jatin Sharma, and Kalika Bali. 2013. Query word labeling and back transliteration for indian languages: Shared task system description. In *FIRE Working Notes*.

Kevin Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of ACL*.

Moises Goldszmidt, Marc Najork, and Stelios Paparizos. 2013. Boot-strapping language identifiers for short colloquial postings. In *Machine Learning and Knowledge Discovery in Databases*, volume 8189 of *Lecture Notes in Computer Science*, pages 95–111.

John J. Gumperz. 1964. Hindi-punjabi code-switching in Delhi. In *Proceedings of the Ninth International Congress of Linguistics*. Mouton:The Hague.

John J. Gumperz. 1982. *Discourse Strategies*. Oxford University Press.

Kanika Gupta, Monojit Choudhury, and Kalika Bali. 2012. Mining Hindi-English transliteration pairs from online Hindi lyrics. In *Proceedings of LREC*.

Parth Gupta, Kalika Bali, Rafael E. Banchs, Monojit Choudhury, and Paolo Rosso. 2014. Query expansion for mixed-script information retrieval. In *Proc. of SIGIR*, pages 677–686. ACM Association for Computing Machinery.

Susan Herring, editor. 2003. *Media and Language Change*. Special issue of Journal of Historical Pragmatics 4:1.

Ben King and Steven Abney. 2013. Labeling the languages of words in mixed-language documents using weakly supervised methods. In *Proceedings of NAACL-HLT*, pages 1110–1119.

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

Carol Myers-Scotton. 1993. *Dueling Languages: Grammatical Structure in Code-Switching*. Claredon, Oxford.

Dong Nguyen and A. Seza Dogruoz. 2013. Word level language identification in online multilingual communication. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 857–862.

Olutobi Owoputi, Brendan OConnor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.

Rishiraj Saha Roy, Monojit Choudhury, Prasenjit Majumder, and Komal Agarwal. 2013. Overview and datasets of fire 2013 track on transliterated search. In *FIRE Working Notes*.

Bhaskaran Sankaran, Kalika Bali, Monojit Choudhury, Tanmoy Bhattacharya, Pushpak Bhattacharyya, Girish Nath Jha, S. Rajendran, K. Saravanan, L. Sobha, and K. V. Subbarao. 2008. A common parts-of-speech tagset framework for indian languages. In *Proceedings of LREC*.

Thamar Solorio and Yang Liu. 2008a. Learning to predict code-switching points. In *Proceedings of the Empirical Methods in natural Language Processing*.

Thamar Solorio and Yang Liu. 2008b. Parts-of-speech tagging for English-Spanish code-switched text. In *Proceedings of the Empirical Methods in natural Language Processing*.

V. B. Sowmya, Monojit Choudhury, Kalika Bali, Tirthankar Dasgupta, and Anupam Basu. 2010. Resource creation for training and testing of transliteration systems for indian languages. In *Proceedings of the Language Resource and Evaluation Conference (LREC)*.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*.

Data Driven Grammatical Error Detection in Transcripts of Children's Speech

Eric Morley
CSLU
OHSU
Portland, OR 97239
morleye@gmail.com

Anna Eva Hallin
Department of Communicative Sciences and Disorders
New York University
New York, NY
ae.hallin@nyu.edu

Brian Roark
Google Research
New York, NY 10011
roarkbr@gmail.com

Abstract

We investigate grammatical error detection in spoken language, and present a data-driven method to train a dependency parser to automatically identify and label grammatical errors. This method is agnostic to the label set used, and the only manual annotations needed for training are grammatical error labels. We find that the proposed system is robust to disfluencies, so that a separate stage to elide disfluencies is not required. The proposed system outperforms two baseline systems on two different corpora that use different sets of error tags. It is able to identify utterances with grammatical errors with an F1-score as high as 0.623, as compared to a baseline F1 of 0.350 on the same data.

1 Introduction

Research into automatic grammatical error detection has primarily been motivated by the task of providing feedback to writers, whether they be native speakers of a language or second language learners. Grammatical error detection, however, is also useful in the clinical domain, for example, to assess a child's ability to produce grammatical language. At present, clinicians and researchers into child language must manually identify and classify particular kinds of grammatical errors in transcripts of children's speech if they wish to assess particular aspects of the child's linguistic ability from a sample of spoken language. Such manual annotation, which is called *language sample analysis* in the clinical field, is expensive, hindering its widespread adoption. Manual annotations may also be inconsistent, particularly between different research groups, which may be investigating different phenomena. Automated grammatical error detection has the potential to address both of these issues, being both cheap and consistent.

Aside from performance, there are at least two key requirements for a grammatical error detector to be useful in a clinical setting: 1) it must be able to handle spoken language, and 2) it must be trainable. Clinical data typically consists of transcripts of spoken language, rather than formal written language. As a result, a system must be prepared to handle disfluencies, utterance fragments, and other phenomena that are entirely grammatical in speech, but not in writing. On the other hand, a system designed for transcripts of speech does not need to identify errors specific to written language such as punctuation or spelling mistakes. Furthermore, a system designed for clinical data must be able to handle language produced by children who may have atypical language due to a developmental disorder, and therefore may produce grammatical errors that would be unexpected in written language. A grammatical error detector appropriate for a clinical setting must also be trainable because different groups of clinicians may wish to investigate different phenomena, and will therefore prefer different annotation standards. This is quite different from grammatical error detectors for written language, which may have models for different domains, but which are not typically designed to enable the detection of novel error sets.

We examine two baseline techniques for grammatical error detection, then present a simple data-driven technique to turn a dependency parser into a grammatical error detector. Interestingly, we find that the dependency parser-based approach massively outperforms the baseline systems in terms of identifying ungrammatical utterances. Furthermore, the proposed system is able to identify specific error codes, which the baseline systems cannot do. We find that disfluencies do not degrade performance of the proposed detector, obviating the need (for this task) for explicit disfluency detection. We also analyze the output of our system to see which errors it finds, and which it misses.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 980–989,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Code	Description	Example
[EO]	Overgeneralization errors	He falled [EO] .
[EW]	Other word level errors	He were [EW] looking .
[EU]	Utterance level errors	And they came to stopped .
[OM]	Omitted bound morpheme	He go [OM] .
[OW]	Omitted word	She [OW] running .

Table 1: Error codes proposed in the SALT manual. Note that in SALT annotated transcripts, [OM] and [OW] are actually indicated by '*' followed by the morpheme or word hypothesized to be omitted. When treating codes (other than [EU]) as tags, they are attached to the previous word in the string.

Finally, we evaluate our detector on a second set of data with a different label set and annotation standards. Although our proposed system does not perform as well on the second data set, it still outperforms both baseline systems. One interesting difference between the two data sets, which does appear to impact performance, is that the latter set more strictly follows SALT guidelines (see Section 2.1) to collapse multiple errors into a single label. This yields transcripts with a granularity of labeling somewhat less amenable to automation, to the extent that labels are fewer and can be reliant on non-local context for aggregation.

2 Background

2.1 Systematic Analysis of Language Transcripts (SALT)

The Systematic Analysis of Language Transcripts (SALT) is the de facto standard for clinicians looking to analyze samples of natural language. The SALT manual includes guidelines for transcription, as well as three types of annotations, of which two are relevant here: *maze* annotations, and *error codes*.[1]

Mazes are similar to what is referred to as 'disfluencies' in the speech literature. The SALT manual defines mazes as "filled pauses, false starts, repetitions, reformulations, and interjections" (Miller et al., 2011, p. 6), without defining any of these terms. Partial words, which are included and marked in SALT-annotated transcripts, are also included in mazes. Mazes are delimited by parentheses, and have no internal structure, unlike disfluencies annotated following the Switchboard guidelines (Meteer et al., 1995), which are commonly followed by the speech and language processing communities. An example maze annotation would be: "He (can not) can not get up."

The SALT manual proposes the set of error codes shown (with examples) in Table 1, but research groups may use a subset of these codes, or augment them with additional codes. For example, the SALT-annotated Edmonton Narrative Norms Instrument (ENNI) corpus (Schneider et al., 2006) rarely annotates omitted morphemes ([OM]), instead using the [EW] code. Other SALT-annotated corpora include errors that are not described in the SALT manual. For example the CSLU ADOS corpus (Van Santen et al., 2010) includes the [EX] tag for extraneous words, and the Narrative Story Retell corpus (SALT Software, 2014b) uses the code [EP] to indicate pronominal errors (albeit inconsistently, as many such errors are coded as [EW] in this corpus). We note that the definitions of certain SALT errors, notably [EW] and [EU], are open to interpretation, and that these codes capture a wide variety of errors. For example, some of the errors captured by the [EW] code are: pronominal case and gender errors; verb tense errors; confusing 'a' and 'an'; and using the wrong preposition.

The SALT guidelines specify as a general rule that annotators should not mark utterances with more than two omissions ([OM] or [OW]) and/or word-level errors (ex [EW], [EP]) (SALT Software, 2014a). Instead, annotators are instructed to code such utterances with an utterance-level error ([EU]). How strictly annotators adhere to this rule affects the distribution of errors, reducing the number of word-level errors and increasing the number of utterance-level errors. Following this rule also increases the variety of errors captured by the [EU] code. The annotations in different corpora, including ENNI and NSR, vary in how strictly they follow this rule, even though this is not mentioned in the the published descriptions of

[1] SALT also prescribes annotation of bound morphemes and clitics, for example -ed in past tense verbs. We preprocess all of the transcripts to remove bound morpheme and clitic annotations.

these corpora.

2.2 Grammatical Error Detection

The most visible fruits of research into grammatical error detection are the spellchecking and grammar checking tools commonly included with word processors, for example Microsoft Word's grammar checker. Although developed for handling written language, many of the techniques used to address these tasks could still be applicable to transcripts of speech because many of the same errors can still occur. The earliest grammaticality tools simply performed pattern matching (Macdonald et al., 1982), but this approach is not robust enough to identify many types of errors, and pattern matching systems are not trainable, and therefore cannot be adapted quickly to new label sets. Subsequent efforts to create grammaticality classifiers and detectors leveraged information extracted from parsers (Heidorn et al., 1982) and language models (Atwell, 1987). These systems, however, were developed for formal written English produced by well-educated adults, as opposed to spoken English produced by young children, particularly children with suspected developmental delays.

There have been a few investigations into techniques to automatically identify particular constructions in transcripts of spoken English. Bowden and Fox (2002) proposed a rule-based system to classify many types of errors made by learners of English. Although their system could be used on either transcripts of speech, or on written English, they did not evaluate their system in any way. Caines and Buttery (2010) use a logistic regression model to identify the zero-auxiliary construction (e.g., 'you going home?') with over 96% accuracy. Even though the zero-auxilliary construction is not necessarily ungrammatical, identifying such constructions may be useful as a preprocessing step to a grammaticality classifier. Caines and Buttery also demonstrate that their detector can be integrated into a statistical parser yielding improved performance, although they are vague about the nature of the parse improvement (see Caines and Buttery, 2010, p. 6).

Hassanali and Liu (2011) conducted the first investigation into grammaticality detection and classification in both speech of children, and speech of children with language impairments. They identified 11 types of errors, and compared three types of systems designed to identify the presence of each type of error: 1) rule based systems; 2) decision trees that use rules as features; and 3) naive Bayes classifiers that use a variety of features. They were able to identify all error types well ($F1 > 0.9$ in all cases), and found that in general the statistical systems outperformed the rule based systems. Hassanali and Liu's system was designed for transcripts of spoken language collected from children with impaired language, and is able to detect the set of errors they defined very well. However, it cannot be straightforwardly adapted to novel error sets.

Morley et al. (2013) evaluated how well the detectors proposed by Hassanali and Liu could identify utterances with SALT error codes. They found that a simplified version of one of Hassanali and Liu's detectors was the most effective at identifying utterances with any SALT error codes, although performance was very low (F1=0.18). Their system uses features extracted solely from part of speech tags with the Bernoulli Naive Bayes classifier in Scikit (Pedregosa et al., 2012). Their detector may be adaptable to other annotation standards, but it does not identify which errors are in each utterance; it only identifies which utterances have errors, and which do not.

2.3 Redshift Parser

We perform our experiments with the redshift parser[2], which is an arc-eager transition-based dependency parser. We selected redshift because of its ability to perform disfluency detection and dependency parsing jointly. Honnibal and Johnson (2014) demonstrate that this system achieves state-of-the-art performance on disfluency detection, even compared to single purpose systems such as the one proposed by Qian and Liu (2013). Rasooli and Tetreault (2014) have developed a system that performs disfluency detection and dependency parsing jointly, and with comparable performance to redshift, but it is not publicly available as of yet.

Redshift uses an averaged perceptron learner, and implements several feature sets. The first feature set, which we will refer to as ZHANG is the one proposed by Zhang and Nivre (2011). It includes 73 templates that capture various aspects of: the word at the top of the stack, along with its

[2]Redshift is available at `https://github.com/syllog1sm/redshift`. We use the version in the `experiment` branch from May 15, 2014.

leftmost and rightmost children, parent and grandparent; and the word on the buffer, along with its leftmost children; and the second and third words on the buffer. Redshift also includes features extracted from the Brown clustering algorithm (Brown et al., 1992). Finally, redshift includes features that are designed to help identify disfluencies; these capture rough copies, exact copies, and whether neighboring words were marked as disfluent. We will refer to the feature set containing all of the features implemented in redshift as FULL. We refer the reader to Honnibal and Johnson (2014) for more details.

3 Data, Preprocessing, and Evaluation

Our investigation into using a dependency parser to identify and label grammatical errors requires training data with two types of annotations: dependency labels, and grammatical error labels. We are not aware of any corpora of speech with both of these annotations. Therefore, we use two different sets of training data: the Switchboard corpus, which contains syntactic parses; and SALT annotated corpora, which have grammatical error annotations.

3.1 Switchboard

The Switchboard treebank (Godfrey et al., 1992) is a corpus of transcribed conversations that have been manually parsed. These parses include EDITED nodes, which span disfluencies. We preprocess the Switchboard treebank by removing all partial words as well as all words dominated by EDITED nodes, and converting all words to lowercase. We then convert the phrase-structure trees to dependencies using the Stanford dependency converter (De Marneffe et al., 2006) with the basic dependency scheme, which produces dependencies that are strictly projective.

3.2 SALT Annotated Corpora

We perform two sets of experiments on the two SALT-annotated corpora described in Table 2. We carry out the first set of experiments on on the Edmonton Narrative Norms Instrument (ENNI) corpus, which contains 377 transcripts collected from children between the ages of 3 years 11 months and 10 years old. The children all lived in Edmonton, Alberta, Canada, were typically developing, and were native speakers of English.

After exploring various system configurations, we evaluate how well our method works when it is applied to another corpus with different annotation standards. Specifically, we train and test our technique on the Narrative Story Retell (NSR) corpus (SALT Software, 2014b), which contains 496 transcripts collected from typically developing children living in Wisconsin and California who were between the ages of 4 years 4 months and 12 years 8 months old. The ENNI and NSR corpora were annotated by two different research groups, and as Table 2 illustrates, they contain a different distribution of errors. First, ENNI uses the [EW] (other word-level error) tag to code both overgeneralization errors instead of [EO], and omitted morphemes instead of [OM]. The [EU] code is also far more frequent in ENNI than NSR. Finally, the NSR corpus includes an error code that does not appear in the ENNI corpus: [EP], which indicates a pronominal error, for example using the wrong person or case. [EP], however, is rarely used.

	ENNI		NSR	
	Words	Utts	Words	Utts
Train	360,912	44,915	103,810	11,869
Dev.	45,504	5,614	12,860	1,483
Test	44,996	5,615	12,982	1,485
% with error		13.2		14.3

(a) Word and utterance counts

	ENNI	NSR
[EP]	0	20
[EO]	0	495
[EW]	4,916	1,506
[EU]	3,332	568
[OM]	10	297
[OW]	766	569
Total	9,024	3,455

(b) Error code counts

Table 2: Summary of ENNI and NSR Corpora. There can be multiple errors per utterance. Word counts include mazes.

We preprocess the ENNI and NSR corpora to reconstruct surface forms from bound morpheme annotations (ex. 'go/3S' becomes 'goes'), partial words, and non-speech sounds. We also either excise manually identified mazes or remove maze annotations, depending upon the experiment.

3.3 Evaluation

Evaluating system performance in tagging tasks on manually annotated data is typically straight-

Evaluation Level:	ERROR			UTTERANCE
	Individual error codes			Has error?
Gold error codes:	[EW]	[EW]		Yes
Predicted error codes:	[EW]		[OW]	Yes
Evaluation:	TP	FN	FP	TP

Figure 1: Illustration of UTTERANCE and ERROR level evaluation
TP = true positive; FP = false positive; FN = false negative

forward: we simply compare system output to the gold standard. Such evaluation assumes that the best system is the one that most faithfully reproduces the gold standard. This is not necessarily the case with applying SALT error codes for three reasons, and each of these reasons suggests a different form of evaluation.

First, automatically detecting SALT error codes is an important task because it can aid clinical investigations. As Morley et al. (2013) illustrated, even extremely coarse features derived from SALT annotations, for example a binary feature for each utterance indicating the presence of any error codes, can be of immense utility for identifying language impairments. Therefore, we will evaluate our system as a binary tagger: each utterance, both in the manually annotated data and system output either contains an error code, or it does not. We will label this form of evaluation as UTTERANCE level.

Second, clinicians are not only interested in how many utterances have an error, but also which particular errors appear in which utterances. To address this issue, we will compute precision, recall, and F1 score from the counts of each error code in each utterance. We will label this form of evaluation as ERROR level. Figure 1 illustrates both UTTERANCE and ERROR level evaluation. Note that the utterance level error code [EU] is only allowed to appear once per utterance. As a result, we will ignore any predicted [EU] codes beyond the first.

Third, the quality of the SALT annotations themselves is unknown, and therefore evaluation in which we treat the manually annotated data as a gold standard may not yield informative metrics. Morley et al. (2014) found that there are likely inconsistencies in maze annotations both within and across corpora. In light of that finding, it is possible that error code annotations are somewhat inconsistent as well. Furthermore, our approach has a critical difference from manual annotation: we perform classification one utterance at a time, while manual annotators have access to the context of an utterance. Therefore certain types of errors, for example using a pronoun of the wrong gender, or responding ungrammatically to a question (ex. 'What are you doing?' 'Eat.') will appear grammatical to our system, but not to a human annotator. We address both of these issues with an in-depth analysis of the output of one of our systems, which includes manually re-coding utterances out of context.

4 Detecting Errors in ENNI

4.1 Baselines

We evaluate two existing systems to see how effectively they can identify utterances with SALT error codes: 1) Microsoft Word 2010's grammar check, and 2) the simplified version of Hassanali and Liu's grammaticality detector (2011) proposed by Morley et al. (2013) (mentioned in Section 2.2). We configured Microsoft Word 2010's grammar check to look for the following classes of errors: negation, noun phrases, subject-verb agreement, and verb phrases (see `http://bit.ly/1kphUHa`). Most error classes in grammar check are not relevant for transcribed speech, for example capitalization errors or confusing *it's* and *its*; we selected classes of errors that would typically be indicated by SALT error codes.

Note that these baseline systems can only give us an indication of whether there is an error in the utterance or not; they do not provide the specific error tags that mimic the SALT guidelines. Hence we evaluate just the UTTERANCE level performance of the baseline systems on the ENNI development and test sets. These results are given in the top two rows of each section of Table 3. We apply these systems to utterances in two conditions: with mazes (i.e., disfluencies) excised; and with unannotated mazes left in the utterances. As can be seen in Table 3, the performance Microsoft Word's grammar checker degrades severely when

(a)

Him [EW] (can not) can not get up .

(b)

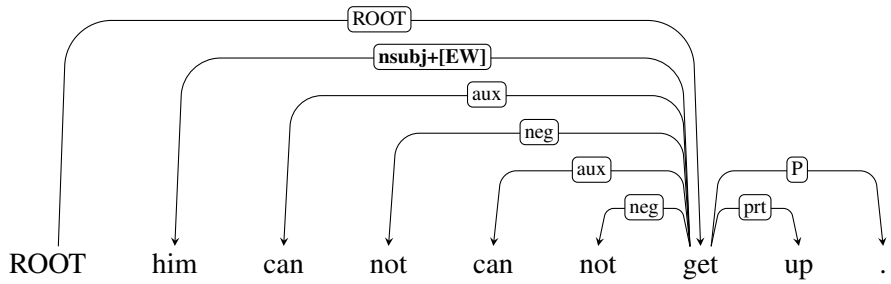


Figure 2: (a) SALT annotated utterance; mazes indicated by parentheses; (b) Dependency parse of same utterance parsed with a grammar trained on the Switchboard corpus and augmented dependency labels. We use a corpus of parses with augmented labels to train our grammaticality detector.

mazes are not excised, but this is not the case for the Morley et al. (2013) detector.

4.2 Proposed System

Using the ENNI corpus, we now explore various configurations of a system for grammatical error code detection. All of our systems use redshift to learn grammars and to parse. First, we train an initial grammar G_0 on the Switchboard treebank (Godfrey et al., 1992) (preprocessed as described in Section 3.1). Redshift learns a model for part of speech tagging concurrently with G_0. We use G_0 to parse the training portion of the ENNI corpus. Then, using the SALT annotations, we append error codes to the dependency arc labels in the parsed ENNI corpus, assigning each error code to the word it follows in the SALT annotated data. Figure 2 shows a SALT annotated utterance, as well as its dependency parse augmented with error codes. Finally, we train a grammar G_{Err} on the parse of the ENNI training fold that includes the augmented arc labels. We can now use G_{Err} to automatically apply SALT error codes: they are simply encoded in the dependency labels. We also apply the [EW] label to any word that is in a list of overgeneralization errors[3].

We modify three variables in our initial trials on the ENNI development set. First, we change the proportion of utterances in the training data that contain an error by removing utterances.[4] Doing so allows us to alter the operating point of our system in terms of precision and recall. Second, we again train and test on two versions of the ENNI corpus: one which has had mazes excised, and the other which has them present (but not annotated). Third, we evaluate two feature sets: ZHANG and FULL.

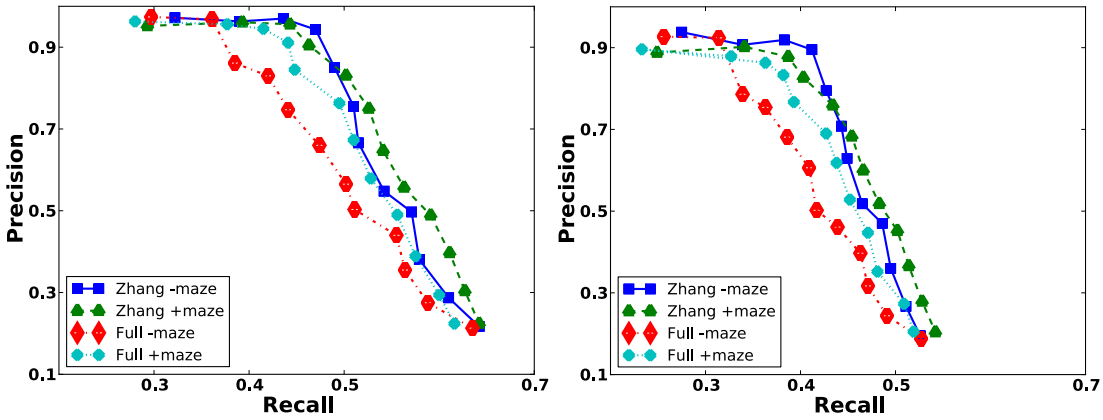
The plots in Figure 3 show how the performances of our systems at different operating points vary, while Table 3 shows the performance of our best system configurations on the ENNI development and test sets. Surprisingly, we see that neither the choice of feature set, nor the presence of mazes has much of an effect on system performance. This is in strong contrast to Microsoft Word's grammar check, which is minimally effective when mazes are included in the data. The Morley et al. (2013) system is robust to mazes, but still performs substantially worse than our proposed system.

4.3 Error Analysis

We now examine the errors produced by our best performing system for data in which mazes are present. As shown in Table 3, when we apply our system to ENNI-development, the UTTERANCE P/R/F1 is 0.831 / 0.502 / 0.626 and the ERROR P/R/F1is 0.759 / 0.434 / 0.552. This system's performance detecting specific error codes is shown in Table 4. We see that the recall of [EU] errors is quite low compared with the recall for [EW] and [OW] errors. This is not surprising, as human annotators may need to leverage the context of an utterance to identify [EU] errors, while our system makes predictions for each utterance in isolation.

[3]The list of overgeneralization errors was generously provided by Kyle Gorman

[4]Of course, we never modify the development or test data.



(a) UTTERANCE level evaluation

(b) ERROR level evaluation

Figure 3: SALT error code detection performance at various operating points on ENNI development set

System	Eval type	Mazes Excised P	R	F1	Mazes Present P	R	F1
Development							
MS Word	UTT	0.843	0.245	0.380	0.127	0.063	0.084
Morley et al. (2013)	UTT	0.407	0.349	0.376	0.343	0.321	0.332
Current paper	UTT	0.943	0.470	0.627	0.831	0.502	0.626
	ERR	0.895	0.412	0.564	0.759	0.434	0.552
Test							
MS Word	UTT	0.824	0.209	0.334	0.513	0.219	0.307
Morley et al. (2013)	UTT	0.375	0.328	0.350	0.349	0.252	0.293
Current Paper	UTT	0.909	0.474	0.623	0.809	0.501	0.618
	ERR	0.682	0.338	0.452	0.608	0.360	0.452

Table 3: Baseline and current paper systems' performance on ENNI. Evaluation is at the UTTERANCE (UTT) level except for the current paper's system, which also presents evaluation at the ERROR (ERR) level.

Error Code	P	R	F1
EU	0.639	0.193	0.297
EW	0.832	0.582	0.685
OW	0.680	0.548	0.607

Table 4: ERROR level detection performance for each code (system trained on ENNI; 30% error utterances; ZHANG feature set; with mazes)

We randomly sampled 200 utterances from the development set that have a manually annotated error, are predicted by our system to have an error, or both. A speech-language pathologist who has extensive experience with using SALT for research purposes in both clinical and typically developing populations annotated the errors in each utterance. She annotated each utterance in isolation so as to ignore contextual errors. We compare our annotations to the original annotations, and system performance using our annotations and the original annotations as different gold standards. The results of this comparison are shown in Table 5.

Comparing our manual annotations to the original annotations, we notice some disagreements. We suspect there are two reasons for this. First, unlike the original annotators, we annotate these utterances out of context. This may explain why we identify far fewer utterance level error [EU] codes than the original annotators (20 compared with 67). Second, we may be using different criteria for each error code than the original annotators. This is an inevitable issue, as the SALT guidelines do not provide detailed definitions of the error codes, nor do individual groups of annotators. To illustrate, the "coding notes" section of

Tag	Gold	Gold Count	Disagreement	P	R	F1
[EU]	Original	67	52	0.500	0.149	0.230
	Revised	20		0.450	0.333	0.383
[EW]	Original	137	27	0.859	0.533	0.658
	Revised	126		0.800	0.540	0.645
[OW]	Original	16	13	0.667	0.275	0.480
	Revised	15		0.444	0.267	0.333

Table 5: System performance using ERROR level evaluation on 200 utterances selected from ENNI-dev using original and revised annotations as gold standard

	UTTERANCE level			ERROR level		
System	P	R	F1	P	R	F1
ENNI-trained	0.310	0.124	0.178	0.157	0.057	0.084
NSR-trained	0.243	0.249	**0.277**	0.150	0.195	0.170
MS Word	0.561	0.171	0.261	–	–	–
Morley et al. (2013)	0.250	0.281	0.264	–	–	–
NSR ∪ MS Word	0.291	0.447	0.353	–	–	–
NSR ∪ Morley et al. (2013)	0.297	0.387	0.336	–	–	–
All 3	0.330	0.498	**0.397**	–	–	–

Table 6: Error detection performance on NSR-development, mazes included

the description of the ENNI corpus[5] only lists the error codes that were used consistently, but does not describe how to apply them. These findings illustrate the importance of having a rapidly trainable error code detector: research groups will be interested in different phenomena, and therefore will likely have different annotation standards.

5 Detecting Errors in NSR

We apply our system directly to the NSR corpus with mazes included. We use the same parameters set on the ENNI corpus in Section 4.2. We apply the model trained on ENNI to NSR, but find that it does not perform very well as illustrated in Table 6. These results further underscore the need for a trainable error code detector in this domain, as opposed to the static error detectors that are more common in the grammatical error detection literature.

We see in Table 6 that retraining our model on NSR data improves performance substantially (UTTERANCE F1 improves from 0.178 to 0.277), but not to the level we observed on the ENNI corpus. The Morley et al. (2013) system also performs worse when trained and tested on NSR, as compared with ENNI. When mazes are included, the performance of Microsoft Word's grammar check is higher on NSR than on ENNI (F1=0.261 vs 0.084), but it it still yields the lowest performance of the three systems. We find that combining our proposed system with either or both of the baseline systems further improves performance.

The NSR corpus differs from ENNI in several ways: it is smaller, contains fewer errors, and uses a different set of tags with a different distribution from the ENNI corpus, as shown in Table 2. We found that the smaller amount of training data is not the only reason for the degradation in performance; we trained a model for ENNI with a set of training data that is the same size as the one for NSR, but did not observe a major drop in performance. We found that UTTERANCE F1 drops from 0.626 to 0.581, and ERROR F1 goes from 0.552 to 0.380, not nearly the magnitude drop in accuracy observed for NSR.

We believe that a major reason for why our system performs worse on NSR than ENNI may be that the ENNI annotations adhere less strictly to certain SALT recommendations than do the ones in NSR. The SALT guidelines suggest that utterances with two or more word-level [EW] and/or omitted word [OW] errors should only be tagged with an utterance-level [EU] error (SALT Software, 2014a). ENNI, however, has many utter-

[5] http://www.saltsoftware.com/salt/databases/ENNIRDBDoc.pdf

ances with multiple [EW] and [OW] error codes, along with utterances containing all three error codes. NSR has very few utterances with [EU] and other codes, or multiple [EW] and [OW] codes. The finer grained annotations in ENNI may simply be easier to learn.

6 Conclusion and Future Directions

We have proposed a very simple method to rapidly train a grammatical error detector and classifier. Our proposed system only requires training data with error code annotations, and is agnostic as to the nature of the specific error codes. Furthermore, our system's performance does not appear to be affected by disfluencies, which reduces the burden required to produce training data.

There are several key areas we plan to investigate in the future. First, we would like to explore different update functions for the parser; the predicted error codes are a byproduct of parsing, but we do not care what the parse itself looks like. At present, the parser is updated whenever it produces a parse that diverges from the gold standard. It may be better to update only when the error codes predicted for an utterance differ from the gold standard. Second, we hope to explore features that could be useful for identifying grammatical errors in multiple data sets. Finally, we plan to investigate why our system performed so much better on ENNI than on NSR.

Acknowledgments

We would like to thank the following people for valuable input into this study: Joel Tetreault, Jan van Santen, Emily Prud'hommeaux, Kyle Gorman, Steven Bedrick, Alison Presmanes Hill and others in the CSLU Autism research group at OHSU. This material is based upon work supported by the National Institute on Deafness and Other Communication Disorders of the National Institutes of Health under award number R21DC010033. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

Eric Atwell. 1987. How to detect grammatical errors in a text without parsing it. In Bente Maegaard, editor, *EACL*, pages 38–45, Copenhagen, Denmark, April. The Association for Computational Linguistics.

Mari I Bowden and Richard K Fox. 2002. A diagnostic approach to the detection of syntactic errors in english for non-native speakers. *The University of Texas–Pan American Department of Computer Science Technical Report*.

Peter F. Brown, Vincent J. Della Pietra, Peter V. de Souza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Andrew Caines and Paula Buttery. 2010. You talking to me?: A predictive model for zero auxiliary constructions. In *Proceedings of the 2010 Workshop on NLP and Linguistics: Finding the Common Ground*, pages 43–51.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520.

Khairun-nisa Hassanali and Yang Liu. 2011. Measuring language development in early childhood education: a case study of grammar checking in child language transcripts. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 87–95.

George E. Heidorn, Karen Jensen, Lance A. Miller, Roy J. Byrd, and Martin S Chodorow. 1982. The EPISTLE text-critiquing system. *IBM Systems Journal*, 21(3):305–326.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *TACL*, 2:131–142.

Nina H Macdonald, Lawrence T Frase, Patricia S Gingrich, and Stacey A Keenan. 1982. The writer's workbench: Computer aids for text analysis. *Educational psychologist*, 17(3):172–179.

Marie W Meteer, Ann A Taylor, Robert MacIntyre, and Rukmini Iyer. 1995. *Dysfluency annotation stylebook for the switchboard corpus*. University of Pennsylvania.

Jon F Miller, Karen Andriacchi, and Ann Nockerts. 2011. *Assessing language production using SALT software: A clinician's guide to language sample analysis*. SALT Software, LLC.

Eric Morley, Brian Roark, and Jan van Santen. 2013. The utility of manual and automatic linguistic error codes for identifying neurodevelopmental disorders. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–10, Atlanta, Georgia, June. Association for Computational Linguistics.

Eric Morley, Anna Eva Hallin, and Brian Roark. 2014. Challenges in automating maze detection. In *Proceedings of the First Workshop on Computational Linguistics and Clinical Psychology*, pages 69–77, Baltimore, Maryland, June.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2012. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490.

Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In Lucy Vanderwende, Hal Daumé III, and Katrin Kirchhoff, editors, *HLT-NAACL*, pages 820–825, Atlanta, Georgia, USA, June. The Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2014. Non-monotonic parsing of fluent umm i mean disfluent sentences. In Gosse Bouma and Yannick Parmentier, editors, *EACL*, pages 48–53, Gothenburg, Sweden, April. The Association for Computational Linguistics.

LLC SALT Software. 2014a. Course 1306: Transcription - Conventions Part 3. `http://www.saltsoftware.com/onlinetraining/section-page?OnlineTrainingCourseSectionPageId=76`. [Online; accessed 29-May-2104].

LLC SALT Software. 2014b. Narrative Story Retell Database. `http://www.saltsoftware.com/salt/databases/NarStoryRetellRDBDoc.pdf`. [Online; accessed 29-May-2104].

Phyllis Schneider, Denyse Hayward, and Rita Vis Dubé. 2006. Storytelling from pictures using the edmonton narrative norms instrument. *Journal of Speech Language Pathology and Audiology*, 30(4):224.

Jan PH Van Santen, Emily T Prud'hommeaux, Lois M Black, and Margaret Mitchell. 2010. Computational prosodic markers for autism. *Autism*, 14(3):215–236.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *ACL (Short Papers)*, pages 188–193, Portland, Oregon, USA, June. The Association for Computational Linguistics.

A* CCG Parsing with a Supertag-factored Model

Mike Lewis
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
mike.lewis@ed.ac.uk

Mark Steedman
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
steedman@inf.ed.ac.uk

Abstract

We introduce a new CCG parsing model which is factored on lexical category assignments. Parsing is then simply a deterministic search for the most probable category sequence that supports a CCG derivation. The parser is extremely simple, with a tiny feature set, no POS tagger, and no statistical model of the derivation or dependencies. Formulating the model in this way allows a highly effective heuristic for A* parsing, which makes parsing extremely fast. Compared to the standard C&C CCG parser, our model is more accurate out-of-domain, is four times faster, has higher coverage, and is greatly simplified. We also show that using our parser improves the performance of a state-of-the-art question answering system.

1 Introduction

CCG is a strongly lexicalized grammatical formalism, in which the vast majority of the decisions made during interpretation involve choosing the correct definitions of words. We explore the effect of modelling this explicitly in a parser, by only using a probabilistic model of lexical categories (based on a local context window), rather than modelling the derivation or dependencies.

Existing state-of-the-art CCG parsers use complex pipelines of POS-tagging, supertagging and parsing—each with its own feature sets and parameters (and sources of error)—together with further parameters governing their integration (Clark and Curran, 2007). We show that much simpler models can achieve high performance. Our model predicts lexical categories based on a tiny feature set of word embeddings, capitalization, and 2-character suffixes—with no parsing model beyond a small set of CCG combinators, and no POS-tagger. Simpler models are easier to implement, replicate and extend.

Another goal of our model is to parse CCG optimally and efficiently, without using excessive pruning. CCG's large set of lexical categories, and generalized notion of constituency, mean that sentences can have a huge number of potential parses. Fast existing CCG parsers rely on aggressive pruning—for example, the C&C parser uses a supertagger to dramatically cut the search space considered by the parser. Even the loosest beam setting for their supertagger discards the correct parse for 20% of sentences. The structure of our model allows us to introduce a simple but powerful heuristic for A* parsing, meaning it can parse almost 50 sentences per second exactly, with no beam-search or pruning. Adding very mild pruning increases the speed to 186 sentences per second with minimal loss of accuracy.

Our approach faces two obvious challenges. Firstly, categories are assigned based on a local window, which may not contain the necessary context for resolving some attachment decisions. For example, in *I saw a squirrel 2 weeks ago with a nut*, the model cannot make an informed decision on whether to assign *with* an adverbial or adnominal preposition category, as the crucial words *saw* and *squirrel* fall outside the local context window. Secondly, even if the supertagger makes all lexical category decisions correctly, then the parser can still make erroneous decisions. One example is in coordination-scope ambiguities, such as *clever boys and girls*, where the two interpretations use the same assignment of categories.

We hypothesise that such decisions are relatively rare, and are challenging for any parsing model, so a weak model is unlikely to result in substantially lower accuracy. Our implementation of this model[1], which we call EASYCCG, has high

[1]Available from https://github.com/mikelewis0/easyccg

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 990–1000,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

accuracy—suggesting that most parsing decisions can be made accurately based on a local context window.

Of course, there are many parsing decisions that can only be made accurately with more complex models. However, exploring the power and limitations of simpler models may help focus future research on the more challenging cases.

2 Background

2.1 Combinatory Categorial Grammar

CCG (Steedman, 2000) is a strongly lexicalized grammatical formalism. Words have categories representing their syntactic role, which are either atomic, or functions from one category to another.

Phrase-structure grammars have a relatively small number of lexical categories types (e.g. POS-tags), and a large set of rules used to build a syntactic analysis of a complete sentence (e.g. an adjective and noun can combine into a noun). In contrast, CCG parsing has many lexical category types (we use 425), but a small set of combinatory rule types (we use 10 binary and 13 unary rule schemata). This means that, aside from the lexicon, the grammar is small enough to be hand-coded—which allows us, in this paper, to confine the entire statistical model to the lexicon.

CCG's generalized notion of constituency means that many derivations are possible for a given a set of lexical categories. However, most of these derivations will be semantically equivalent—for example, deriving the same dependency structures—in which case the actual choice of derivation is unimportant. Such ambiguity is often called *spurious*.

2.2 Existing CCG Parsing Models

The seminal C&C parser is by far the most popular choice of CCG parser (Clark and Curran, 2007). It showed that it was possible to parse to an expressive linguistic formalism with high speed and accuracy. The performance of the parser has enabled large-scale logic-based distributional research (Harrington, 2010; Lewis and Steedman, 2013a; Lewis and Steedman, 2013b; Reddy et al., 2014), and it is a key component of Boxer (Bos, 2008).

The C&C parser uses CKY chart parsing, with a log-linear model to rank parses. The vast number of possible parses means that computing the complete chart is impractical. To resolve this problem, a *supertagger* is first run over the sentence to prune the set of lexical categories considered by the parser for each word. The initial beam outputs an average of just 1.2 categories per word, rather than the 425 possible categories—making the standard CKY parsing algorithm very efficient. If the parser fails to find any analysis of the complete sentence with this set of supertags, the supertagger re-analyses the sentence with a more relaxed beam (*adaptive supertagging*).

2.3 A* Parsing

Klein and Manning (2003a) introduce A* parsing for PCFGs. The parser maintains a chart and an agenda, which is a priority queue of items to add to the chart. The agenda is sorted based on the items' inside probability, and a heuristic upper-bound on the outside probability—to give an upper bound on the probability of the complete parse. The chart is then expanded in best-first order, until a complete parse for the sentence is found.

Klein and Manning calculate an upper bound on the outside probability of a span based on a summary of the context. For example, the summary for the SX heuristic is the category of the span, and the number of words in the sentence before and after the span. The value of the heuristic is the probability of the best possible sentence meeting these restrictions. These probabilities are pre-computed for every non-terminal symbol and for every possible number of preceding and succeeding words, leading to large look-up tables.

Auli and Lopez (2011b) find that A* CCG parsing with this heuristic is very slow. However, they achieve a modest 15% speed improvement over CKY when A* is combined with adaptive supertagging. One reason is that the heuristic estimate is rather coarse, as it deals with the best possible outside context, rather than the actual sentence. We introduce a new heuristic which gives a tighter upper bound on the outside probability.

3 Model

3.1 Lexical Category Model

As input, our parser takes a distribution over all CCG lexical categories for each word in the sentence. These distributions are assigned using Lewis and Steedman (2014)'s semi-supervised supertagging model. The supertagger is a unigram log-linear classifier that uses features of the ± 3 word context window surrounding a word. The

key feature is word embeddings, initialized with the 50-dimensional embeddings trained in Turian et al. (2010), and fine-tuned during supervised training. The model also uses 2-character suffixes and capitalization features.

The use of word embeddings, which are trained on a large unlabelled corpus, allows the supertagger to generalize well to words not present in the labelled data. It does not use a POS-tagger, which avoids problems caused by POS-tagging errors.

Our methods could be applied to any supertagging model, but we find empirically that this model gives higher performance than the C&C supertagger.

3.2 Parsing Model

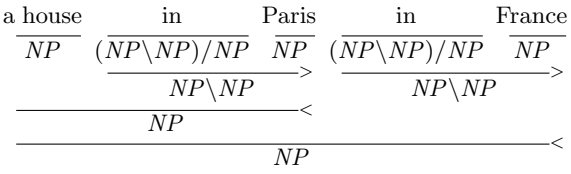
Let a CCG parse y of a sentence S be a list of lexical categories $c_1 \dots c_n$ and a derivation. If we assume all derivations licensed by our grammar are equally likely, and that lexical category assignments are conditionally independent given the sentence, we can compute the optimal parse $\hat{y}$ as:

$\hat{y} = argmax_y \prod_{i=1}^{n} p(c_i|S)$

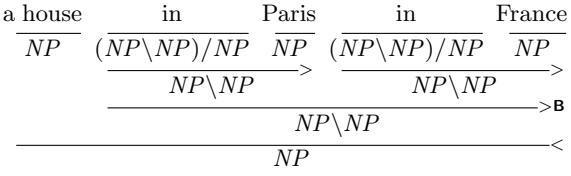
As discussed in Section 2.1, many derivations are possible given a sequence of lexical categories, some of which may be semantically distinct. However, our model will assign all of these an equal score, as they use the same sequence of lexical categories. Therefore we extend our model with a simple deterministic heuristic for ranking parses that use the same lexical categories. Given a set of derivations with equal probability, we output the one maximizing the sums of the length of all arcs in the corresponding dependency tree.

The effect of this heuristic is to prefer non-local attachments in cases of ambiguity, which we found worked better on development data than favouring local attachments. In cases of spurious ambiguity, all parses will have the same value of this heuristic, so one is chosen arbitrarily. For example, one of the parses in Figures 1a and 1b would be selected over the parse in Figure 1c.

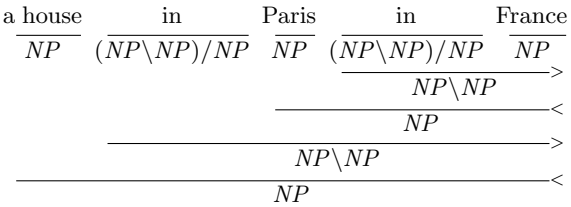
Of course, we could use any function of the parses in place of this heuristic, for example a head-dependency model. However, one aim of this paper is to demonstrate that an extremely simple parsing model can achieve high performance, so we leave more sophisticated alternatives to future work.



(a) A standard derivation of *a house in Paris in France*, with a dependency from *in France* to *house*



(b) A derivation of *a house in Paris in France*, which is spuriously equivalent to Figure 1a. A composition combinator is used to compose the predicates *in Paris* and *in France*, creating a constituent which creates dependencies to its argument from both *in Paris* and *in France*.



(c) A derivation of *a house in Paris in France*, which yields different dependencies to Figures 1a and 1b: here, there is a dependency from *in France* to *Paris*, not *house*.

Figure 1: Three CCG parses of *a house in Paris in France*, given the same set of supertags. The first two are spuriously equivalent, but the third is semantically distinct.

3.3 A* Search

For parsing, we use an A* search for the most-probable complete CCG derivation of a sentence. A key advantage of A* parsing over CKY parsing is that it does not require us to prune the search space first with a supertagger, allowing the parser to consider the complete distribution of 425 categories for each word (in contrast to an average of 3.57 categories per word considered by the C&C parser's most relaxed beam). This is possible because A* only searches for the Viterbi parse of a sentence, rather than building a complete chart with every possible category per word (another alternative, used by Hockenmaier (2003), is to use a highly aggressive beam search in the parser).

In A* parsing, items on the agenda are sorted by their *cost*; the product of their inside probability and an upper bound on their outside probability.

For a span $w_i \dots w_j$ with lexical categories $c_i \dots c_j$ in a sentence $S = w_1 \dots w_n$, the inside probability is simply: $\prod_{k=i}^{j} p(c_k|S)$

The factorization of our model lets us give the following upper-bound on the outside probability:

$$h(w_i \dots w_j) = \prod_{k=1}^{k<i} max_{c_k} p(c_k|S) \times \prod_{k=j+1}^{k \leq n} max_{c_k} p(c_k|S)$$

This heuristic assumes that all words outside the span will take their highest-probability supertag. Because the model is factored on lexical categories, this estimate is clearly an upper bound. As supertagging is over 90% accurate, the upper bound will often be exact, and in Section 4.3 we show empirically that it is extremely efficient. The values of the heuristic can be computed once for each sentence and cached.

To implement the preference for non-local attachment described in Section 3.2, if two agenda items have the same cost, the one with the longer dependencies is preferred.

Intuitively, the parser first attempts to find a parse for the sentence using the 1-best category for each word, by building as complete a chart as possible. If it fails to find a parse for the complete sentence, it adds one more supertag to the chart (choosing the most probable tag not already in the chart), and tries again. This strategy allows the parser to consider an unbounded number of categories for each word, as it does not build a complete chart with all supertags.

3.4 Grammar

Here, we describe the set of combinators and unary rules in the EASYCCG grammar. Because we do not have any probabilistic model of the derivation, all rules can apply with equal probability. This means that some care needs to be taken in designing the grammar to ensure that all the rules are generally applicable. We also try to limit spurious ambiguity, and build derivations which are compatible with the C&C parser's scripts for extracting dependencies (for evaluation). We describe the grammar in detail, to ensure replicability.

Our parser uses the following binary combinators from Steedman (2012): *forward application*, *backward application*, *forward composition*, *backward crossed composition*, *generalized forward composition*, *generalized backward crossed composition*. These combinators are posited to be linguistically universal. The generalized rules are generalized to degree 2. Following Steedman (2000) and Clark and Curran (2007), backward composition is blocked where the argument of the right-hand category is an N or NP. The unhelpful *[nb]* feature is ignored.

Initial	Result	Usage
N	NP	Bare noun phrases
NP	$S/(S\backslash NP)$	Type raising
NP	$(S\backslash NP)/((S\backslash NP)/NP)$	
PP	$(S\backslash NP)/((S\backslash NP)/PP)$	
$S_{pss}\backslash NP$	$NP\backslash NP$	Reduced relative clauses
$S_{ng}\backslash NP$	$NP\backslash NP$	
$S_{adj}\backslash NP$	$NP\backslash NP$	
$S_{to}\backslash NP$	$NP\backslash NP$	
$S_{to}\backslash NP$	$N\backslash N$	
S_{dcl}/NP	$NP\backslash NP$	
$S_{pss}\backslash NP$	S/S	VP Sentence Modifiers
$S_{ng}\backslash NP$	S/S	
$S_{to}\backslash NP$	S/S	

Table 1: Set of unary rules used by the parser.

As in the C&C parser, we add a special *Conjunction* rule:

$$\frac{Y \qquad X}{X\backslash X}>$$

Where $Y \in \{conj, comma, semicolon\}$. We block conjunctions where the right-hand category is type-raised, punctuation, N, or $NP\backslash NP$. This rule (and the restrictions) could be removed by changing CCGBank to analyse conjunctions with $(X\backslash X)/X$ categories.

We also add syntagmatic rules for removing any punctuation to the right, and for removing open-brackets and open-quotes to the left

The grammar also contains 13 unary rules, listed in Table 1. These rules were chosen based on their frequency in the training data, and their clear semantic interpretations.

Following Clark and Curran (2007), we also add a (switchable) constraint that only category combinations that have combined in the training data may combine in the test data. We found that this was necessary for evaluation, as the C&C conversion tool for extracting predicate-argument dependencies had relatively low coverage on the CCG derivations produced by our parser. While this restriction is theoretically inelegant, we found it did increase parsing speed without lowering lexi-

cal category accuracy.

We also use Eisner Normal Form Constraints (Eisner, 1996), and Hockenmaier and Bisk's (2010) Constraint 5, which automatically rule out certain spuriously equivalent derivations, improving parsing speed.

We add a hard constraint that the root category of the sentence must be a declarative sentence, a question, or a noun-phrase.

This grammar is smaller and cleaner than that used by the C&C parser, which uses 32 unary rules (some of which are semantically dubious, such as $S[dcl] \rightarrow NP\backslash NP$), and non-standard binary combinators such as merging two NPs into an NP. The C&C parser also has a large number of special case rules for handling punctuation. Our smaller grammar reduces the grammar constant, eases implementation, and simplifies the job of building downstream semantic parsers such as those of Bos (2008) or Lewis and Steedman (2013a) (which must implement semantic analogs of each syntactic rule).

3.5 Extracting Dependency Structures

The parsing model defined in Section 3.2 requires us to compute unlabelled dependency trees from CCG derivations (to prefer non-local attachments). It is simple to extract an unlabelled dependency tree from a CCG parse, by defining one argument of each binary rule instantiation to be the head. For forward application and (generalized) forward composition, we define the head to be the left argument, unless the left argument is an endocentric head-passing modifier category X/X. We do the inverse for the corresponding 'backward' combinators. For punctuation rules, the head is the argument which is not punctuation, and the head of a *Conjunction* rule is the right-hand argument.

The standard CCG parsing evaluation uses a different concept of dependencies, corresponding to the predicate-argument structure defined by CCGBank. These dependencies capture a deeper information—for example by assigning both *boy* and *girl* as subjects of *talk* in *a boy and a girl talked*. We extract these dependencies using the `generate` program supplied with the C&C parser.

3.6 Pruning

Our parsing model is able to efficiently and optimally search for the best parse. However, we found that over 80% of the run-time of our pipeline was spent during supertagging. Naively, the log-linear model needs to output a probability for each of the 425 categories. This is expensive both in terms of the number of dot products required, and the cost of building the initial priority-queue for the A* parsing agenda. It is also largely unnecessary—for example, periods at the end of sentences always have the same category, but our supertagger calculates a distribution over all possible categories.

Note that the motivation for introducing pruning here is fundamentally different from for the C&C pipeline. The C&C supertagger prunes the the categories so that the parser can build the complete set of derivations given those categories. In contrast, our parser can efficiently search large (or infinite) spaces of categories, but pruning is helpful for making supertagging itself more efficient, and for building the initial agenda.

We therefore implemented the following strategies to improve efficiency:

- Only allowing at most 50 categories per word. The C&C parser takes on average 1.27 tags per word (and an average of 3.57 at its loosest beam setting), so this restriction is a very mild one. Nevertheless, it considerably reduces the potential size of the agenda.

- Using a variable-width beam β which prunes categories less likely than β times the probability of the best category. We set $\beta = 0.00001$, which is two orders-of-magnitude smaller than the equivalent C&C beam. Again, this heuristic is useful for reducing the length of the agenda.

- Using a tag dictionary of possible categories for each word, so that weights are only calculated for a subset of the categories. Unlike the other methods, this approach does affect the probabilities which are calculated, as the normalizing constant is only computed for a subset of the categories. However, the probability mass contained in the pruned categories is small, and it only slightly decreases parsing accuracy. To build the tag dictionary, we parsed 42 million sentences of Wikipedia using our parser, and for all words occurring at least 500 times, we stored the set of observed word-category combinations. When parsing new sentences, these words are only allowed to occur with one of these categories.

Supertagger	Parser	CCGBank				Wikipedia			Bioinfer		
		F1 (cov)	COV	F1 (all)	Time	F1 (cov)	COV	F1 (all)	F1 (cov)	COV	F1 (all)
C&C	C&C	85.47	99.63	85.30	54s	81.19	99.0	80.64	76.08	97.2	74.88
EASYCCG	EASYCCG	83.37	99.96	83.37	13s	81.75	100	81.75	77.24	100	77.24
EASYCCG	C&C	86.14	99.96	86.11	69s	82.46	100	82.46	78.00	99.8	77.88

Table 2: Parsing F1-scores for labelled dependencies across a range of domains. F1 (cov) refers to results on sentences which the parser is able to parse, and F1 (all) gives results over all sentences. For the EASYCCG results, scores are only over parses where the C&C dependency extraction script was successful, which was 99.3% on CCGBank, 99.5% on Wikipedia, and 100% on Bioinfer.

4 Experiments

4.1 Experimental Setup

We trained our model on Sections 02-21 of CCGBank (Hockenmaier and Steedman, 2007), using Section 00 for development. For testing, we used Section 23 of CCGBank, a Wikipedia corpus annotated by Honnibal and Curran (2009), and the Bioinfer corpus of biomedical abstracts (Pyysalo et al., 2007). The latter two are out-of-domain, so are more challenging for the parsers.

We compare the performance of our model against both the C&C parser, and the system described in Lewis and Steedman (2014). This model uses the same supertagger as used in EASYCCG, but uses the C&C parser for parsing, using adaptive supertagging with the default values.

All timing experiments used the same 1.8Ghz AMD machine.

4.2 Parsing Accuracy

Results are shown in Table 2. Our parser performs competitively with a much more complex parsing model, and outperforms the C&C pipeline on both out-of-domain datasets. This result confirms our hypothesis that the majority of parsing decisions can be made accurately with a simple tagging model and a deterministic parser.

We see that the combination of the EASYCCG supertagger and the C&C parser achieves the best accuracy across all domains. This result shows that, unsurprisingly, there is some value to having a statistical model of the dependencies that the parser is evaluated on. However, the difference is not large, particularly out-of-domain, considering that a sophisticated and complex statistical parser is being compared with a deterministic one. Our parser is also far faster than this baseline.

It is interesting that the performance gap is much lower on out-of-domain datasets (2.8 points in domain, but only 0.65-0.75 out-of-domain), suggesting that much of the C&C parser's dependency model is domain specific, and does not generalize well to other domains.

	Speed (sentences/second)		
System	Tagger	Parser	Total
C&C	**343**	52	45
EASYCCG tagger + C&C parser	299	58	49
EASYCCG baseline	56	222	45
+Tag Dictionary	185	217	99
+Max 50 tags/word	238	345	141
+β=0.00001	299	**493**	**186**
EASYCCG — *null heuristic*	300	221	127

Table 3: Effect of our optimizations of parsing speed.

We also briefly experimented using the C&C supertagger (with a beam of $\beta = 10^{-5}$) with the EASYCCG parser. Performance was much worse, with an F-score of 79.63% on the 97.8% of sentences it parsed on CCGBank Section 23. This shows that our model is reliant on the accuracy of the supertagger.

4.3 Parsing Speed

CCG parsers have been used in distributional approaches to semantics (Lewis and Steedman, 2013a; Lewis and Steedman, 2013b), which benefit from large corpora. However, even though the C&C parser is relatively fast, it will still take over 40 CPU-days to parse the Gigaword corpus on our hardware, which is slow enough to be an obstacle to scaling distributional semantics to larger cor-

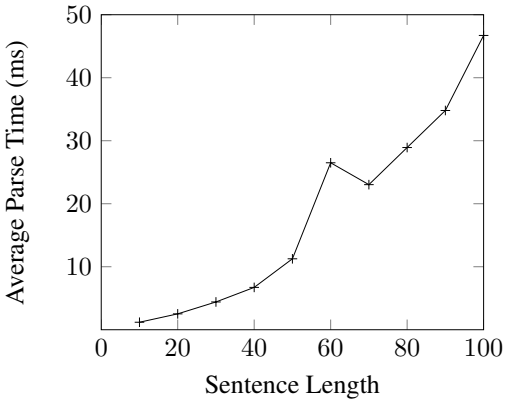


Figure 2: Average parse times in milliseconds, by sentence length.

pora such as ClueWeb. Therefore, it is important to be able to parse sentences at a high speed.

We measured parsing times on Section 23 of CCGBank (after developing against Section 00), using the optimizations described in Section 4.3. We also experimented with the *null heuristic*, which always estimates the outside probability as being 1.0. Times exclude the time taken to load models.

Results are shown in Table 3. The best EASYCCG model is roughly four times faster than the C&C parser[2]. Adding the tag dictionary caused accuracy to drop slightly from 83.46 to 83.37, and meant the parser failed to parse a single sentence in the test set ("*Among its provisions :*") but other changes did not affect accuracy. The pruning in the supertagger improves parsing speed, by limiting the length of the priority queue it builds for the agenda. Of course, we could use a backoff model to ensure full coverage (analogously to adaptive supertagging), but we leave that to future work. Using our A* heuristic doubles the speed of parsing (excluding supertagging).

To better understand the properties of our model, we also investigate how parsing time varies with sentence length. Unlike the cubic CKY algorithm typically used by chart parsers, our A* search potentially takes exponential time in the sentence length. For this experiment, we used the Sections 02-21 of CCGBank. Sentences were divided into bins of width 10, and we calculated the average parsing time for sentences in each bin.

Results are shown in Figure 2, and demonstrate that while parsing is highly efficient for sentences of up to 50 words (over 95% of CCGBank), it scales super-linearly with long sentences. In fact, Section 00 contains a sentence of 249 words, which took 37 *seconds* to parse (3 times longer than the other 1912 sentences put together). In practice, this scaling is unlikely to be problematic, as long sentences are typically filtered when processing large corpora.

4.4 Semantic Parsing

A major motivation for CCG parsing is to exploit its transparent interface to the semantics, allowing syntactic parsers to do much of the work of semantic parsers. Therefore, perhaps the most relevant measure of the performance of a CCG parser is its effect on the accuracy of downstream applications.

We experimented with a supervised version of Reddy et al. (2014)'s model for question-answering on Freebase (i.e. without using Reddy et al.'s lexicon derived from unlabelled text), using the WEBQUESTIONS dataset (Berant et al., 2013)[3]. The model learns to map CCG parses to database queries. We compare the performance of the QA system using both our parser and C&C, taking the 10-best parses from each parser for each sentence. Syntactic question parsing models were trained from the combination of 10 copies of Rimell and Clark (2008)'s question dataset and one copy of the CCGBank

The accuracy of Reddy et al. (2014)'s model varies significantly between iterations of the training data. Rather than tune the number of iterations, we instead measure the accuracy after each iteration. We experimented with the models' 1-best answers, and the oracle accuracy of their 100 best answers. The oracle accuracy gives a better indication of the performance of the parser, by mitigating errors caused by the semantic component.

Results are shown in Figure 3, and demonstrate that using EASYCCG can lead to better downstream performance than the C&C parser. The improvement is particularly large on oracle accuracy, increasing the upper bound on the performance of the semantic parser by around 4 points.

5 Related Work

CCG parsing has been the subject of much research. We have already described the C&C pars-

[2]It is worth noting that the C&C parser code is written in highly-optimized C++, compared to our simple Java implementation. It seems likely that our parser could be made substantially faster with a similar level of engineering effort.

[3]Using the *Business*, *Film* and *People* domains, with 1115 questions for training and 570 for testing.

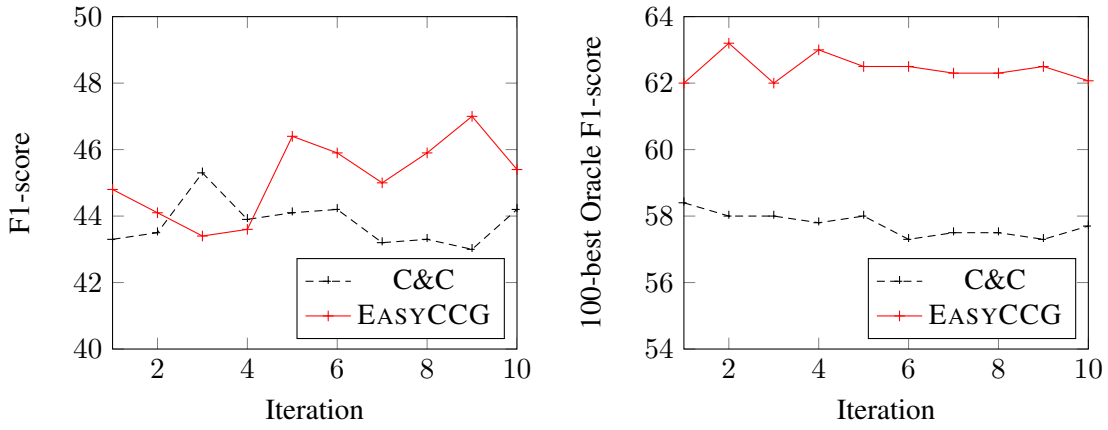


Figure 3: Question Answering accuracy per iteration of Reddy et al. (2014)'s supervised model.

ing model. Kummerfeld et al. (2010) showed that the speed of the C&C parser can be improved with domain-specific self-training—similar improvements may be possible applying this technique to our model. Auli and Lopez (2011a) have achieved the best CCG parsing accuracy, by allowing the parser and supertagger to perform joint inference (though there is a significant speed penalty). Auli and Lopez (2011b) were the first to use A* parsing for CCG, but their system is both much slower and less accurate than ours (due to a different model and a different A* heuristic). Krishnamurthy and Mitchell (2014) show how CCG parsing can be improved by jointly modelling the syntax and semantics. Fowler and Penn (2010) apply the Petrov parser to CCG, making a small improvement in accuracy over the C&C parser, at the cost of a 300-fold speed decrease. Zhang and Clark (2011) and Xu et al. (2014) explored shift-reduce CCG parsing, but despite the use of a linear-time algorithm, parsing speed in practice is significantly slower than the C&C parser.

Parsers based on supertagging models have previously been applied to other strongly lexicalized formalisms, such as to LTAG (Bangalore and Joshi, 1999) and to HPSG (Ninomiya et al., 2006). A major contribution of our work over these is showing that factoring models on lexical categories allows fast and exact A* parsing, without the need for beam search. Our parsing approach could be applied to any strongly lexicalized formalism.

Our work fits into a tradition of attempting to simplify complex models without sacrificing performance. Klein and Manning (2003b) showed that unlexicalized parsers were only slightly less accurate than their lexicalized counterparts. Collobert et al. (2011) showed how a range of NLP tagging tasks could be performed at high accuracy using a small feature set based on vector-space word embeddings. However, the extension of this work to phrase-structure parsing (Collobert, 2011) required a more complex model, and did not match the performance of traditional parsing techniques. We achieve state-of-the-art results using the same feature set and a simpler model by exploiting CCG's lexicalized nature, which makes it more natural to delegate parsing decisions to a tagging model.

Other parsing research has focused on building fast parsers for web-scale processing, typically using dependency grammars (e.g. Nivre (2003)). CCG has some advantages over dependency grammars, such as supporting surface-compositional semantics. The fastest dependency parsers use an *easy-first* strategy, in which edges are added greedily in order of their score, with $\mathcal{O}(nlog(n))$ complexity (Goldberg and Elhadad, 2010; Tratz and Hovy, 2011). This strategy is reminiscent of our A* search, which expands the chart in a best-first order. A* has higher asymptotic complexity, but finds a globally optimal solution.

6 Future Work

We believe that our model opens several interesting directions for future research.

One interesting angle would be to increase the amount of information in CCGBank's lexical entries, to further reduce the search space for the parser. For example, *PP* categories could be distinguished with the relevant preposition as a feature; punctuation and coordination could be given more detailed categories to avoid needing their own combinators, and slashes could be extended

with Baldridge and Kruijff (2003)'s multi-modal extensions to limit over-generation. Honnibal and Curran (2009) show how unary rules can be lexicalized in CCG. Such improvements may improve both the speed and accuracy of our model.

Because our parser is factored on a unigram tagging model, it can be trained from isolated annotated words, and does not require annotated parse trees or full sentences. Reducing the requirements for training data eases the task for human annotators. It may also make the model more amenable to semi-supervised approaches to CCG parsing, which have typically focused on extending the lexicon (Thomforde and Steedman, 2011; Deoskar et al., 2014). Finally, it may make it easier to convert other annotated resources, such as UCCA (Abend and Rappoport, 2013) or AMR (Banarescu et al., 2013), to CCG training data—as only specific words need to be converted, rather than full sentences.

Our model is weak at certain kinds of decisions, e.g. coordination-scope ambiguities or non-local attachments. Incorporating specific models for such decisions may improve accuracy, while still allowing fast and exact search—for example, we intend to try including Coppola et al. (2011)'s model for prepositional phrase attachment.

7 Conclusions

We have shown that a simple, principled, deterministic parser combined with a tagging model can parse an expressive linguistic formalism with high speed and accuracy. Although accuracy is not state-of-the-art on CCGBank, our model gives excellent performance on two out-of-domain datasets, and improves the accuracy of a question-answering system. We have shown that this model allows an efficient heuristic for A* parsing, which makes parsing extremely fast, and may enable logic-based distributional semantics to scale to larger corpora. Our methods are directly applicable to other lexicalized formalisms, such as LTAG, LFG and HPSG.

Acknowledgments

We would like to thank Tejaswini Deoskar, Bharat Ram Ambati, Michael Roth and the anonymous reviewers for helpful feedback on an earlier version of this paper, and Siva Reddy for running the semantic parsing experiments.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of ACL*.

Michael Auli and Adam Lopez. 2011a. A comparison of loopy belief propagation and dual decomposition for integrated CCG supertagging and parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 470–480. Association for Computational Linguistics.

Michael Auli and Adam Lopez. 2011b. Efficient CCG parsing: A* versus adaptive supertagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1577–1585. Association for Computational Linguistics.

Jason Baldridge and Geert-Jan M Kruijff. 2003. Multi-modal combinatory categorial grammar. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 211–218. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, Sofia, Bulgaria, August. Association for Computational Linguistics.

Srinivas Bangalore and Aravind K Joshi. 1999. Supertagging: An approach to almost parsing. *Computational linguistics*, 25(2):237–265.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*.

Johan Bos. 2008. Wide-coverage semantic analysis with boxer. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, Research in Computational Semantics, pages 277–286. College Publications.

Stephen Clark and James R Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

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

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics*, number EPFL-CONF-192374.

Gregory F Coppola, Alexandra Birch, Tejaswini Deoskar, and Mark Steedman. 2011. Simple semi-supervised learning for prepositional phrase attachment. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 129–139. Association for Computational Linguistics.

Tejaswini Deoskar, Christos Christodoulopoulos, Alexandra Birch, and Mark Steedman. 2014. Generalizing a Strongly Lexicalized Parser using Unlabeled Data. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Jason Eisner. 1996. Efficient normal-form parsing for combinatory categorial grammar. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 79–86. Association for Computational Linguistics.

Timothy AD Fowler and Gerald Penn. 2010. Accurate context-free parsing with combinatory categorial grammar. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 335–344. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750. Association for Computational Linguistics.

Brian Harrington. 2010. A semantic network approach to measuring relatedness. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 356–364. Association for Computational Linguistics.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Julia Hockenmaier. 2003. Data and models for statistical parsing with combinatory categorial grammar.

Matthew Honnibal and James R Curran. 2009. Fully lexicalising CCGbank with hat categories. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1212–1221. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2003a. A* parsing: fast exact viterbi parse selection. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 40–47. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2003b. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Jayant Krishnamurthy and Tom M Mitchell. 2014. Joint syntactic and semantic parsing with combinatory categorial grammar. June.

Jonathan K. Kummerfeld, Jessika Roesner, Tim Dawborn, James Haggerty, James R. Curran, and Stephen Clark. 2010. Faster parsing by supertagger adaptation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 345–355. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2013a. Combined Distributional and Logical Semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Mike Lewis and Mark Steedman. 2013b. Unsupervised induction of cross-lingual semantic relations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 681–692, Seattle, Washington, USA, October. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2014. Improved CCG parsing with Semi-supervised Supertagging. *Transactions of the Association for Computational Linguistics (to appear)*.

Takashi Ninomiya, Takuya Matsuzaki, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2006. Extremely lexicalized models for accurate and fast hpsg parsing. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 155–163, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. 2007. Bioinfer: a corpus for information extraction in the biomedical domain. *BMC bioinformatics*, 8(1):50.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale Semantic Parsing without Question-Answer Pairs. *Transactions of the Association for Computational Linguistics (to appear)*.

Laura Rimell and Stephen Clark. 2008. Adapting a lexicalized-grammar parser to contrasting domains. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 475–484. Association for Computational Linguistics.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Mark Steedman. 2012. *Taking Scope: The Natural Semantics of Quantifiers*. MIT Press.

Emily Thomforde and Mark Steedman. 2011. Semi-supervised CCG lexicon extension. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1246–1256. Association for Computational Linguistics.

Stephen Tratz and Eduard Hovy. 2011. A fast, effective, non-projective, semantically-enriched parser. In *Proceedings of EMNLP*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Wenduan Xu, Stephen Clark, and Yue Zhang. 2014. Shift-reduce ccg parsing with a dependency model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*. Association for Computational Linguistics, June.

Yue Zhang and Stephen Clark. 2011. Shift-reduce CCG parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 683–692. Association for Computational Linguistics.

A Dependency Parser for Tweets

Lingpeng Kong Nathan Schneider Swabha Swayamdipta
Archna Bhatia Chris Dyer Noah A. Smith
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{lingpenk,nschneid,swabha,archna,cdyer,nasmith}@cs.cmu.edu

Abstract

We describe a new dependency parser for English tweets, TWEEBOPARSER. The parser builds on several contributions: new syntactic annotations for a corpus of tweets (TWEEBANK), with conventions informed by the domain; adaptations to a statistical parsing algorithm; and a new approach to exploiting out-of-domain Penn Treebank data. Our experiments show that the parser achieves over 80% unlabeled attachment accuracy on our new, high-quality test set and measure the benefit of our contributions.

Our dataset and parser can be found at http://www.ark.cs.cmu.edu/TweetNLP.

1 Introduction

In contrast to the edited, standardized language of traditional publications such as news reports, social media text closely represents language as it is used by people in their everyday lives. These informal texts, which account for ever larger proportions of written content, are of considerable interest to researchers, with applications such as sentiment analysis (Greene and Resnik, 2009; Kouloumpis et al., 2011). However, their often nonstandard content makes them challenging for traditional NLP tools. Among the tools currently available for tweets are a POS tagger (Gimpel et al., 2011; Owoputi et al., 2013) and a named entity recognizer (Ritter et al., 2011)—but not a parser.

Important steps have been taken. The English Web Treebank (Bies et al., 2012) represents an annotation effort on web text—which likely lies somewhere between newspaper text and social media messages in formality and care of editing—that was sufficient to support a shared task (Petrov and McDonald, 2012). Foster et al. (2011b) annotated a small test set of tweets to evaluate parsers trained on the Penn Treebank (Marcus et al., 1993), augmented using semi-supervision and in-domain data. Others, such as Soni et al. (2014), have used existing Penn Treebank–trained models on tweets.

In this work, we argue that the Penn Treebank approach to annotation—while well-matched to edited genres like newswire—is poorly suited to more informal genres. Our starting point is that rapid, small-scale annotation efforts performed by imperfectly-trained annotators should provide enough evidence to train an effective parser. We see this starting point as a necessity, given observations about the rapidly changing nature of tweets (Eisenstein, 2013), the attested difficulties of domain adaptation for parsing (Dredze et al., 2007), and the expense of creating Penn Treebank–style annotations (Marcus et al., 1993).

This paper presents TWEEBOPARSER, the first syntactic dependency parser designed explicitly for English tweets. We developed this parser following current best practices in empirical NLP: we annotate a corpus (TWEEBANK) and train the parameters of a statistical parsing algorithm. Our research contributions include:

- a survey of key challenges posed by syntactic analysis of tweets (by humans or machines) and decisions motivated by those challenges and by our limited annotation-resource scenario (§2);
- our annotation process and quantitative measures of the quality of the annotations (§3);
- adaptations to a statistical dependency parsing algorithm to make it fully compatible with the above, and also to exploit information from out-of-domain data cheaply and without a strong commitment (§4); and
- an experimental analysis of the parser's unlabeled attachment accuracy—which surpasses 80%—and contributions of various important components (§5).

The dataset and parser can be found at http://www.ark.cs.cmu.edu/TweetNLP.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1001–1012,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

2 Annotation Challenges

Before describing our annotated corpus of tweets (§3), we illustrate some of the challenges of syntactic analysis they present. These challenges motivate an approach to annotation that diverges significantly from conventional approaches to treebanking. Figure 1 presents a single example illustrating four of these: token selection, multiword expressions, multiple roots, and structure within noun phrases. We discuss each in turn.

2.1 Token Selection

Many elements in tweets have no syntactic function. These include, in many cases, hashtags, URLs, and emoticons. For example:

RT @justinbieber : now Hailee get a twitter

The retweet discourse marker, username, and colon should not, we argue, be included in the syntactic analysis. By contrast, consider:

Got #college admissions questions ? Ask them tonight during #CampusChat I'm looking forward to advice from @collegevisit http://bit.ly/cch0Tk

Here, both the hashtags and the at-mentioned username are syntactically part of the utterances, while the punctuation and the hyperlink are not. In the example of Figure 1, the unselected tokens include several punctuation tokens and the final token #believer, which marks the topic of the tweet.

Typically, dependency parsing evaluations ignore punctuation token attachment (Buchholz and Marsi, 2006), and we believe it is a waste of annotator (and parser) time to decide how to attach punctuation and other non-syntactic tokens. Ma et al. (2014) recently proposed to treat punctuation as context features rather than dependents, and found that this led to state-of-the-art performance in a transition-based parser. A small adaptation to our graph-based parsing approach, described in §4.2, allows a similar treatment.

Our approach to annotation (§3) forces annotators to explicitly select tokens that have a syntactic function. 75.6% tokens were selected by the annotators. Against the annotators' gold standard, we found that a simple rule-based filter for usernames, hashtags, punctuation, and retweet tokens achieves 95.2% (with gold-standard POS tags) and 95.1% (with automatic POS tags) average accuracy in the task of selecting tokens with a syntactic function in a ten-fold cross-validation experiment. To take context into account, we developed a first-order sequence model and found that it achieves 97.4% average accuracy (again, ten-fold cross-validated) with either gold-standard or automatic POS tags. Features include POS; shape features that recognize the retweet marker, hashtags, usernames, and hyperlinks; capitalization; and a binary feature for tokens that include punctuation. We trained the model using the structured perceptron (Collins, 2002).

2.2 Multiword Expressions

We consider multiword expressions (MWEs) of two kinds. The first, proper names, have been widely modeled for information extraction purposes, and even incorporated into parsing (Finkel and Manning, 2009). (An example found in Figure 1 is LA Times.) The second, lexical idioms, have been a "pain in the neck" for many years (Sag et al., 2002) and have recently received shallow treatment in NLP (Baldwin and Kim, 2010; Constant and Sigogne, 2011; Schneider et al., 2014). Constant et al. (2012), Green et al. (2012), Candito and Constant (2014), and Le Roux et al. (2014) considered MWEs in parsing. Figure 1 provides LA Times and All the Rage as examples.

Penn Treebank–style syntactic analysis (and dependency representations derived from it) does not give first-class treatment to this phenomenon, though there is precedent for marking multiword lexical units and certain kinds of idiomatic relationships (Hajič et al., 2012; Abeillé et al., 2003).[1]

We argue that internal analysis of MWEs is not critical for many downstream applications, and therefore annotators should not expend energy on developing and respecting conventions (or making arbitrary decisions) within syntactically opaque or idiosyncratic units. We therefore allow annotators to decide to group words as explicit MWEs, including: proper names (Justin Bieber, World Series), noncompositional or entrenched nominal compounds (belly button, grilled cheese), connectives (as well as), prepositions (out of), adverbials (so far), and idioms (giving up, make sure).

From an annotator's perspective, a MWE functions as a *single node* in the dependency parse, with no internal structure. For idioms whose internal syntax *is* easily characterized, the parse can be used to capture compositional structure, an at-

[1] The popular Stanford typed dependencies (de Marneffe and Manning, 2008) scheme includes a special dependency type for multiwords, though this is only applied to a small list.

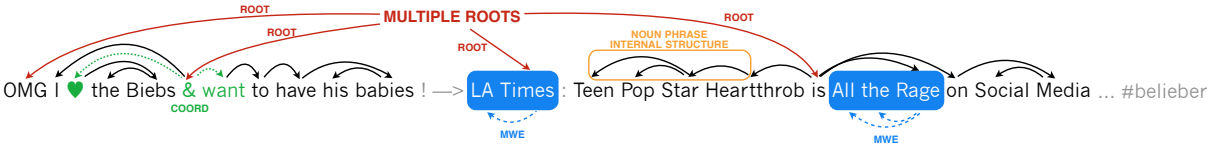


Figure 1: Parse tree for a (constructed) example illustrating annotation challenges discussed in §2. Colors highlight token selection (gray; §2.1), multiword expressions (blue; §2.2), multiple roots (red; §2.3), coordination (dotted arcs, green; §3.2), and noun phrase internal structure (orange; §2.4). The internal structure of multiword expressions (dashed arcs below the sentence) was predicted automatically by a parser, as described in §2.2.

tractive property from the perspective of semantic processing.

To allow training a fairly conventional statistical dependency parser from these annotations, we find it expedient to apply an automatic conversion to the MWE annotations, in the spirit of Johnson (1998). We apply an existing dependency parser, the first-order TurboParser (Martins et al., 2009) trained on the Penn Treebank, to parse each MWE independently, assigning structures like those for LA Times and All the Rage in Figure 1. Arcs involving the MWE in the annotation are then reconnected to the MWE-internal root, so that the resulting tree respects the original tokenization. The MWE-internal arcs are given a special label so that the transformation can be reversed and MWEs reconstructed from parser output.

2.3 Multiple Roots

For news text such as that found in the Penn Treebank, sentence segmentation is generally considered a very easy task (Reynar and Ratnaparkhi, 1997). Tweets, however, often contain multiple sentences or fragments, which we call "utterances," each with its own syntactic root disconnected from the others. The selected tokens in Figure 1 comprise four utterances.

Our approach to annotation allows multiple utterances to emerge directly from the connectedness properties of the graph implied by an annotator's decisions. Our parser allows multiple attachments to the "wall" symbol, so that multi-rooted analyses can be predicted.

2.4 Noun Phrase Internal Structure

A potentially important drawback of deriving dependency structures from phrase-structure annotations, as is typically done using the Penn Treebank, is that flat annotations lead to loss of information. This is especially notable for noun phrases in the Penn Treebank (Vadas and Curran, 2007). Consider Teen Pop Star Heartthrob in Figure 1; Penn Treebank conventions would label this as a single NP with four NN children and no internal structure. Dependency conversion tools would likely attach the first three words in the NP to Heartthrob. Direct dependency annotation (rather than phrase-structure annotation followed by automatic conversion) allows a richer treatment of such structures, which is potentially important for semantic analysis (Vecchi et al., 2013).

3 A Twitter Dependency Corpus

In this section, we describe the TWEEBANK corpus, highlighting data selection (§3.1), the annotation process (§3.2), important convention choices (§3.3), and measures of quality (§3.4).

3.1 Data Selection

We added manual dependency parses to 929 tweets (12,318 tokens) drawn from the POS-tagged Twitter corpus of Owoputi et al. (2013), which are tokenized and contain manually annotated POS tags.

Owoputi et al.'s data consists of two parts. The first, originally annotated by Gimpel et al. (2011), consists of tweets sampled from a particular day, October 27, 2010—this is known as OCT27. Due to concerns about overfitting to phenomena specific to that day (e.g., tweets about a particular sports game), Owoputi et al. (2013) created a new set of 547 tweets (DAILY547) consisting of one random English tweet per day from January 2011 through June 2012.

Our corpus is drawn roughly equally from OCT27 and DAILY547.[2] Despite its obvious temporal skew, there is no reason to believe this sample is otherwise biased; our experiments in §5 suggest that this property is important.

3.2 Annotation

Unlike a typical treebanking project, which may take years and involve thousands of person-hours of work by linguists, most of TWEEBANK was built in a day by two dozen annotators, most of whom had only cursory training in the annotation scheme.

[2]This results from a long-term goal to fully annotate both.

(1) RT @FRIENDSHIP : Friendship is love without kissing ...
```
Friendship > is < love < without < kissing
```

(2) bieber is an alien ! :O he went down to earth .
```
bieber > is** < alien < an
he > [went down]** < to < earth
```

(3) RT @YourFavWhiteGuy : Helppp meeeee . I'mmm meltiiinngggg → http://twitpic.com/316cjg
```
Helppp** < meeeee
I'mmm** < meltiiinngggg
```

Figure 2: Examples of GFL annotations from the corpus.

Our annotators used the Graph Fragment Language (GFL), a text-based notation that facilitates keyboard entry of parses (Schneider et al., 2013). A Python Flask web application allows the annotator to validate and visualize each parse (Mordowanec et al., 2014). Some examples are shown in Figure 2. Note that all of the challenges in §2 are handled easily by GFL notation: "retweet" information, punctuation, and a URL are not selected by virtue of their exclusion from the GFL expression; in (2) went down is annotated as a MWE using GFL's square bracket notation; in (3) the tokens are grouped into two utterances whose roots are marked by the ** symbol.

Schneider et al.'s GFL offers some additional features, only some of which we made use of in this project. One important feature allows an annotator to leave the parse *underspecified* in some ways. We allowed our annotators to make use of this feature; however, we excluded from our training and testing data any parse that was incomplete (i.e., any parse that contained multiple disconnected fragments with no explicit root, excluding unselected tokens). Learning to parse from incomplete annotations is a fascinating topic explored in the past (Hwa, 2001; Pereira and Schabes, 1992) and, in the case of tweets, left for future work.

An important feature of GFL that we did use is special notation for coordination structures. For the coordination structure in Figure 1, for example, the notation is:

```
$a :: {♥ want} :: {&}
```

where `$a` creates a new node in the parse tree as it is visualized for the annotator, and this new node attaches to the syntactic parent of the conjoined structure, avoiding the classic forced choice between coordinator and conjunct as parent. For learning to parse, we transform GFL's coordination structures into specially-labeled dependency parses collapsing nodes like `$a` with the coordinator and labeling the attachments specially for postprocessing, following Schneider et al. (2013). In our evaluation (§5), these are treated like other attachments.

3.3 Annotation Conventions

A wide range of dependency conventions are in use; in many cases these are *conversion* conventions specifying how dependency trees can be derived from phrase-structure trees. For English, the most popular are due to Yamada and Matsumoto (2003) and de Marneffe and Manning (2008), known as "Yamada-Matsumoto" (YM) and "Stanford" dependencies, respectively. The main differences between them are in whether the auxiliary is the parent of the main verb (or vice versa) and whether the preposition or its argument heads a prepositional phrase (Elming et al., 2013).

A full discussion of our annotation conventions is out of scope. We largely followed the conventions suggested by Schneider et al. (2013), which in turn are close to those of YM. Auxiliary verbs are parents of main verbs, and prepositions are parents of their arguments. The key differences from YM are in coordination structures (discussed in §3.2; YM makes the first conjunct the head) and possessive structures, in which the possessor is the child of the clitic, which is the child of the semantic head, e.g., `the > king > 's > horses`.

3.4 Intrinsic Quality

Our approach to developing this initial corpus of syntactically annotated tweets was informed by an aversion to making the perfect the enemy of the good; that is, we sought enough data of sufficient quality to build a usable parser within a relatively short amount of time. If our research goals had been to develop a replicable process for annotation, more training and more quality control would have been called for. Under our budgeted time and annotator resources, this overhead was simply too costly. Nonetheless, we performed a few analyses that give a general picture of the quality of the annotations.

Inter-annotator agreement. 170 of the tweets were annotated by multiple users. By the *softComPrec* measure (Schneider et al., 2013),[3] the agreement rate on dependencies is above 90%.

Expert linguistic judgment. A linguist coauthor examined a stratified sample (balanced

[3] *softComPrec* is a generalization of attachment accuracy that handles unselected tokens and MWEs.

across annotators) of 60 annotations and rated their quality on a 5-point scale. 30 annotations were deemed to have "no obvious errors," 15 only minor errors, 3 a major error (i.e., clear violation of annotation guidelines),[4] 4 a major error and at least one minor error, and 8 as containing multiple major errors. Thus, 75% are judged as having no major errors. We found this encouraging, considering that this sample is skewed in favor of people who annotated less (including many of the less experienced and/or lower-proficiency annotators).

Pairwise ranking. For 170 of the doubly annotated tweets, an experienced annotator examined whether one or the other was markedly better. In 100 cases the two annotations were of comparable quality (neither was obviously better) and did not contain any obvious major errors. In only 7 pairs did both of the annotations contain a serious error.

Qualitatively, we found several unsurprising sources of error or disagreement, including embedded/subordinate clauses, subject-auxiliary inversion, predeterminers, and adverbial modifiers following a modal/auxiliary verb and a main verb. Clarification of the conventions, or even explicit rule-based checking in the validation step, might lead to quality improvements in further annotation efforts.

4 Parsing Algorithm

For parsing, we start with TurboParser, which is open-source and has been found to perform well on a range of parsing problems in different languages (Martins et al., 2013; Kong and Smith, 2014). The underlying model allows for flexible incorporation of new features and changes to specification in the output space. We briefly review the key ideas in TurboParser (§4.1), then describe decoder modifications required for our problem (§4.2). We then discuss features we added to TurboParser (§4.3).

4.1 TurboParser

Let an input sentence be denoted by x and the set of possible dependency parses for x be denoted by $\mathcal{Y}_x$. A generic linear scoring function based on a feature vector representation $\mathbf{g}$ is used in parsing algorithms that seek to find:

$$\text{parse}^*(x) = \arg\max_{y \in \mathcal{Y}_x} \mathbf{w}^\top \mathbf{g}(x, y) \quad (1)$$

The score is parameterized by a vector $\mathbf{w}$ of weights, which are learned from data (most commonly using MIRA, McDonald et al., 2005a).

The decomposition of the features into local "parts" is a critical choice affecting the computational difficulty of solving Eq. 1. The most aggressive decomposition leads to an "arc-factored" or "first-order" model, which permits exact, efficient solution of Eq. 1 using spanning tree algorithms (McDonald et al., 2005b) or, with a projectivity constraint, dynamic programming (Eisner, 1996).

Second- and third-order models have also been introduced, typically relying on approximations, since less-local features increase the computational cost, sometimes to the point of NP-hardness (McDonald and Satta, 2007). TurboParser attacks the parsing problem using a compact integer linear programming (ILP) representation of Eq. 1 (Martins et al., 2009), then employing alternating directions dual decomposition (AD^3; Martins et al., 2011). This enables inclusion of second-order features (e.g., on a word with its sibling or grandparent; Carreras, 2007) and third-order features (e.g., a word with its parent, grandparent, and a sibling, or with its parent and two siblings; Koo and Collins, 2010).

For a collection of (possibly overlapping) parts for input x, $\mathcal{S}_x$ (which includes the union of all parts of all trees in $\mathcal{Y}_x$), we will use the following notation. Let

$$\mathbf{g}(x, y) = \sum_{s \in \mathcal{S}_x} \mathbf{f}_s(x, y), \quad (2)$$

where $\mathbf{f}_s$ only considers part s and is nonzero only if s is present in y. In the ILP framework, each s has a corresponding binary variable z_s indicating whether part s is included in the output. A collection of constraints relating z_s define the set of feasible vectors $\mathbf{z}$ that correspond to valid outputs and enfore agreement between parts that overlap. Many different versions of these constraints have been studied (Riedel and Clarke, 2006; Smith and Eisner, 2008; Martins et al., 2009, 2010).

A key attraction of TurboParser is that many overlapping parts can be handled, making use of separate combinatorial algorithms for efficiently handling *subsets* of constraints. For example, the constraints that force $\mathbf{z}$ to encode a valid tree can be exploited within the framework by making calls

[4] What we deemed *major* errors included, for example, an incorrect dependency relation between an auxiliary verb and the main verb (like `ima > [have to]`). *Minor* errors included an incorrect attachment between two modifiers of the same head, as in `the > only > [grocery store]`—the correct annotation would have two attachments to a single head, i.e. `the > [grocery store] < only` (or equivalent).

to classic arborescence algorithms (Chu and Liu, 1965; Edmonds, 1967). As a result, when describing modifications to TurboParser, we need only to explain additional constraints and features imposed on *parts*.

4.2 Adapted Parse Parts

The first collection of parts we adapt are simple arcs, each consisting of an ordered pair of indices of words in x; $\text{arc}(p,c)$ corresponds to the attachment of x_c as a child of x_p (iff $z_{\text{arc}(p,c)} = 1$). Our representation explicitly excludes some tokens from being part of the syntactic analysis (§2.1); to handle this, we constrain $z_{\text{arc}(i,j)} = 0$ whenever x_i or x_j is excluded.

The implication is that excluded tokens are still "visible" to feature functions that involve other edges. For example, some conventional first-order features consider the tokens occurring *between* a parent and child. Even if a token plays no syntactic role of its own, it might still be informative about the syntactic relationships among other tokens. We note three alternative methods:

1. We might remove all unselected tokens from x before running the parser. In §5.6 we find this method to fare 1.7–2.3% worse than our modified decoding algorithm.
2. We might remove unselected tokens but use them to define new features, so that they still serve as evidence. This is the approach taken by Ma et al. (2014) for punctuation. We judge our simple modification to the decoding algorithm to be more expedient, and leave the translation of existing context-word features into that framework for future exploration.
3. We might incorporate the token selection decisions into the parser, performing joint inference for selection and parsing. The AD^3 algorithm within TurboParser is well-suited to this kind of extension: z-variables for each token's selection could be added, and similar scores to those of our token selection sequence model (§2.1) could be integrated into parsing. Given, however, that the sequence model achieves over 97% accuracy, and that perfect token selection would gain only 0.1–1% in parsing accuracy (reported in §5.5), we leave this option for future work as well.

For first-order models, the above change is all that is necessary. For second- and third-order models, TurboParser makes use of head automata, in particular "grand-sibling head automata" that assign scores to word tuples of x_g, its child x_p, and two of x_p's adjacent children, x_c and x'_c (Koo et al., 2010). The second-order models in our experiments include parts for $\text{sibling}(p,c,c')$ and $\text{grandparent}(p,c,g)$ and use the grand-sibling head automaton to reason about these together. Automata for an unselected x_p or x_g, and transitions that consider unselected tokens as children, are eliminated. In order to allow the scores to depend on unselected tokens between x_c and x'_c, we added the binned counts of unselected tokens (mostly punctuation) joint with the word form and POS tag of x_p and the POS tag of x_c and x'_c as features scored in the $\text{sibling}(p,c,c')$ part. The changes discussed above comprise the totality of adaptations we made to the TurboParser algorithm; we refer to them as "parsing adaptations" in the experiments.

4.3 Additional Features

Brown clusters. Owoputi et al. (2013) found that Brown et al. (1992) clusters served as excellent features in Twitter POS tagging. Others have found them useful in parsing (Koo et al., 2008) and other tasks (Turian et al., 2010). We therefore follow Koo et al. in incorporating Brown clusters as features, making use of the publicly available Twitter clusters from Owoputi et al.[5] We use 4 and 6 bit cluster representations to create features wherever POS tags are used, and full bit strings to create features wherever words were used.

Penn Treebank features. A potential danger of our choice to "start from scratch" in developing a dependency parser for Twitter is that the resulting annotation conventions, data, and desired output are very different from dependency parses derived from the Penn Treebank. Indeed, Foster et al. (2011a) took a very different approach, applying Penn Treebank conventions in annotation of a test dataset for evaluation of a parser trained using Penn Treebank trees. In §5.4, we replicate, for dependencies, their finding that a Penn Treebank–trained parser is hard to beat *on their dataset*, which was not designed to be topically representative of English Twitter. When we turn to a more realistic dataset like ours, we find the performance of the Penn Treebank–trained parser to be poor.

Nonetheless, it is hard to ignore such a large amount of high-quality syntactic data. We there-

[5] http://www.ark.cs.cmu.edu/TweetNLP/clusters/50mpaths2

fore opted for a simple, stacking-inspired incorporation of Penn Treebank information into our model.[6] We define a feature on every candidate arc whose value is the (quantized) score of the same arc under a first-order model trained on the Penn Treebank converted using head rules that are as close as possible to our conventions (discussed in more detail in §5.1). This lets a Penn Treebank model literally "weigh in" on the parse for a tweet, and lets the learning algorithm determine how much consideration it deserves.

5 Experiments

Our experiments quantify the contributions of various components of our approach.

5.1 Setup

We consider two test sets. The first, TEST-NEW, consists of 201 tweets from our corpus annotated by the most experienced of our annotators (one of whom is a co-author of this work). Given very limited data, we believe using the highest quality data for measuring performance, and lower-quality data for training, is a sensibly realistic choice.

Our second test set, TEST-FOSTER, is the dataset annotated by Foster et al. (2011b), which consists of 250 sentences. Recall that their corpus was annotated with phrase structures according to Penn Treebank conventions. Conversion to match our annotation conventions was carried out as follows:

1. We used the PennConverter tool with head rule options selected to approximate our annotation conventions as closely as possible.[7]
2. An experienced annotator manually modified the automatically converted trees by:
   (a) Performing token selection (§2.1) to remove the tokens which have no syntactic function.
   (b) Grouping MWEs (§2.2). Here, most of the MWEs are named entities such as Manchester United.
   (c) Attaching the roots of the utterance in tweets to the "wall" symbol (§2.3).[8]
   (d) Recovering the internal structure of the noun phrases.
   (e) Fixing a difference in conventions with respect to subject-auxiliary inversion.[9]

	TRAIN	TEST-NEW	TEST-FOSTER
tweets	717	201	$< 250^{\dagger}$
unique tweets	569	201	$< 250^{\dagger}$
tokens	9,310	2,839	2,841
selected tokens	7,015	2,158	2,366
types	3,566	1,461	1,230
utterances	1,473	429	337
multi-root tweets	398	123	60
MWEs	387	78	109

Table 1: Statistics of our datasets. (A tweet with k annotations in the training set is counted k times for the totals of tokens, utterances, etc.). †TEST-FOSTER contains 250 manually split sentences. The number of tweets should be smaller but is not recoverable from the data release.

We consider two training sets. TRAIN-NEW consists of the remaining 717 tweets from our corpus (§3) annotated by the rest of the annotators. Some of these tweets have annotations from multiple annotators; 11 annotations for tweets that also occurred in TEST-NEW were excluded. TRAIN-PTB is the conventional training set from the Penn Treebank (§2–21). The PennConverter tool was used to extract dependencies, with head rule options selected to approximate our annotation conventions as closely as possible (see footnote 7). The resulting annotations lack the same attention to noun phrase–internal structure (§2.4) and handle subject-auxiliary inversions differently than our data. Part-of-speech tags were coarsened to be compatible with the Twitter POS tags, using the mappings specified by Gimpel et al. (2011).

Statistics for the in-domain datasets are given in Table 1. As we can see in the table, more than half of the tweets in our corpus have multiple utterances. The out-of-vocabulary rate for our TRAIN/TEST-NEW split is 33.7% by token and 62.5% by type; for TRAIN/TEST-FOSTER it is 41.4% and 64.6% respectively. These are much higher than the 2.5% and 13.2% in the standard Penn Treebank split.

All evaluations here are on unlabeled attachment F_1 scores.[10] Our parser provides labels for coordination structures and MWEs (§2), but we do not present detailed evaluations of those due to space constraints.

[6] Stacking is a machine learning method where the predictions of one model are used to create features for another. The second model may be from a different family. Stacking has been found successful for dependency parsing by Nivre and McDonald (2008) and Martins et al. (2008). Johansson (2013) describes further advances that use path features.

[7] `http://nlp.cs.lth.se/software/treebank_converter`; run with `-rightBranching=false -coordStructure=prague -prepAsHead=true -posAsHead=true -subAsHead=true -imAsHead=true -whAsHead=false`.

[8] This was infrequent; their annotations split most multi-utterance tweets into separate sentence-instances.

[9] For example, in the sentence Is he driving, we attached he to driving while PennConverter attaches it to Is.

[10] Because of token selection, precision and recall may not be equal.

5.2 Preprocessing

Because some of the tweets in our test set were also in the training set of Owoputi et al. (2013), we retrained their POS tagger on all the annotated data they have minus the 201 tweets in our test set. Its tagging accuracy was 92.8% and 88.7% on TEST-NEW and TEST-FOSTER, respectively. The token selection model (§2.1) achieves 97.4% on TEST-NEW with gold or automatic POS tagging; and on TEST-FOSTER, 99.0% and 99.5% with gold and automatic POS tagging, respectively.

As noted in §4.3, Penn Treebank features were developed using a first-order TurboParser trained on TRAIN-PTB; Brown clusters were included in computing these Penn Treebank features if they were available in the parser to which the features (i.e. Brown clusters) were added.

5.3 Main Parser

The second-order TurboParser described in §4, trained on TRAIN-NEW (default hyperparameter values), achieves 80.9% unlabeled attachment accuracy on TEST-NEW and 76.1% on TEST-FOSTER. The experiments consider variations on this main approach, which is the version released as TWEEBOPARSER.

The discrepancy between the two test sets is easily explained: as noted in §3.1, the dataset from which our tweets are drawn was designed to be representative of English on Twitter. Foster et al. (2011b) selected tweets from Bermingham and Smeaton's (2010) corpus, which uses fifty predefined topics like politics, business, sports, and entertainment—in short, topics not unlike those found in the Penn Treebank. Relative to the Penn Treebank training set, the by-type out-of-vocabulary rates are 45.2% for TEST-NEW and only 21.6% for TEST-FOSTER (cf. 13.2% for the Penn Treebank test set).

Another mismatch is in the handling of utterances. In our corpus, utterance segmentation emerges from multi-rooted annotations (§2.3). Foster et al. (2011b) manually split each tweet into utterances and treat those as separate instances in their corpus, so that our model trained on often multi-rooted tweets from TRAIN is being tested only on single-rooted utterances.

5.4 Experiment: Which Training Set?

We consider the direct use of TRAIN-PTB instead of TRAIN-NEW. Table 2 reports the results on both test sets, with various options. "Baseline" is off-the-shelf second-order TurboParser. We consider augmenting it with Brown cluster features (§4.3; "+ Brown") and then also with the parsing adaptations of §4.2 ("+ Brown & PA"). Another choice is whether to modify the POS tags at test time; the modified version ("mod. POS") maps at-mentions to pronoun, and hashtags and URLs to noun.

	Unlabeled Attachment F_1 (%)	
	mod. POS	POS as-is
TEST-NEW		
Baseline	73.0	73.5
+ Brown	73.7	73.3
+ Brown & PA	72.9	73.1
TEST-FOSTER		
Baseline	76.3	75.2
+ Brown	75.5	76.7
+ Brown & PA	76.9	77.0

Table 2: Performance of second-order TurboParser trained on TRAIN-PTB, with various preprocessing options. The main parser (§5.3) achieves 80.9% and 76.1% on the two test sets, respectively; see §5.4 for discussion.

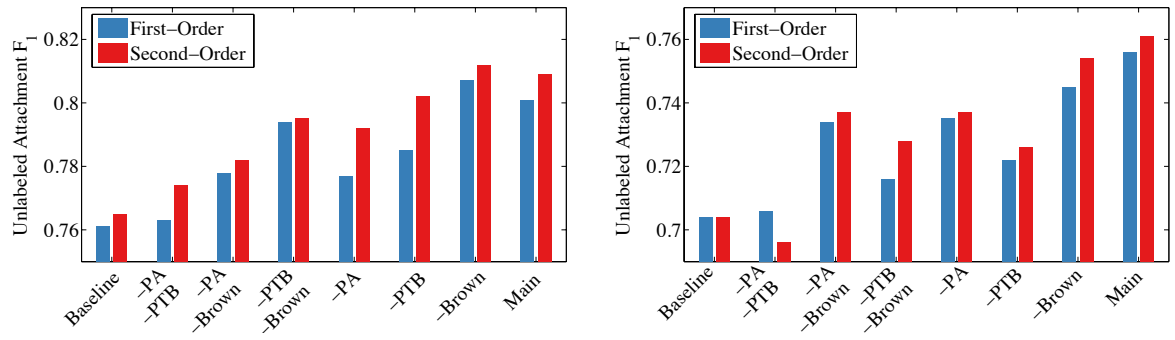
We note that comparing these scores to our main parser (§5.3) conflates three very important independent variables: the amount of training data (39,832 Penn Treebank sentences vs. 1,473 Twitter utterances), the annotation method, and the source of the data. However, we are encouraged that, on what we believe is the superior test set (TEST-NEW), our overall approach obtains a 7.8% gain with an order of magnitude less annotated data.

5.5 Experiment: Effect of Preprocessing

Table 3 (second block, italicized) shows the performance of the main parser on both test sets with gold-standard and automatic POS tagging and token selection. On TEST-NEW, with either gold-standard POS tags or gold-standard token selection, performance increases by 1.1%; with both, it increases by 2.3%. On TEST-FOSTER, token selection matters much less, but POS tagging accounts for a drop of more than 6%. This is consistent with Foster et al.'s finding: using a fine-grained Penn Treebank–trained POS tagger (achieving around 84% accuracy on Twitter), they saw 5–8% improvement in unlabeled dependency attachment accuracy using gold-standard POS tags.

5.6 Experiment: Ablations

We ablated each key element of our main parser—PTB features, Brown features, second order features and decoding, and the parsing adaptations of



(a) TEST-NEW (b) TEST-FOSTER

Figure 3: Feature ablations; these charts present the same scores shown in Table 3 and more variants of the first-order model.

	Unlabeled Attachment F_1 (%)	
	TEST-NEW	**TEST-FOSTER**
Main parser	80.9	**76.1**
Gold POS and TS	*83.2*	*82.8*
Gold POS, automatic TS	*82.0*	*82.3*
Automatic POS, gold TS	*82.0*	*76.2*
Single ablations:		
– PTB	80.2	72.6
– Brown	**81.2**	75.4
– 2nd order	80.1	75.6
– PA	79.2	73.7
Double ablations:		
– PTB, – Brown	79.5	72.8
– PTB, – 2nd order	78.5	72.2
– PTB, – PA	77.4	69.6
– Brown, – 2nd order	80.7	74.5
– Brown, – PA	78.2	73.7
– 2nd order, – PA	77.7	73.5
Baselines:		
Second order	76.5	70.4
First order	76.1	70.4

Table 3: Effects of gold-standard POS tagging and token selection (TS; §5.5) and of feature ablation (§5.6). The "baselines" are TurboParser without the parsing adaptations in §4.2 and without Penn Treebank or Brown features. The best result in each column is bolded. See also Figure 3.

§4.2—as well as each pair of these. These conditions use automatic POS tags and token selection. The "– PA" condition, which ablates parsing adaptations, is accomplished by deleting punctuation (in training and test data) and parsing using TurboParser's existing algorithm.

Results are shown in Table 3. Further results with first- and second-order TurboParsers are plotted in Figure 3. Notably, a 2–3% gain is obtained by modifying the parsing algorithm, and our stacking-inspired use of Penn Treebank data contributes in both cases, quite a lot on TEST-FOSTER (unsurprisingly given that test set's similarity to the Penn Treebank). More surprisingly, we find that Brown cluster features do not consistently improve performance, at least not as instantiated here, with our small training set.

6 Conclusion

We described TWEEBOPARSER, a dependency parser for English tweets that achieves over 80% unlabeled attachment score on a new, high-quality test set. This is on par with state-of-the-art reported results for news text in Turkish (77.6%; Koo et al., 2010) and Arabic (81.1%; Martins et al., 2011). Our contributions include important steps taken to build the parser: a consideration of the challenges of parsing tweets that informed our annotation process, the resulting new TWEEBANK corpus, adaptations to a statistical parsing algorithm, a new approach to exploiting data in a better-resourced domain (the Penn Treebank), and experimental analysis of the decisions we made. The dataset and parser can be found at `http://www.ark.cs.cmu.edu/TweetNLP`.

Acknowledgments

The authors thank the anonymous reviewers and André Martins, Yanchuan Sim, Wang Ling, Michael Mordowanec, and Alexander Rush for helpful feedback, as well as the annotators Waleed Ammar, Jason Baldridge, David Bamman, Dallas Card, Shay Cohen, Jesse Dodge, Jeffrey Flanigan, Dan Garrette, Lori Levin, Wang Ling, Bill McDowell, Michael Mordowanec, Brendan O'Connor, Rohan Ramanath, Yanchuan Sim, Liang Sun, Sam Thomson, and Dani Yogatama. This research was supported in part by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533 and by NSF grants IIS-1054319 and IIS-1352440.

References

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

Timonthy Baldwin and Su Nam Kim. 2010. Multiword expressions. In *Handbook of Natural Language Processing, Second Edition*. CRC Press, Taylor and Francis Group.

Adam Bermingham and Alan F. Smeaton. 2010. Classifying sentiment in microblogs: Is brevity an advantage? In *Proc. of CIKM*.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English Web Treebank. Technical Report LDC2012T13, Linguistic Data Consortium. URL `http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2012T13`.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

Marie Candito and Matthieu Constant. 2014. Strategies for contiguous multiword expression analysis and dependency parsing. In *Proc. of ACL*.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proc. of EMNLP-CoNLL*.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: theory and experiments with perceptron algorithms. In *Proc. of EMNLP*.

Matthieu Constant and Anthony Sigogne. 2011. MWU-aware part-of-speech tagging with a CRF model and lexical resources. In *Proc. of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*.

Matthieu Constant, Anthony Sigogne, and Patrick Watrin. 2012. Discriminative strategies to integrate multiword expression recognition and parsing. In *Proc. of ACL*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Proc. of COLING Workshop on Cross-Framework and Cross-Domain Parser Evaluation*.

Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, Joao Graca, and Fernando Pereira. 2007. Frustratingly hard domain adaptation for dependency parsing. In *Proc. of EMNLP-CoNLL*.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(233-240):160.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proc. of NAACL-HLT*.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*.

Jakob Elming, Anders Johannsen, Sigrid Klerke, Emanuele Lapponi, Héctor Martínez Alonso, and Anders Søgaard. 2013. Down-stream effects of tree-to-dependency conversions. In *Proc. of NAACL-HLT*.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proc of ACL-HLT*.

Jennifer Foster, Özlem Çetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011a. #hardtoparse: POS tagging and parsing the Twitterverse. In *Proc. of AAAI Workshop on Analyzing Microtext*.

Jennifer Foster, Özlem Çetinoğlu, Joachim Wagner, Joseph Le Roux, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011b. From news to comment: resources and benchmarks for parsing the language of Web 2.0. In *Proc. of IJCNLP*.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *Proc. of ACL-HLT*.

Spence Green, Marie-Catherine de Marneffe, and Christopher D. Manning. 2012. Parsing models for identifying multiword expressions. *Computational Linguistics*, 39(1):195–227.

Stephan Greene and Philip Resnik. 2009. Syntactic packaging and implicit sentiment. In *Proc. of NAACL*.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Prague Czech-English Dependency Treebank 2.0. Technical Report LDC2012T08, Linguistic Data Consortium. URL `http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2012T08`.

Rebecca Hwa. 2001. *Learning Probabilistic Lexicalized Grammars for Natural Language Processing*. Ph.D. thesis, Harvard University.

Richard Johansson. 2013. Training parsers on incompatible treebanks. In *Proc. of NAACL-HLT*.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Lingpeng Kong and Noah A. Smith. 2014. An empirical comparison of parsing methods for Stanford dependencies. ArXiv:1404.4314.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. of ACL*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proc. of ACL*.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proc. of EMNLP*.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the OMG! In *Proc. of ICWSM*.

Joseph Le Roux, Matthieu Constant, and Antoine Rozenknop. 2014. Syntactic parsing and compound recognition via dual decomposition: application to French. In *Proc. of COLING*.

Ji Ma, Yue Zhang, and Jingbo Zhu. 2014. Punctuation processing for projective dependency parsing. In *Proc. of ACL*.

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

André F.T. Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of ACL*.

André F.T. Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proc. of EMNLP*.

André F.T. Martins, Noah A. Smith, Pedro M.Q. Aguiar, and Mário A.T. Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proc. of EMNLP*.

André F.T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proc. of ACL-IJCNLP*.

André F.T. Martins, Noah A. Smith, Eric P. Xing, Pedro M.Q. Aguiar, and Mário A.T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proc. of EMNLP*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. of ACL*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proc. of IWPT*.

Michael T. Mordowanec, Nathan Schneider, Chris. Dyer, and Noah A. Smith. 2014. Simplified dependency annotations with GFL-Web. In *Proc. of ACL, demonstration track*.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proc. of ACL-HLT*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proc. of NAACL-HLT*.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proc. of ACL*.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 Shared Task on Parsing the Web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language*.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proc. of ANLP*.

Sebastian Riedel and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proc. of EMNLP*.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: an experimental study. In *Proc. of EMNLP*.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proc. of CICLing*.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014. Discriminative lexical semantic segmentation with gaps: Running the MWE gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206.

Nathan Schneider, Brendan O'Connor, Naomi Saphra, David Bamman, Manaal Faruqui, Noah A. Smith, Chris Dyer, and Jason Baldridge. 2013. A framework for (under)specifying dependency syntax without overloading annotators. In *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

David A. Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

Sandeep Soni, Tanushree Mitra, Eric Gilbert, and Jacob Eisenstein. 2014. Modeling factuality judgments in social media text. In *Proc. of ACL*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proc. of ACL*.

David Vadas and James Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Proc. of ACL*.

Eva Maria Vecchi, Roberto Zamparelli, and Marco Baroni. 2013. Studying the recursive behaviour of adjectival modification with compositional distributional semantics. In *Proc. of EMNLP*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*.

Greed is Good if Randomized: New Inference for Dependency Parsing

Yuan Zhang[*], **Tao Lei**[*], **Regina Barzilay, and Tommi Jaakkola**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{yuanzh, taolei, regina, tommi}@csail.mit.edu

Abstract

Dependency parsing with high-order features results in a provably hard decoding problem. A lot of work has gone into developing powerful optimization methods for solving these combinatorial problems. In contrast, we explore, analyze, and demonstrate that a substantially simpler randomized greedy inference algorithm already suffices for near optimal parsing: a) we analytically quantify the number of local optima that the greedy method has to overcome in the context of first-order parsing; b) we show that, as a decoding algorithm, the greedy method surpasses dual decomposition in second-order parsing; c) we empirically demonstrate that our approach with up to third-order and global features outperforms the state-of-the-art dual decomposition and MCMC sampling methods when evaluated on 14 languages of non-projective CoNLL datasets.[1]

1 Introduction

Dependency parsing is typically guided by parameterized scoring functions that involve rich features exerting refined control over the choice of parse trees. As a consequence, finding the highest scoring parse tree is a provably hard combinatorial inference problem (McDonald and Pereira, 2006). Much of the recent work on parsing has focused on solving these problems using powerful optimization techniques. In this paper, we follow a different strategy, arguing that a much simpler inference strategy suffices. In fact, we demonstrate that a randomized greedy method of inference surpasses the state-of-the-art performance in dependency parsing.

Our choice of a randomized greedy algorithm for parsing follows from a successful track record of such methods in other hard combinatorial problems. These conceptually simple and intuitive algorithms have delivered competitive approximations across a broad class of NP-hard problems ranging from set cover (Hochbaum, 1982) to MAX-SAT (Resende et al., 1997). Their success is predicated on the observation that most realizations of problems are much easier to solve than the worst-cases. A simpler algorithm will therefore suffice in typical cases. Evidence is accumulating that parsing problems may exhibit similar properties. For instance, methods such as dual decomposition offer certificates of optimality when the highest scoring tree is found. Across languages, dual decomposition has shown to lead to a certificate of optimality for the vast majority of the sentences (Koo et al., 2010; Martins et al., 2011). These remarkable results suggest that, as a combinatorial problem, parsing appears simpler than its broader complexity class would suggest. Indeed, we show that a simpler inference algorithm already suffices for superior results.

In this paper, we introduce a randomized greedy algorithm that can be easily used with any rich scoring function. Starting with an initial tree drawn uniformly at random, the algorithm makes only local myopic changes to the parse tree in an attempt to climb the objective function. While a single run of the hill-climbing algorithm may indeed get stuck in a locally optimal solution, multiple random restarts can help to overcome this problem. The same algorithm is used both for learning the parameters of the scoring function as well as for parsing test sentences.

The success of a randomized greedy algorithm is tied to the number of local maxima in the search space. When the number is small, only a few restarts will suffice for the greedy algorithm to find the highest scoring parse. We provide an al-

[*] Both authors contributed equally.

[1] Our code is available at https://github.com/taolei87/RBGParser.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1013–1024,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

gorithm for explicitly counting the number of local optima in the context of first-order parsing, and demonstrate that the number is typically quite small. Indeed, we find that a first-order parser trained with exact inference or using our randomized greedy algorithm delivers basically the same performance.

We hypothesize that parsing with high-order scoring functions exhibits similar properties. The main rationale is that, even in the presence of high-order features, the resulting scoring function remains first-order dominant. The performance of a simple arc-factored first-order parser is only a few percentage points behind higher-order parsers. The higher-order features in the scoring function offer additional refinement but only a few changes above and beyond the first-order result. As a consequence, most of the arc choices are already determined by a much simpler, polynomial time parser.

We use dual decomposition to show that the greedy method indeed succeeds as an inference algorithm even with higher-order scoring functions. In fact, with second-order features, regardless of which method was used for training, the randomized greedy method outperforms dual decomposition by finding higher scoring trees. For the sentences that dual decomposition is optimal (obtains a certificate), the greedy method finds the same solution in over 99% of the cases. Our simple inference algorithm is therefore likely to scale to higher-order parsing and we demonstrate empirically that this is indeed so.

We validate our claim by evaluating the method on the CoNLL dependency benchmark that comprises treebanks from 14 languages. Averaged across all languages, our method outperforms state-of-the-art parsers, including TurboParser (Martins et al., 2013) and our earlier sampling-based parser (Zhang et al., 2014). On seven languages, we report the best published results. The method is not sensitive to initialization. In fact, drawing the initial tree uniformly at random results in the same performance as when initialized from a trained first-order distribution. In contrast, sufficient randomization of the starting point is critical. Only a small number of restarts suffices for finding (near) optimal parse trees.

2 Related Work

Finding Optimal Structure in Parsing The use of rich-scoring functions in dependency parsing inevitably leads to the challenging combinatorial problem of finding the maximizing parse. In fact, McDonald and Pereira (2006) demonstrated that the task is provably NP-hard for non-projective second-order parsing. Not surprisingly, approximate inference has been at the center of parsing research. Examples of these approaches include easy-first parsing (Goldberg and Elhadad, 2010), inexact search (Johansson and Nugues, 2007; Zhang and Clark, 2008; Huang et al., 2012; Zhang et al., 2013), partial dynamic programming (Huang and Sagae, 2010) and dual decomposition (Koo et al., 2010; Martins et al., 2011).

Our work is most closely related to the MCMC sampling-based approaches (Nakagawa, 2007; Zhang et al., 2014). In our earlier work, we developed a method that learns to take guided stochastic steps towards a high-scoring parse (Zhang et al., 2014). In the heart of that technique are sophisticated samplers for traversing the space of trees. In this paper, we demonstrate that a substantially simpler approach that starts from a tree drawn from the uniform distribution and uses hill-climbing for parameter updates achieves similar or higher performance.

Another related greedy inference method has been used for non-projective dependency parsing (McDonald and Pereira, 2006). This method relies on hill-climbing to convert the highest scoring projective tree into its non-projective approximation. Our experiments demonstrate that when hill-climbing is employed as a primary learning mechanism for high-order parsing, it exhibits different properties: the distribution for initialization does not play a major role in the final outcome, while the use of restarts contributes significantly to the quality of the resulting tree.

Greedy Approximations for NP-hard Problems There is an expansive body of research on greedy approximations for NP-hard problems. Examples of NP-hard problems with successful greedy approximations include the traveling saleman problem problem (Held and Karp, 1970; Rego et al., 2011), the MAX-SAT problem (Mitchell et al., 1992; Resende et al., 1997) and vertex cover (Hochbaum, 1982). While some greedy methods have poor worst-case complexity, many

of them work remarkably well in practice. Despite the apparent simplicity of these algorithms, understanding their properties is challenging: often their "theoretical analyses are negative and inconclusive" (Amenta and Ziegler, 1999; Spielman and Teng, 2001). Identifying conditions under which approximations are provably optimal is an active area of research in computer science theory (Dumitrescu and Tóth, 2013; Jonsson et al., 2013).

In NLP, randomized and greedy approximations have been successfully used across multiple applications, including machine translation and language modeling (Brown et al., 1993; Ravi and Knight, 2010; Daumé III et al., 2009; Moore and Quirk, 2008; Deoras et al., 2011). In this paper, we study the properties of these approximations in the context of dependency parsing.

3 Method

3.1 Preliminaries

Let x be a sentence and $\mathcal{T}(x)$ be the set of possible dependency trees over the words in x. We use $y \in \mathcal{T}(x)$ to denote a dependency tree for x, and $y(m)$ to specify the head (parent) of the modifier word indexed by m in tree y. We also use m to denote the indexed word when there is no ambiguity. In addition, we define $\mathcal{T}(y, m)$ as the set of "neighboring trees" of y obtained by changing only the head of the modifier, i.e. $y(m)$.

The dependency trees are scored according to $S(x, y) = \theta \cdot \phi(x, y)$, where θ is a vector of parameters and $\phi(x, y)$ is a sparse feature vector representation of tree y for sentence x. In this work, $\phi(x, y)$ will include up to third-order features as well as a range of global features commonly used in re-ranking methods (Collins, 2000; Charniak and Johnson, 2005; Huang, 2008).

The parameters θ in the scoring function are estimated on the basis of a training set $D = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^{N}$ of sentences $\hat{x}_i$ and the corresponding gold (target) trees $\hat{y}_i$. We adopt a max-margin framework for this learning problem. Specifically, we aim to find parameter values that score the gold target trees higher than others:

$$\forall i \in \{1, \cdots, N\}, y \in \mathcal{T}(\hat{x}_i),$$
$$S(\hat{x}_i, \hat{y}_i) \geq S(\hat{x}_i, y) + \|\hat{y}_i - y\|_1 - \xi_i$$

where $\xi_i \geq 0$ is the slack variable (non-zero values are penalized against) and $\|\hat{y}_i - y\|_1$ is the hamming distance between the gold tree $\hat{y}_i$ and a candidate parse y.

In an online learning setup, parameters are updated successively after each sentence. Each update still requires us to find the "strongest violation", i.e., a candidate tree $\tilde{y}$ that scores higher than the gold tree $\hat{y}_i$:

$$\tilde{y} = \arg\max_{y \in \mathcal{T}(\hat{x}_i)} \{S(\hat{x}_i, y) + \|y - \hat{y}_i\|_1\}$$

The parameters are then revised so as to select against the offending $\tilde{y}$. Instead of a standard parameter update based on $\tilde{y}$ as in perceptron, stochastic gradient descent, or passive-aggressive updates, our implementation follows Lei et al. (2014) where the first-order parameters are broken up into a tensor. Each tensor component is updated successively in combination with the parameters corresponding to MST features (McDonald et al., 2005) and higher-order features (when included).[2]

3.2 Algorithm

During training and testing, the key combinatorial problem we must solve is that of decoding, i.e., finding the highest scoring tree $\tilde{y} \in \mathcal{T}(x)$ for each sentence x (or $\hat{x}_i$). In our notation,

$$\tilde{y} = \arg\max_{y \in \mathcal{T}(\hat{x}_i)} \{\theta \cdot \phi(\hat{x}_i, y) + \|y - \hat{y}_i\|_1\} \quad \text{(train)}$$

$$\tilde{y} = \arg\max_{y \in \mathcal{T}(x)} \{\theta \cdot \phi(x, y)\} \quad \text{(test)}$$

While the decoding problem with feature sets similar to ours has been shown to be NP-hard, many approximation algorithms work remarkably well. We commence with a motivating example.

Locality and Parsing One possible reason for why greedy or other approximation algorithms work well for dependency parsing is that typical sentences and therefore the learned scoring functions $S(x, y) = \theta \cdot \phi(x, y)$ are primarily "local". By this we mean that head-modifier decisions could be made largely without considering the surrounding structure (the context). For example, in English an adjective and a determiner are typically attached to the following noun.

We demonstrate the degree of locality in dependency parsing by comparing a first-order tree-based parser to the parser that predicts each head word independently of others. Note that the independent prediction of dependency arcs does not necessarily give rise to a tree. The parameters of

[2] We refer the readers to Lei et al. (2014) for more details about the tensor scoring function and the online update.

Dataset	Indp. Pred	Tree Pred
Slovene	83.7	84.2
Arabic	79.0	79.2
Japanese	93.4	93.7
English	91.6	91.9
Average	86.9	87.3

Table 1: Head attachment accuracy of a first-order local classifier (left) and a first-order structural prediction model (right). The two types of models are trained using the same set of features.

Input: parameter θ, sentence x
Output: dependency tree $\tilde{y}$

Randomly initialize tree $y^{(0)}$;
$t = 0$;
repeat
  list = bottom-up node list of $y^{(t)}$;
  for each word m in list **do**
    $y^{(t+1)} = \arg\max_{y \in \mathcal{T}(y^{(t)}, m)} S(x, y)$;
    $t = t + 1$;
  end for
until no change in this iteration
return $\tilde{y} = y^{(t)}$;

Figure 1: A randomized hill-climbing algorithm for dependency parsing.

the two parsers, the independent prediction and a tree-based parser, are trained separately with the corresponding decoding algorithm but with the same feature set.

Table 1 shows that the accuracy of the independent prediction ranges from 79% to 93% on four CoNLL datasets. The results are on par with the first-order structured prediction model. This experiment reinforces the conclusion in Liang et al. (2008), where a local classifier was shown to achieve comparable accuracy to a sequential model (e.g. CRF) in POS tagging and named-entity recognition.

Hill-Climbing with Random Restarts We build here on the motivating example and explore greedy algorithms as generalizations of purely local decoding. Greedy algorithms break the decoding problem into a sequence of simple local steps, each required to improve the solution. In our case, simple local steps correspond to choosing the head for each modifier word.

We begin with a tree $y^{(0)}$, which can be a sample drawn uniformly from $\mathcal{T}(x)$ (Wilson, 1996). Our greedy algorithm then updates $y^{(t)}$ to a better tree $y^{(t+1)}$ by revising the head of one modifier word while maintaining the constraint that the resulting structure is a tree. The modifiers are considered in the bottom-up order relative to the current tree (the word furthest from the root is considered first). We provide an analysis to motivate this bottom-up update strategy in Section 4.1. The algorithm continues until the score can no longer be improved by changing the head of a single word. The resulting tree represents a locally optimal prediction relative to a single-arc greedy algorithm. Figure 1 gives the algorithm in pseudo-code.

There are many possible variations of the simple randomized greedy hill-climbing algorithm. First, the Wilson sampling algorithm (Wilson, 1996) can be naturally extended to obtain i.i.d. samples from any first-order distributions. Therefore, we could initialize the tree $y^{(0)}$ with a tree from a first-order parser, or draw the initial tree from a first-order distribution other than uniform. However, perhaps surprisingly, as we demonstrate later, little is lost with uniform initialization. Second, since a single run of randomized hill-climbing is relatively cheap and runs are independent to each other, it is easy to execute multiple runs independently in parallel. The final predicted tree is then simply the highest scoring tree across the multiple runs. We demonstrate that only a small number of parallel runs are necessary for near optimal prediction.

4 Analysis

4.1 First-Order Parsing

We provide here a firmer basis for why the randomized greedy algorithm can be expected to work. While the focus of the rest of the paper is on higher-order parsing, we limit ourselves in this subsection to first-order parsing. The reasons for this are threefold. First, a simple greedy algorithm is already not guaranteed a priori to work in the context of a first-order scoring function. The conclusions from this analysis are therefore likely to carry over to higher-order parsing scenarios as well. Second, a first-order arc-factored scoring provides us an easy way to ascertain when the randomized greedy algorithm indeed found the highest scoring tree. Finally, we are able to count the

Dataset	Average Len.	# of local optima at percentile			fraction of finding global optima (%)	
		50%	70%	90%	0 <Len.≤ 15	Len.> 15
Turkish	12.1	1	1	2	100	100
Slovene	15.9	2	20	3647	100	98.1
English	24.0	21	121	2443	100	99.3
Arabic	36.8	2	35	>10000	100	99.1

Table 2: The left part of the table shows the local optimum statistics of the first-order model. The sentences are sorted by the number of local optima. Columns 3 to 5 show the number of local optima of a sentence at different percentile of the sorted list. For example, on English 50% of the sentences have no more than 21 local optimum trees. The right part shows the fraction of finding global optima using 300 uniform restarts for each sentence.

number of locally optimal solutions for a greedy algorithm in the context of first-order parsing and can therefore relate this property to the success rates of the algorithm.

Reachability We begin by highlighting a basic property of trees, namely that single arc changes suffice for transforming any tree to any other tree in a small number of steps while maintaining that each intermediate structure is also a tree. In this sense, a target tree is reachable from any starting point using only single arc changes. More formally, let y be any starting tree and y' the desired target. Let $m_1, m_2, \cdots, m_n$ be the bottom-up list of words (modifiers) corresponding to tree y, where m_1 is the word furthest from the root. We can simply change each head $y(m_i)$ to that of $y'(m_i)$ in this order $i = 1, \ldots, n$. The bottom-up order guarantees that no cycle is introduced with respect to the remaining (yet unmodified) nodes of y. The fact that y' is a valid tree implies no cycle will appear with respect to the already modified nodes.

Note that, according to this property, any tree is reachable from any starting point using only k modifications, where k is the number of head differences, i.e. $k = |\{m : y(m) \neq y'(m)\}|$. The result also suggests that it may be helpful to perform the greedy steps in the bottom-up order, a suggestion that we follow in our implementation.

Broadly speaking, we have established that the greedy algorithm is not inherently limited by virtue of its basic steps. Of course, it is a different question whether the scoring function supports such local changes towards the correct target tree.

Locally Optimal Trees While greedy algorithms are notoriously prone to getting stuck in locally optimal solutions, we establish here that decoding with learned scoring functions involves only a small number of local optima. In our case, a local optimum corresponds to a tree y where no single change of head $y(m)$ results in a higher scoring tree. Clearly, the highest scoring tree is also a local optimum in this sense. If there were many such local optima, finding the one with the highest score would be challenging for a greedy algorithm, even with randomization.

We begin with a worst case analysis and estab-

Function **CountOptima**($G = \langle V, E \rangle$)
$V = \{w_0, w_1, \cdots, w_n\}$ where w_0 is the root
$E = \{e_{ij} \in \mathbb{R}\}$ are the arc scores
Return: the number of local optima

1: Let $y(0) = \emptyset$ and $y(i) = \arg\max_j e_{ji}$;
2: **if** y is a tree (no cycle) **then return** 1;
3: Find a cycle $C \subset V$ in y;
4: count $= 0$;
// contract the cycle
5: create a vertex w_*;
6: $\forall j \notin C : e_{*j} = \max_{k \in C} e_{kj}$;
7: **for** each vertex $w_i \in C$ **do**
8: $\quad \forall j \notin C : e_{j*} = e_{ji}$;
9: $\quad V' = V \cup \{w_*\} \setminus C$;
10: $\quad E' = E \cup \{e_{*j}, e_{j*} \mid \forall j \notin C\}$
11: $\quad$ count += CountOptima($G' = \langle V', E' \rangle$);
12: **end for**
13: **return** count;

Figure 2: A recursive algorithm for counting local optima for a sentence with words $w_1, \cdots, w_n$ (first-order parsing). The algorithm resembles the Chu-Liu-Edmonds algorithm for finding the maximum directed spanning tree (Chu and Liu, 1965).

Dataset	Trained with Hill-Climbing (HC)				Trained with Dual Decomposition (DD)			
	%Cert (DD)	$s_{DD} > s_{HC}$	$s_{DD} = s_{HC}$	$s_{DD} < s_{HC}$	%Cert (DD)	$s_{DD} > s_{HC}$	$s_{DD} = s_{HC}$	$s_{DD} < s_{HC}$
Turkish	98.7	0.0	99.8	0.2	98.7	0.0	100.0	0.0
Slovene	94.5	0.0	98.7	1.3	92.3	0.2	99.0	0.8
English	94.5	0.3	98.7	1.0	94.6	0.5	98.7	0.8
Arabic	78.8	3.4	93.9	2.7	75.3	4.7	88.4	6.9

Table 3: Decoding quality comparison between hill-climbing (HC) and dual decomposition (DD). Models are trained either with HC (left) or DD (right). s_{HC} denotes the score of the tree retrieved by HC and s_{DD} gives the analogous score for DD. The columns show the percentage of all test sentences for which one method succeeds in finding a higher or the same score. "Cert" column gives the percentage of sentences for which DD finds a certificate.

lish a tight upper bound on the number of local optima for a first-order scoring function.

Theorem 1 *For any first-order scoring function that factorizes into the sum of arc scores* $S(x, y) = \sum S_{arc}(y(m), m)$*: (a) the number of locally optimal trees is at most* 2^{n-1} *for* n *words; (b) this upper bound is tight.*[3]

While the number of possible dependency trees is $(n + 1)^{n-1}$ (Cayley's formula), the number of local optima is at most 2^{n-1}. This is still too many for longer sentences, suggesting that, in the worst case, a randomized greedy algorithm is unlikely to find the highest scoring tree. However, the scoring functions we learn for dependency parsing are considerably easier.

Average Case Analysis In contrast to the worst-case analysis above, we will count here the actual number of local optima *per sentence* for a first-order scoring function learned from data with the randomized greedy algorithm. Figure 2 provides pseudo-code for our counting algorithm. The algorithm is derived by tailoring the proof of Theorem 1 to each sentence.

Table 2 shows the empirical number of locally optimal trees estimated by our algorithm across 4 different languages. Decoding with trained scoring functions in the average case is clearly substantially easier than the worst case. For example, on the English test set more than 70% of the sentences have at most 121 locally optimal trees. Since the average sentence length is 24, the discrepancy between the typical number (e.g., 121) and the worst case (2^{24-1}) is substantial. As a result, only a small number of restarts is likely to suffice for finding optimal trees in practice.

Optimal Decoding We can easily verify whether the randomized greedy algorithm indeed succeeds in finding the highest scoring trees with a learned first-order scoring function. We have established above that there are typically only a small number of locally optimal trees. We would therefore expect the algorithm to work. We show the results in the second part of Table 2. For short sentences of length up to 15, our method finds the global optimum for all the test sentences. Success rates remain high even for longer test sentences.

4.2 Higher-Order Parsing

Exact decoding with high-order features is known to be provably hard (McDonald et al., 2005). We begin our analysis here with a second-order (sibling/grandparent) model, and compare our randomized hill-climbing (HC) method to dual decomposition (DD), re-implementing Koo et al. (2010). Table 3 compares decoding quality for the two methods across four languages. Overall, in 97.8% of the sentences, HC obtains the same score as DD, in 1.3% of the cases HC finds a higher scoring tree, and in 0.9% of cases DD results in a better tree. The results follow the same pattern regardless of which method was used to train the scoring function. The average rate of certificates for DD was 92%. In over 99% of these sentences, HC reaches the same optimum.

We expect that these observations about the success of HC carry over to other high-order parsing models for several reasons. First, a large number of arcs are pruned in the initial stage, considerably reducing the search space and minimizing the number of possible locally optimal trees. Second, many dependencies can be determined already with independent arc prediction (see our motivating example above), predictions that are readily achieved with a greedy algorithm. Finally, high-order features represent smaller refinements, i.e., suggest only a few changes above and beyond the dominant first-order scores. Greedy al-

[3] A proof sketch is given in Appendix.

gorithms are therefore likely to be able to leverage at least some of this potential. We demonstrate below that this is indeed so.

Our methods are trained within the max-margin framework. As a result, we are expected to find the highest scoring competing tree for each training sentence (the "strongest violation"). One may question therefore whether possible sub-optimal decoding for some training sentences (finding "a violation" rather than the "strongest violation") impacts the learned parser. To this end, Huang et al. (2012) have established that weaker violations do suffice for separable training sets.

5 Experimental Setup

Dataset and Evaluation Measures We evaluate our model on CoNLL dependency treebanks for 14 different languages (Buchholz and Marsi, 2006; Surdeanu et al., 2008), using standard training and testing splits. We use part-of-speech tags and the morphological information provided in the corpus. Following standard practice, we use Unlabeled Attachment Score (UAS) excluding punctuation (Koo et al., 2010; Martins et al., 2013) as the evaluation metric in all our experiments.

Baselines We compare our model with the TurboParser (Martins et al., 2013) and our earlier sampling-based parser (Zhang et al., 2014). For both parsers, we directly compare with the recent published results on the CoNLL datasets. We also compare our parser against the best published results for the individual languages in our datasets. This comparison set includes four additional parsers: Martins et al. (2011), Koo et al. (2010), Zhang et al. (2013) and our tensor-based parser (Lei et al., 2014).

Features We use the same feature templates as in our prior work (Zhang et al., 2014; Lei et al., 2014)[4]. Figure 3 shows the first- to third-order feature templates that we use in our model. For the global features we use right-branching, coordination, PP attachment, span length, neighbors, valency and non-projective arcs features.

Implementation Details Following standard practices, we train our model using the passive-aggressive online learning algorithm (MIRA) and parameter averaging (Crammer et al., 2006; Collins, 2002). By default we use an adaptive strategy for running the hill-climbing algorithm – for a given sentence we repeatedly run the algorithm in parallel[5] until the best tree does not change for $K = 300$ consecutive restarts. For each restart, by default we initialize the tree $y^{(0)}$ by sampling from the first-order distribution using the current learned parameter values (and first-order scores). We train our first-order and third-order model for 10 epochs and our full model for 20 epochs for all languages, and report the average performance across three independent runs.

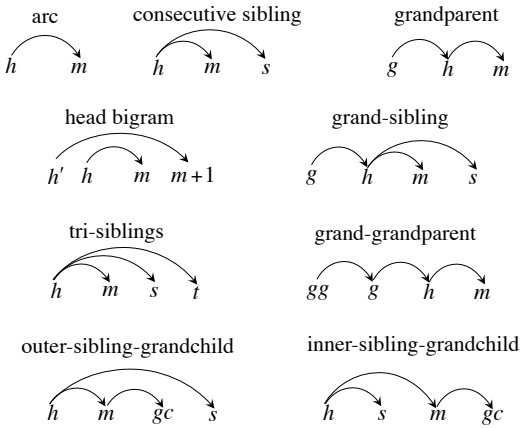


Figure 3: First- to third-order features.

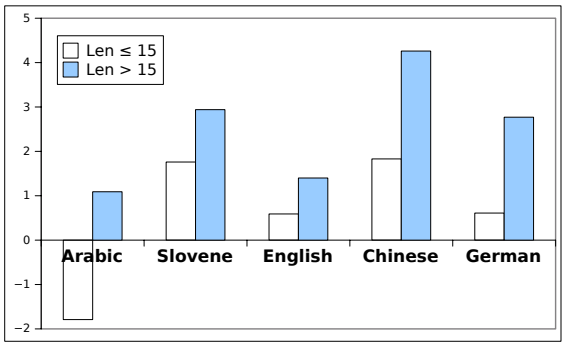


Figure 4: Absolute UAS improvement of our full model over the first-order model. Sentences in the test set are divided into 2 groups based on their lengths.

6 Results

Comparison with the Baselines Table 4 summarizes the results of our model, along with the state-of-the-art baselines. On average across 14 languages, our full model with the tensor component outperforms both TurboParser and the sampling-based parser. The direct comparison

[4]We refer the readers to Zhang et al. (2014) and Lei et al. (2014) for the detailed definition of each feature template.

[5]We use 8 threads in all the experiments.

	Our Model				Exact 1st	Turbo (MA13)	Sampling (ZL14)	Best Published
	1st	3rd	$\text{Full}_{\text{w/o tensor}}$	Full				
Arabic	78.98	79.95	79.38	80.24	79.22	79.64	80.12	81.12 (MS11)
Bulgarian	92.15	93.38	93.69	93.72	92.24	93.10	93.30	94.02 (ZH13)
Chinese	91.20	93.00	92.76	**93.04**	91.17	89.98	92.63	92.68 (LX14)
Czech	87.65	90.11	90.34	90.77	87.82	90.32	91.04	91.04 (ZL14)
Danish	90.50	91.43	91.66	91.86	90.56	91.48	91.80	92.00 (ZH13)
Dutch	84.49	86.43	87.04	**87.39**	84.79	86.19	86.47	86.47 (ZL14)
English	91.85	93.01	93.20	**93.25**	91.94	93.22	92.94	93.22 (MA13)
German	90.52	91.91	92.64	**92.67**	90.54	92.41	92.07	92.41 (MA13)
Japanese	93.78	**93.80**	93.35	93.56	93.74	93.52	93.42	93.74 (LX14)
Portuguese	91.12	92.07	92.60	92.36	91.16	92.69	92.41	93.03 (KR10)
Slovene	84.29	86.48	**87.06**	86.72	84.15	86.01	86.82	86.95 (MS11)
Spanish	85.52	87.87	88.17	**88.75**	85.59	85.59	88.24	88.24 (ZL14)
Swedish	89.89	91.17	91.35	91.08	89.78	91.14	90.71	91.62 (ZH13)
Turkish	76.57	76.80	76.13	76.68	76.40	76.90	77.21	77.55 (KR10)
Average	87.75	89.10	89.24	89.44	87.79	88.72	89.23	89.58

Table 4: Results of our model and several state-of-the-art systems. "Best Published UAS" includes the most accurate parsers among Martins et al. (2011), Martins et al. (2013), Koo et al. (2010), Zhang et al. (2013), Lei et al. (2014) and Zhang et al. (2014). For the third-order model, we use the feature set of TurboParser (Martins et al., 2013). The full model combines features of our sampling-based parser (Zhang et al., 2014) and tensor features (Lei et al., 2014).

Dataset	MAP-1st		Uniform		Rnd-1st	
	UAS	Init.	UAS	Init.	UAS	Init.
Slovene	85.2	80.1	86.7	13.7	86.7	34.2
Arabic	78.8	75.1	79.7	12.4	80.2	32.8
English	91.1	82.0	93.3	39.6	93.3	55.6
Chinese	87.2	75.3	93.2	36.8	93.0	54.5
Dutch	84.8	79.5	87.0	26.9	87.4	45.6
Average	85.4	78.4	88.0	25.9	88.1	44.5

Table 5: Comparison between different initialization strategies: (a) MAP-1st: only the MAP tree of the first-order score; (b) Uniform: random trees are sampled from the uniform distribution; and (c) Rnd-1st: random trees are sampled from the first-order distribution. For each method, the table shows the average accuracy of the initial tree and the final parsing accuracy.

with TurboParser is achieved by restricting our model to third order features which still outperforms TurboParser (89.10% vs 88.72%). To compare against the sampling-based parser, we employ our model without the tensor component. The two models achieve a similar average performance (89.24% and 89.23% respectively). Since relative parsing performance depends on a target language, we also include comparison with the best published results. The model achieves the best published results for seven languages.

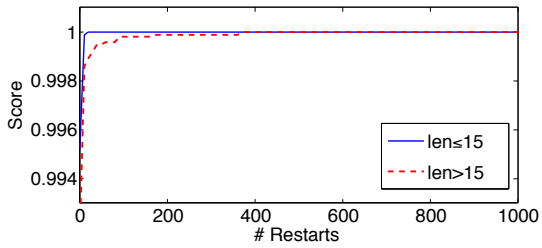
Another noteworthy comparison concerns first-order parsers. As Table 4 shows, the exact and approximate versions of the first-order parser deliver almost identical performance.

Impact of High-Order Features Table 4 shows that the model can effectively utilize high-order features. Comparing the average performance of the model variants, we see that the accuracy on the benchmark languages consistently improves when higher-order features are added. This characteristic of the randomized greedy parser is in line with findings about other state-of-the-art high-order parsers (Martins et al., 2013; Zhang et al., 2014). Figure 4 breaks down these gains based on the sentence length. As expected, on most languages high-order features are particularly helpful when parsing longer sentences.

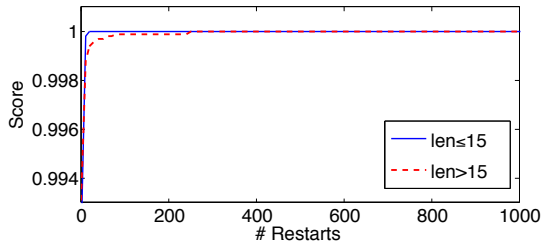
Impact of Initialization and Restarts Table 5 shows the impact of initialization on the model performance for several languages. We consider three strategies: the MAP estimate of the first-order score from the model, uniform sampling and sampling from the first-order distribution. The accuracy of initial trees varies greatly, ranging from 78.4% for the MAP estimate to 25.9% and 44.5% for the latter randomized strategies. However, the resulting parsing accuracy is not determined by the initial accuracy. In fact, the two sampling strategies result in almost identical parsing performance. While the first-order MAP estimate gives the best initial guess, the overall parsing accuracy of this method lags behind. This result demonstrates the importance of restarts – in contrast to the randomized strategies, the MAP initialization performs only a single run of hill-climbing.

	Length ≤ 15	Length > 15
Slovene	100	98.11
English	100	99.12

Table 6: Fractions (%) of the sentences that find the best solution among 3,000 restarts within the first 300 restarts.



(a) Slovene

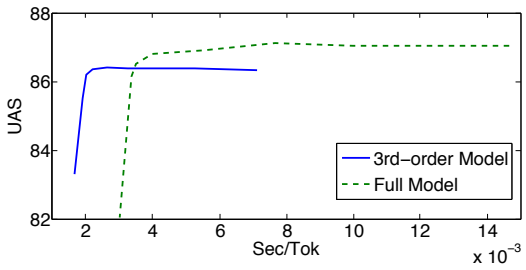


(b) English

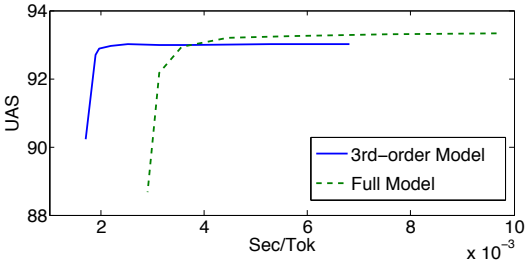
Figure 5: Convergence analysis on Slovene and English datasets. The graph shows the normalized score of the output tree as a function of the number of restarts. The score of each sentence is normalized by the highest score obtained for this sentence after 3,000 restarts. We only show the curves up to 1,000 restarts because they all reach convergence after around 500 restarts.

Convergence Properties Figure 5 shows the score of the trees retrieved by our full model with respect to the number of restarts, for short and long sentences in English and Slovene. To facilitate the comparison, we normalize the score of each sentence by the maximal score obtained for this sentence after 3,000 restarts. Overall, most sentences converge quickly. This view is also supported by Table 6 which shows the fraction of the sentences that converge within the first 300 restarts. We can see that all the short sentences (length up to 15) reach convergence within the allocated restarts. Perhaps surprisingly, more than 98% of the long sentences also converge within 300 restarts.

Decoding Speed As the number of restarts impacts the parsing accuracy, we can trade performance for speed. Figure 6 shows that the model



(a) Slovene



(b) English

Figure 6: Trade-off between performance and speed on Slovene and English datasets. The graph shows the accuracy as a function of decoding speed measured in second per token. Variations in decoding speed is achieved by changing the number of restarts.

achieves high performance with acceptable parsing speed. While various system implementation issues such as programming language and computational platform complicate a direct comparison with other parsing systems, our model delivers parsing time roughly comparable to other state-of-the-art graph-based systems (for example, TurboParser and MST parser) and the sampling-based parser.

7 Conclusions

We have shown that a simple, generally applicable randomized greedy algorithm for inference suffices to deliver state-of-the-art parsing performance. We argued that the effectiveness of such greedy algorithms is contingent on having a small number of local optima in the scoring function. By algorithmically counting the number of locally optimal solutions in the context of first-order parsing, we show that this number is indeed quite small. Moreover, we show that, as a decoding algorithm, the greedy method surpasses dual decomposition in second-order parsing. Finally, we empirically demonstrate that our approach with up to third-order and global features outperforms the state-of-the-art parsers when evaluated on 14 languages of

non-projective CoNLL datasets.

Appendix

We provide here a more detailed justification for the counting algorithm in Figure 2 and, by extension, a proof sketch of Theorem 1. The bullets below follow the operation of the algorithm.

- Whenever independent selection of the heads results in a valid tree, there is only 1 optimum (Lines 1&2 of the algorithm). Otherwise there must be a cycle C in y (Line 3 of the algorithm)
- We claim that any locally optimal tree y' of the graph $G = (V, E)$ must contain $|C| - 1$ arcs of the cycle $C \subseteq V$. This can be shown by contradiction. If y' contains less than $|C| - 1$ arcs of C, then (a) we can construct a tree y'' that contains $|C| - 1$ arcs; (b) the heads in y'' are strictly better than those in y' over the unused part of the cycle; (c) by reachability, there is a path $y' \rightarrow y''$ so y' cannot be a local optimum.
- Any locally optimal tree in G must select an arc in C and reassign it. The rest of the $|C|-1$ arcs will then result in a chain.
- By contracting cycle C we obtain a new graph G' of size $|G| - |C| + 1$ (Lines 5-11 of the algorithm). Easy to verify that (not shown): any local optimum in G' is a local optimum in G and vice versa.

The theorem follows as a corollary of these steps. To see this, let $F(G_m)$ be the number of local optima in the graph of size m:

$$F(G_m) \leq \max_{C \subseteq V(G)} \sum_i F(G^{(i)}_{m-c+1})$$

where $G^{(i)}_{m-c+1}$ is the graph (of size $m - c + 1$) created by selecting the i^{th} arc in cycle C and contracting G_m accordingly, and $c = |C|$ is the size of the cycle. Define $\hat{F}(m)$ as the upper bound of $F(G_m)$ for any graph of size m. By the above formula, we know that

$$\hat{F}(m) \leq \max_{2 \leq c < m} \hat{F}(m - c + 1) \times c$$

By solving for $\hat{F}(m)$ we get $\hat{F}(m) \leq 2^{m-2}$. Since $m = n+1$ for a sentence with n words, the upper-bound of local optima is 2^{n-1}.

To show the tightness, for any $n > 0$, create the graph G_{n+1} with arc scores $e_{ij} = e_{ji} = i$ for any $0 \leq i < j \leq n$. Note that $w_n \rightarrow w_{n-1} \rightarrow w_n$ forms the circle C of size 2, it can be shown by induction on n and $F(G_{n+1})$ that $F(G_{n+1}) = F(G_n) \times 2 = 2^{n-1}$.

Acknowledgments

This research is developed in collaboration with the Arabic Language Technologies (ALT) group at Qatar Computing Research Institute (QCRI) within the IYAS project. The authors acknowledge the support of the U.S. Army Research Office under grant number W911NF-10-1-0533, and of the DARPA BOLT program. We thank the MIT NLP group and the ACL reviewers for their comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Nina Amenta and Günter Ziegler, 1999. *Deformed Products and Maximal Shadows of Polytopes*. Contemporary Mathematics. American Mathematics Society.

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

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 175–182.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural*

Language Processing - Volume 10, EMNLP '02. Association for Computational Linguistics.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Anoop Deoras, Tomáš Mikolov, and Kenneth Church. 2011. A fast re-scoring strategy to capture long distance dependencies. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1116–1127. Association for Computational Linguistics.

Adrian Dumitrescu and Csaba D Tóth. 2013. The traveling salesman problem for lines, balls and planes. In *SODA*, pages 828–843. SIAM.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750. Association for Computational Linguistics.

Michael Held and Richard M. Karp. 1970. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162.

Dorit S. Hochbaum. 1982. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556.

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

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*, pages 586–594.

Richard Johansson and Pierre Nugues. 2007. Incremental dependency parsing using online learning. In *EMNLP-CoNLL*, pages 1134–1138.

Peter Jonsson, Victor Lagerkvist, Gustav Nordh, and Bruno Zanuttini. 2013. Complexity of sat problems, clone theory and the exponential time hypothesis. In *SODA*, pages 1264–1277. SIAM.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proceedings of the 25th international conference on Machine learning*, pages 592–599. ACM.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11. Association for Computational Linguistics.

André F. T. Martins, Miguel B. Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

David Mitchell, Bart Selman, and Hector Levesque. 1992. Hard and easy distributions of sat problems. In *AAAI*, volume 92, pages 459–465. Citeseer.

Robert C. Moore and Chris Quirk. 2008. Random restarts in minimum error rate training for statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 585–592. Association for Computational Linguistics.

Tetsuji Nakagawa. 2007. Multilingual dependency parsing using global features. In *EMNLP-CoNLL*, pages 952–956.

Sujith Ravi and Kevin Knight. 2010. Does giza++ make search errors? *Computational Linguistics*, 36(3):295–302.

César Rego, Dorabela Gamboa, Fred Glover, and Colin Osterman. 2011. Traveling salesman problem heuristics: leading methods, implementations and latest advances. *European Journal of Operational Research*, 211(3):427–441.

Mauricio G. C. Resende, L. S. Pitsoulis, and P. M. Pardalos. 1997. Approximate solution of weighted max-sat problems using grasp. *Satisfiability problems*, 35:393–405.

Daniel Spielman and Shang-Hua Teng. 2001. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 296–305. ACM.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, CoNLL '08. Association for Computational Linguistics.

David B. Wilson. 1996. Generating random spanning trees more quickly than the cover time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 296–303. ACM.

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

Hao Zhang, Liang Zhao, Kai Huang, and Ryan McDonald. 2013. Online learning for inexact hypergraph search. In *Proceedings of EMNLP*.

Yuan Zhang, Tao Lei, Regina Barzilay, Tommi Jaakkola, and Amir Globerson. 2014. Steps to excellence: Simple inference with refined scoring of dependency trees. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

A Unified Model for Word Sense Representation and Disambiguation

Xinxiong Chen, Zhiyuan Liu, Maosong Sun
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology
Tsinghua University, Beijing 100084, China
cxx.thu@gmail.com, {lzy, sms}@tsinghua.edu.cn

Abstract

Most word representation methods assume that each word owns a single semantic vector. This is usually problematic because lexical ambiguity is ubiquitous, which is also the problem to be resolved by word sense disambiguation. In this paper, we present a unified model for joint word sense representation and disambiguation, which will assign distinct representations for each word sense.[1] The basic idea is that both word sense representation (WSR) and word sense disambiguation (WSD) will benefit from each other: (1) high-quality WSR will capture rich information about words and senses, which should be helpful for WSD, and (2) high-quality WSD will provide reliable disambiguated corpora for learning better sense representations. Experimental results show that, our model improves the performance of contextual word similarity compared to existing WSR methods, outperforms state-of-the-art supervised methods on domain-specific WSD, and achieves competitive performance on coarse-grained all-words WSD.

1 Introduction

Word representation aims to build vectors for each word based on its context in a large corpus, usually capturing both semantic and syntactic information of words. These representations can be used as features or inputs, which are widely employed in information retrieval (Manning et al., 2008), document classification (Sebastiani, 2002) and other NLP tasks.

[1] Our sense representations can be downloaded at http://pan.baidu.com/s/1eQcPK8i.

Most word representation methods assume each word owns a single vector. However, this is usually problematic due to the homonymy and polysemy of many words. To remedy the issue, Reisinger and Mooney (2010) proposed a multi-prototype vector space model, where the contexts of each word are first clustered into groups, and then each cluster generates a distinct prototype vector for a word by averaging over all context vectors within the cluster. Huang et al. (2012) followed this idea, but introduced continuous distributed vectors based on probabilistic neural language models for word representations.

These cluster-based models conduct unsupervised word sense induction by clustering word contexts and, thus, suffer from the following issues:

- It is usually difficult for these cluster-based models to determine the number of clusters. Huang et al. (2012) simply cluster word contexts into static K clusters for each word, which is arbitrary and may introduce mistakes.
- These cluster-based models are typically offline , so they cannot be efficiently adapted to new senses, new words or new data.
- It is also troublesome to find the sense that a word prototype corresponds to; thus, these cluster-based models cannot be directly used to perform word sense disambiguation.

In reality, many large knowledge bases have been constructed with word senses available online, such as WordNet (Miller, 1995) and Wikipedia. Utilizing these knowledge bases to learn word representation and sense representation is a natural choice. In this paper, we present a unified model for both word sense representation and disambiguation based on these knowledge bases and large-scale text corpora. The unified model

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1025–1035,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

can (1) perform word sense disambiguation based on vector representations, and (2) learn continuous distributed vector representation for word and sense jointly.

The basic idea is that, the tasks of word sense representation (WSR) and word sense disambiguation (WSD) can benefit from each other: (1) high-quality WSR will capture rich semantic and syntactic information of words and senses, which should be helpful for WSD; (2) high-quality WSD will provide reliable disambiguated corpora for learning better sense representations.

By utilizing these knowledge bases, the problem mentioned above can be overcome:

- The number of senses of a word can be decided by the expert annotators or web users.
- When a new sense appears, our model can be easily applied to obtain a new sense representation.
- Every sense vector has a corresponding sense in these knowledge bases.

We conduct experiments to investigate the performance of our model for both WSR and WSD. We evaluate the performance of WSR using a contextual word similarity task, and results show that out model can significantly improve the correlation with human judgments compared to baselines. We further evaluate the performance on both domain-specific WSD and coarse-grained all-words WSD, and results show that our model yields performance competitive with state-of-the-art supervised approaches.

2 Methodology

We describe our method as a 3-stage process:

1. **Initializing word vectors and sense vectors.** Given large amounts of text data, we first use the Skip-gram model (Mikolov et al., 2013), a neural network based language model, to learn word vectors. Then, we assign vector representations for senses based on their definitions (e.g, glosses in WordNet).
2. **Performing word sense disambiguation.** Given word vectors and sense vectors, we propose two simple and efficient WSD algorithms to obtain more relevant occurrences for each sense.
3. **Learning sense vectors from relevant occurrences.** Based on the relevant occurrences of ambiguous words, we modify the training objective of Skip-gram to learn word vectors and sense vectors jointly. Then, we obtain the sense vectors directly from the model.

Before illustrating the three stages of our method in Sections 2.2, 2.3 and 2.4, we briefly introduce our sense inventory, WordNet, in Section 2.1. Note that, although our experiments will use the WordNet sense inventory, our model is not limited to this particular lexicon. Other knowledge bases containing word sense distinctions and definitions can also serve as input to our model.

2.1 WordNet

WordNet (Miller, 1995) is the most widely used computational lexicon of English where a concept is represented as a synonym set, or *synset*. The words in the same synset share a common meaning. Each synset has a textual definition, or gloss. Table 1 shows the synsets and the corresponding glosses of the two common senses of *bank*.

Before introducing the method in detail, we introduce the notations. The unlabeled texts are denoted as R, and the vocabulary of the texts is denoted as W. For a word w in W, w_{s_i} is the ith sense in WordNet WN. Each sense w_{s_i} has a gloss $gloss(w_{s_i})$ in WN. The word embedding of w is denoted as $vec(w)$, and the sense embedding of its ith sense w_{s_i} is denoted as $vec(w_{s_i})$.

2.2 Initializing Word Vectors and Sense Vectors

Initializing word vectors. First, we use Skip-gram to train the word vectors from large amounts of text data. We choose Skip-gram for its simplicity and effectiveness. The training objective of Skip-gram is to train word vector representations that are good at predicting its context in the same sentence (Mikolov et al., 2013).

More formally, given a sequence of training words w_1, w_2, w_3,...,w_T, the objective of Skip-gram is to maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\left(\sum_{-k\leq j\leq k, j\neq 0}\log p(w_{t+j}|w_t)\right) \quad (1)$$

where k is the size of the training window. The inner summation spans from $-k$ to k to compute

Sense	Synset	Gloss
bank_{s_1}	bank	(sloping land (especially the slope beside a body of water)) "they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents"
bank_{s_2}	depository institution, bank, banking concern, banking company	(a financial institution that accepts deposits and channels the money into lending activities) "he cashed a check at the bank"; "that bank holds the mortgage on my home"

Table 1: Example of a synset in WordNet.

the log probability of correctly predicting the word w_{t+j} given the word in the middle w_t. The outer summation covers all words in the training data.

The prediction task is performed via softmax, a multiclass classifier. There, we have

$$p(w_{t+j}|w_t) = \frac{\exp(vec'(w_{t+j})^\top vec(w_t))}{\sum_{w=1}^{W} \exp(vec'(w)^\top vec(w_t))} \quad (2)$$

where $vec(w)$ and $vec'(w)$ are the "input" and "output" vector representations of w. This formulation is impractical because the cost of computing $p(w_{t+j}|w_t)$ is proportional to W, which is often large($10^5 - 10^7$ terms).

Initializing sense vectors. After learning the word vectors using the Skip-gram model, we initialize the sense vectors based on the glosses of senses. The basic idea of the sense vector initialization is to represent the sense by using the similar words in the gloss. From the content words in the gloss, we select those words whose cosine similarities with the original word are larger than a similarity threshold δ. Formally, for each sense w_{s_i} in WN, we first define a candidate set from $gloss(w_{s_i})$

$$cand(w_{s_i}) = \{u|u \in gloss(w_{s_i}), u \neq w, \\ POS(u) \in CW, \cos(vec(w), vec(u)) > \delta\} \quad (3)$$

where $POS(u)$ is the part-of-speech tagging of the word u and CW is the set of all possible part-of-speech tags that content words could have. In this paper, CW contains the following tags: noun, verb, adjective and adverb.

Then the average of the word vectors in $cand(w_{s_i})$ is used as the initialization value of the sense vector $vec(w_{s_i})$.

$$vec(w_{s_i}) = \frac{1}{|cand(w_{s_i})|} \sum_{u \in cand(w_{s_i})} vec(u) \quad (4)$$

For example, in WordNet, the gloss of the sense bank_{s_1} is "sloping land (especially the slope beside a body of water)) they pulled the canoe up on the bank; he sat on the bank of the river and watched the currents". The gloss contains a definition of the sense and two examples of the sense. The content words and the cosine similarities with the word "bank" are listed as follows: (sloping, 0.12), (land, 0.21), (slope, 0.17), (body, 0.01), (water, 0.10), (pulled, 0.01), (canoe, 0.09), (sat, 0.06), (river, 0.43), (watch, -0.11), (currents, 0.01). If the threshold, δ, is set to 0.05, then $cand(\text{bank}_{s_1})$ is {sloping, land, slope, water, canoe, sat, river}. Then the average of the word vectors in $cand(\text{bank}_{s_i})$ is used as the initialization value of $vec(\text{bank}_{s_i})$.

2.3 Performing Word Sense Disambiguation.

One of the state-of-the-art WSD results can be obtained using exemplar models, i.e., the word meaning is modeled by using relevant occurrences only, rather than merging all of the occurrences into a single word vector (Erk and Pado, 2010). Inspired by this idea, we perform word sense disambiguation to obtain more relevant occurrences.

Here, we perform knowledge-based word sense disambiguation for training data on an all-words setting, i.e., we will disambiguate all of the content words in a sentence. Formally, the sentence S is a sequence of words (w_1,w_2,...,w_n), and we will identify a mapping M from words to senses such that $M(i) \in Senses_{WN}(w_i)$, where $Senses_{WN}(w_i)$ is the set of senses encoded in the WN for word w_i. For sentence S, there are $\prod_{i=1}^{n} |Sense_{WN}(w_i)|$ possible mapping answers, which are impractical to compute. Thus, we design two simple algorithms, L2R (left to right) algorithm and S2C (simple to complex) algorithm, for word sense disambiguation based on the sense vectors.

The main difference between L2R and S2C is

the order of words when performing word sense disambiguation. When given a sentence, the L2R algorithm disambiguates the words from left to right (the natural order of a sentence), whereas the S2C algorithm disambiguates the words with fewer senses first. The main idea of S2C algorithm is that the words with fewer senses are easier to disambiguate, and the disambiguation result can be helpful to disambiguate the words with more senses. Both of the algorithms have three steps:

Context vector initialization. Similar to the initialization of sense vectors, we use the average of all of the content words' vectors in a sentence as the initialization vector of context.

$$vec(context) = \frac{1}{|cand(S)|} \sum_{u \in cand(S)} vec(u) \quad (5)$$

where $cand(S)$ is the set of content words $cand(S) = \{u | u \in S, POS(u) \in CW\}$.

Ranking words. For L2R, we do nothing in this step. For S2C, we rank the words based on the ascending order of $|Senses_{WN}(w_i)|$.

Word sense disambiguation. For both L2R and S2C, we denote the order of words as L and perform word sense disambiguation according to L.

First, we skip a word if the word is not a content word or the word is monosemous ($|Senses_{WN}(w_i)| = 1$). Then, for each word in L, we can compute the cosine similarities between the context vector and its sense vectors. We choose the sense that yields the maximum cosine similarity as its disambiguation result. If the score margin between the maximum and the second maximum is larger than the threshold ε, we are confident with the disambiguation result of w_i and then use the sense vector to replace the word vector in the context vector. Thus, we obtain a more accurate context vector for other words that are still yet to be disambiguated.

For example, given a sentence "He sat on the bank of the lake", we first explain how S2C works. In the sentence, there are three content words, "sat", "bank" and "lake", to be disambiguated. First, the sum of the three word vectors is used as the initialization of the context vector. Then we rank the words by $|SensesWN(w_i)|$, in ascending order, that is, lake (3 senses), bank (10 senses), sat (10 senses). We first disambiguate the word "lake" based on the similarities between its sense vectors and context vector. If the score margin is larger than the threshold ε, then we are confident with this disambiguation result and replace the word vector with the sense vector to update the context vector. It would be helpful to disambiguate the next word, "bank". We repeat this process until all three words are disambiguated.

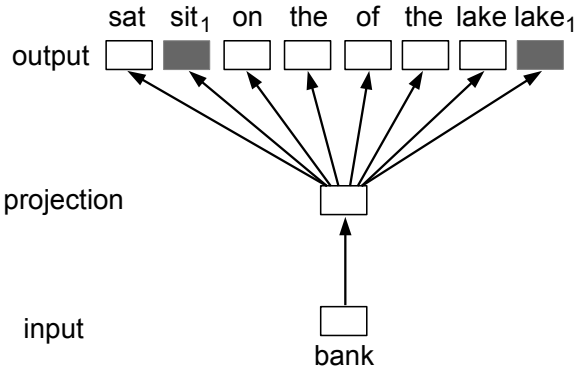


Figure 1: The architecture of our model. The training objective of Skip-gram is to train word vector representations that are not only good at predicting its context words but are also good at predicting its context words' senses. The center word "bank" is used to predict not only its context words but also the sense "sit$_1$" and "lake$_1$".

For L2R, the order of words to be disambiguated will be "sat", "bank" and "lake". In this time, when disambiguating "bank" (10 senses), we still don't know the sense of "lake" (3 senses).

2.4 Learning Sense Vectors from Relevant Occurrences.

Based on the disambiguation result, we modify the training objective of Skip-gram and train the sense vectors directly from the large-scale corpus. Our training objective is to train the vector representations that are not only good at predicting its context words but are also good at predicting its context words' senses. The architecture of our model is shown in Figure 1.

More formally, given the disambiguation result $M(w_1)$, $M(w_2)$, $M(w_3)$,...,$M(w_T)$, the training objective is modified to

$$\frac{1}{T}\sum_{t=1}^{T}\left(\sum_{j=-k}^{k}\log\{p(w_{t+j}|w_t)p(M(w_{t+j})|w_t)\}\right) \quad (6)$$

where k is the size of the training window. The inner summation spans from $-k$ to k to compute the log probability of correctly predicting the word w_{t+j} and the log probability of correctly predicting

the sense $M(w_{t+j})$ given the word in the middle w_t. The outer summation covers all words in the training data.

Because not all of the disambiguation results are correct, we only disambiguate the words that we are confident in. Similar to step 3 of our WSD algorithm, we only disambiguate words under the condition that the score margin between the maximum and the second maximum is larger than the score margin threshold, ε.

We also use the softmax function to define $p(w_{t+j}|w_t)$ and $p(M(w_{t+j})|w_t)$. Then, we use hierarchical softmax (Morin and Bengio, 2005) to greatly reduce the computational complexity and learn the sense vectors directly from the relevant occurrences.

3 Experiments

In this section, we first present the nearest neighbors of some words and their senses, showing that our sense vectors can capture the semantics of words. Then, we use three tasks to evaluate our unified model: a contextual word similarity task to evaluate our sense representations, and two standard WSD tasks to evaluate our knowledge-based WSD algorithm based on the sense vectors. Experimental results show that our model not only improves the correlation with human judgments on the contextual word similarity task but also outperforms state-of-the-art supervised WSD systems on domain-specific datasets and competes with them in a coarse-grained all-words setting.

We choose Wikipedia as the corpus to train the word vectors because of its wide coverage of topics and words usages. We use an English Wikipedia database dump from October 2013 [2], which includes roughly 3 million articles and 1 billion tokens. We use Wikipedia Extractor [3] to preprocess the Wikipedia pages and only save the content of the articles.

We use word2vec [4] to train Skip-gram. We use the default parameters of word2vec and the dimension of the vector representations is 200.

We use WordNet [5] as our sense inventory. The datasets for different tasks are tagged with different versions of WordNet. The version of WordNet is 1.7 for the domain-specific WSD task and 2.1 for the coarse-grained WSD task.

We use the S2C algorithm described in Section 2.3 to perform word sense disambiguation to obtain more relevant occurrences for each sense. We compare S2C and L2R on the coarse-grained WSD task in a all-words setting.

The experimental results of our model are obtained by setting the similarity threshold as $\delta = 0$ and the score margin threshold as $\varepsilon = 0.1$. The influence of parameters on our model can be found in Section 3.5.

Word or sense	Nearest neighbors
bank	banks, IDBI, CitiBank
bank_{s_1}	river, slope, Sooes
bank_{s_2}	mortgage, lending, loans
star	stars, stellar, trek
star_{s_1}	photosphere, radiation, gamma-rays
star_{s_2}	someone, skilled, genuinely
plant	plants, glavaticevo, herbaceous
plant_{s_1}	factories, machinery, manufacturing
plant_{s_2}	locomotion, organism, organisms

Table 2: Nearest neighbors of word vectors and sense vectors learned by our model based on cosine similarity. The subscript of each sense label corresponds to the index of the sense in WordNet. For example, bank_{s_2} is the second sense of the word bank in WordNet.

3.1 Examples for Sense Vectors

Table 2 shows the nearest neighbors of word vectors and sense vectors based on cosine similarity. We see that our sense representations can identify different meanings of a word, allowing our model to capture more semantic and syntactic relationships between words and senses. Note that each sense vector in our model corresponds to a sense in WordNet; thus, our sense vectors can be used to perform knowledge-based word sense disambiguation, whereas the vectors of cluster-based models cannot.

3.2 Contextual Word Similarity

Experimental setting. A standard dataset for evaluating a vector-space model is the WordSim-353 dataset (Finkelstein et al., 2001), which con-

[2] http://download.wikipedia.org.

[3] The tool is available from http://medialab.di.unipi.it/wiki/Wikipedia_Extractor.

[4] The code is available from https://code.google.com/p/word2vec/.

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

Model	$\rho \times 100$
C&W-S	57.0
Huang-S	58.6
Huang-M AvgSim	62.8
Huang-M AvgSimC	65.7
Our Model-S	64.2
Our Model-M AvgSim	66.2
Our Model-M AvgSimC	**68.9**

Table 3: Spearman's ρ on the SCWS dataset. Our Model-S uses one representation per word to compute similarities, while Our Model-M uses one representation per sense to compute similarities. AvgSim calculates the similarity with each sense contributing equally, while AvgSimC weighs the sense according to the probability of the word choosing that sense in context c.

sists of 353 pairs of nouns. However, each pair of nouns in WordSim-353 is presented without context. This is problematic because the meanings of homonymous and polysemous words depend highly on the words' contexts. Thus we choose the Stanford's Contextual Word Similarities (SCWS) dataset from (Huang et al., 2012) [6]. The SCWS dataset contains 2003 pairs of words and each pair is associated with 10 human judgments on similarity on a scale from 0 to 10. In the SCWS dataset, each word in a pair has a sentential context.

In our experiments, the similarity between a pair of words (w, w') is computed as follows:

$$AvgSimC(w,w') = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} p(i|w,c)p(j|w',c')d(vec(w_{s_i}),vec(w'_{s_j})) \quad (7)$$

where $p(i|w,c)$ is the likelihood that word w chooses its ith sense given context c. $d(vec,vec')$ is a function computing the similarity between two vectors, and here we use cosine similarity.

Results and discussion. For evaluation, we compute the Spearman correlation between a model's computed similarity scores and human judgements. Table 3 shows our results compared to previous methods, including (Collobert and Weston, 2008)'s language model (C&W), and Huang's model which utilize the global context and multi-prototype to improve the word representations.

[6]The dataset can be downloaded at `http://ai.stanford.edu/~ehhuang/`.

From Table 3, we observe that:

- Our single-vector version outperforms Huang's single-vector version. This indicates that, by training the word vectors and sense vectors jointly, our model can better capture the semantic relationships between words and senses.
- With one representation per sense, our model can outperform the single-vector version without using context (66.2 vs. 64.2).
- Our model obtains the best performance (68.9) by using AvgSimC, which takes context into account.

3.3 Domain-Specific WSD

Experimental setting. We use Wikipedia as training data because of its wide coverage for specific domains. To test our performance on domain word sense disambiguation, we evaluated our system on the dataset published in (Koeling et al., 2005). This dataset consists of examples retrieved from the Sports and Finance sections of the Reuters corpus. 41 words related to the Sports and Finance domains were selected, with an average polysemy of 6.7 senses, ranging from 2 to 13 senses.

Approximately 100 examples for each word were annotated with senses from WordNet v.1.7 by three reviewers, yielding an inter-tagger agreement of 65%. (Koeling et al., 2005) did not clarify how to select the "correct" sense for each word, so we followed the work of (Agirre et al., 2009) and, used the sense chosen by the majority of taggers as the correct answer.

Baseline methods. As a baseline, we use the most frequent WordNet sense (MFS), as well as a random sense assignment. We also compare our results with four systems [7]: Static PageRank (Agirre et al., 2009), the k nearest neighbor algorithm (k-NN), Degree (Navigli and Lapata, 2010) and Personalized PageRank (Agirre et al., 2009).

Static PageRank applies traditional PageRank over the semantic graph based on WordNet and obtains a context-independent ranking of word senses.

k-NN is a widely used classification method, where neighbors are the k labeled examples most

[7]We compare only with those systems performing token-based WSD, i.e., disambiguating each instance of a target word separately.

Algorithm	Sports Recall	Finance Recall
Random BL	19.5	19.6
MFS BL	19.6	37.1
k-NN	30.3	43.4
Static PR	20.1	39.6
Personalized PR	35.6	46.9
Degree	42.0	47.8
Our Model	**57.3**	**60.6**

Table 4: Performance on the Sports and Finance sections of the dataset from (Koeling et al., 2005).

similar to the test example. The *k*-NN system is trained on SemCor (Miller et al., 1993), the largest publicly available annotated corpus.

Degree and Personalized PageRank are state-of-the-art systems that exploit WordNet to build a semantic graph and exploit the structural properties of the graph in order to choose the appropriate senses of words in context.

Results and discussion. Similar to other work on this dataset, we use recall (the ratio of correct sense labels to the total labels in the gold standard) as our evaluation measure. Table 4 shows the results of different WSD systems on the dataset, and the best results are shown in bold. The differences between other results and the best result in each column of the table are statistically significant at $p < 0.05$.

The results show that:

- Our model outperforms *k*-NN on the two domains by a large margin, supporting the findings from (Agirre et al., 2009) that knowledge-based systems perform better than supervised systems when evaluated across different domains.
- Our model also achieves better results than the state-of-the-art system (+15.3% recall on Sports and +12.8% recall on Finance against Degree). The reason for this is that when dealing with short sentences or context words that are not in WordNet, our model can still compute similarity based on the context vector and sense vectors, whereas Degree will have difficulty building the semantic graph.
- Moreover, our model achieves the best performance by only using the unlabeled text data and the definitions of senses, whereas other methods rely greatly on high-quality semantic relations or annotated data, which are hard to acquire.

Algorithm	Type	Nouns only F_1	All words F_1
Random BL	U	63.5	62.7
MFS BL	Semi	77.4	78.9
SUSSX-FR	Semi	81.1	77.0
NUS-PT	S	82.3	**82.5**
SSI	Semi	**84.1**	**83.2**
Degree	Semi	**85.5**	**81.7**
Our Model$_{L2R}$	U	79.2	73.9
Our Model$_{S2C}$	U	81.6	75.8
Our Model$_{L2R}$	Semi	82.5	79.6
Our Model$_{S2C}$	Semi	**85.3**	**82.6**

Table 5: Performance on Semeval-2007 coarse-grained all-words WSD. In the type column, U, Semi and S stand for unsupervised, semi-supervised and supervised, respectively. The differences between the results in bold in each column of the table are not statistically significant at $p < 0.05$.

3.4 Coarse-grained WSD

Experimental setting. We also evaluate our WSD model on the Semeval-2007 coarse-grained all-words WSD task (Navigli et al., 2007). There are multiple reasons that we perform experiments in a coarse-grained setting: first, it has been argued that the fine granularity of WordNet is one of the main obstacles to accurate WSD (cf. the discussion in (Navigli, 2009)); second, the training corpus of word representations is Wikipedia, which is quite different from WordNet.

Baseline methods. We compare our model with the best unsupervised system SUSSX-FR (Koeling and McCarthy, 2007), and the best supervised system, NUS-PT (Chan et al., 2007), participating in the Semeval-2007 coarse-grained all-words task. We also compare with SSI (Navigli and Velardi, 2005) and the state-of-the-art system Degree (Navigli and Lapata, 2010). We use different baseline methods for the two WSD tasks because we want to compare our model with the state-of-the-art systems that are applicable to different datasets and show that our WSD method can perform robustly in these different WSD tasks.

Results and discussion. We report our results in terms of F_1-measure on the Semeval-2007 coarse-grained all-words dataset (Navigli et al., 2007). Table 5 reports the results for nouns (1,108 words) and all words (2,269 words). The difference between unsupervised and semi-supervised methods is whether the method uses MFS as a back-off strategy.

We can see that the S2C algorithm outperforms the L2R algorithm no matter on the nouns subset or on the entire set. This indicates that words with fewer senses are easier to disambiguate, and it can be helpful to disambiguate the words with more senses.

On the nouns subset, our model yields comparable performance to SSI and Degree, and it outperforms NUS-PT and SUSSX-FR. Moreover, our unsupervised WSD method (S2C) beats the MFS baseline, which is notably a difficult competitor for knowledge-based systems.

On the entire set, our semi-supervised model is significantly better than SUSSX-FR, and it is comparable with SSI and Degree. In contrast to SSI, our model is simple and does not rely on a costly annotation effort to engineer the set of semantic relations.

Overall, our model achieves state-of-the-art performance on the Semeval-2007 coarse-grained all-words dataset compared to other systems, with a simple WSD algorithm that only relies on a large-scale unlabeled text corpora and a sense inventory.

3.5 Parameter Influence

We investigate the influence of parameters on our model with coarse-grained all-words WSD task. The parameters include the similarity threshold, δ, and the score margin threshold, ε.

Similarity threshold. In Table 6, we show the performance of domain WSD when the similarity threshold δ ranges from -0.1 to 0.3. The cosine similarity interval is [-1, 1], and we focus on the performance in the interval [-0.1, 0.3] for two reasons: first, no words are removed from glosses when $\delta < -0.1$; second, nearly half of the words are removed when $\delta > 0.3$ and the performance drops significantly for the WSD task. From table 6, we can see that our model achieves the best performance when $\delta = 0.0$.

Score margin threshold. In Table 7, we show the performance on the coarse-grained all-words WSD when the score margin threshold ε ranges from 0.0 to 0.3. When $\varepsilon = 0.0$, we use every disambiguation result to update the context vector. When $\varepsilon \neq 0$, we only use the confident disambiguation results to update the context vector if the score margin is larger than ε. Our model achieves the best performance when $\varepsilon = 0.1$.

Parameter	Nouns only	All words
$\delta = -0.10$	79.8	74.3
$\delta = -0.05$	81.0	74.6
$\delta = 0.00$	**81.6**	**75.8**
$\delta = 0.05$	81.3	75.4
$\delta = 0.10$	80.8	75.2
$\delta = 0.15$	80.0	75.0
$\delta = 0.20$	77.1	73.3
$\delta = 0.30$	75.0	72.1

Table 6: Evaluation results on the coarse-grained all-words WSD when the similarity threshold δ ranges from -0.1 to 0.3.

Parameter	Nouns only	All words
$\varepsilon = 0.00$	78.2	72.9
$\varepsilon = 0.05$	79.5	74.5
$\varepsilon = 0.10$	**81.6**	**75.8**
$\varepsilon = 0.15$	81.2	74.7
$\varepsilon = 0.20$	80.9	75.1
$\varepsilon = 0.25$	80.2	74.8
$\varepsilon = 0.30$	80.4	74.9

Table 7: Evaluation results on the coarse-grained all-words WSD when the score margin threshold ε ranges from 0.0 to 0.3.

4 Related Work

4.1 Word Representations

Distributed representations for words were proposed in (Rumelhart et al., 1986) and have been successfully used in language models (Bengio et al., 2006; Mnih and Hinton, 2008) and many natural language processing tasks, such as word representation learning (Mikolov, 2012), named entity recognition (Turian et al., 2010), disambiguation (Collobert et al., 2011), parsing and tagging (Socher et al., 2011; Socher et al., 2013). They are very useful in NLP tasks because they can be used as inputs to learning algorithms or as extra word features in NLP systems. Hence, many NLP applications, such as keyword extraction (Li-

u et al., 2010; Liu et al., 2011b; Liu et al., 2012), social tag suggestion (Liu et al., 2011a) and text classification (Baker and McCallum, 1998), may also potentially benefit from distributed word representation. The main advantage is that the representations of similar words are close in vector space, which makes generalization to novel patterns easier and model estimation more robust.

Word representations are hard to train due to the computational complexity. Recently, (Mikolov et al., 2013) proposed two particular models, Skip-gram and CBOW, to learn word representations in large amounts of text data. The training objective of the CBOW model is to combine the representations of the surrounding words to predict the word in the middle, while the Skip-gram model's is to learn word representations that are good at predicting its context in the same sentence (Mikolov et al., 2013). Our paper uses the model architecture of Skip-gram.

Most of the previous vector-space models use one representation per word. This is problematic because many words have multiple senses. The multi-prototype approach has been widely studied. (Reisinger and Mooney, 2010) proposed the multi-prototype vector-space model. (Huang et al., 2012) used the multi-prototype models to learn the vector for different senses of a word. All of these models use the clustering of contexts as a word sense and can not be directly used in word sense disambiguation.

After our paper was submitted, we perceive the following recent advances: (Tian et al., 2014) proposed a probabilistic model for multi-prototype word representation. (Guo et al., 2014) explored bilingual resources to learn sense-specific word representation. (Neelakantan et al., 2014) proposed an efficient non-parametric model for multi-prototype word representation.

4.2 Knowledge-based WSD

The objective of word sense disambiguation (WSD) is to computationally identify the meaning of words in context (Navigli, 2009). There are two approaches of WSD that assign meaning of words from a fixed sense inventory, supervised and knowledge-based methods. Supervised approaches require large labeled training sets, which are time consuming to create. In this paper, we only focus on knowledge-based word sense disambiguation.

Knowledge-based approaches exploit knowledge resources (such as dictionaries, thesauri, ontologies, collocations, etc.) to determine the senses of words in context. However, it has been shown in (Cuadros and Rigau, 2006) that the amount of lexical and semantic information contained in such resources is typically insufficient for high-performance WSD. Much work has been presented to automatically extend existing resources, including automatically linking Wikipedia to WordNet to include full use of the first WordNet sense heuristic (Suchanek et al., 2008), a graph-based mapping of Wikipedia categories to WordNet synsets (Ponzetto and Navigli, 2009), and automatically mapping Wikipedia pages to WordNet synsets (Ponzetto and Navigli, 2010).

It was recently shown that word representations can capture semantic and syntactic information between words (Mikolov et al., 2013). Some researchers tried to incorporate WordNet senses in a neural model to learn better word representations (Bordes et al., 2011). In this paper, we have proposed a unified method for word sense representation and disambiguation to extend the information contained in the vector representations to the existing resources. Our method only requires a large amount of unlabeled text to train sense representations and a dictionary to provide the definitions of word meanings, which makes it easily applicable to other resources.

5 Conclusion

In this paper, we present a unified model for word sense representation and disambiguation that uses one representation per sense. Experimental results show that our model improves the performance of contextual word similarity compared to existing WSR methods, outperforms state-of-the-art supervised methods on domain-specific WSD, and achieves competitive performance on coarse-grained all-words WSD. Our model only requires large-scale unlabeled text corpora and a sense inventory for WSD, thus it can be easily applied to other corpora and tasks.

There are still several open problems that should be investigated further:

1. Because the senses of words change over time (new senses appear), we will incorporate cluster-based methods in our model to find senses that are not in the sense inventory.

2. We can explore other WSD methods based on sense vectors to improve our performance. For example, (Li et al., 2010) used LDA to perform data-driven WSD in a manner similar to our model. We may integrate the advantages of these models and our model together to build a more powerful WSD system.

3. To learn better sense vectors, we can exploit the semantic relations (such as the hypernym and hyponym relations defined in WordNet) between senses in our model.

Acknowledgments

This work is supported by National Key Basic Research Program of China (973 Program 2014CB340500) and National Natural Science Foundation of China (NSFC 61133012).

References

Eneko Agirre, Oier Lopez De Lacalle, Aitor Soroa, and Informatika Fakultatea. 2009. Knowledge-based wsd and specific domains: Performing better than generic supervised wsd. In *Proceedings of IJCAI*, pages 1501–1506.

L Douglas Baker and Andrew Kachites McCallum. 1998. Distributional clustering of words for text classification. In *Proceedings of SIGIR*, pages 96–103.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of AAAI*, pages 301–306.

Yee Seng Chan, Hwee Tou Ng, and Zhi Zhong. 2007. Nus-pt: exploiting parallel texts for word sense disambiguation in the english all-words tasks. In *Proceedings of SemEval*, pages 253–256.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

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

Montse Cuadros and German Rigau. 2006. Quality assessment of large scale knowledge resources. In *Proceedings of EMNLP*, pages 534–541.

Katrin Erk and Sebastian Pado. 2010. Exemplar-based models for word meaning in context. In *Proceedings of ACL*, pages 92–97.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of WWW*, pages 406–414.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning sense-specific word embeddings by exploiting bilingual resources. In *Proceedings of COLING*, pages 497–507.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*, pages 873–882.

Rob Koeling and Diana McCarthy. 2007. Sussx: Wsd using automatically acquired predominant senses. In *Proceedings of SemEval*, pages 314–317.

Rob Koeling, Diana McCarthy, and John Carroll. 2005. Domain-specific sense distributions and predominant sense acquisition. In *Proceedings of HLT-EMNLP*, pages 419–426.

Linlin Li, Benjamin Roth, and Caroline Sporleder. 2010. Topic models for word sense disambiguation and token-based idiom detection. In *Proceedings of ACL*, pages 1138–1147.

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of EMNLP*, pages 366–376.

Zhiyuan Liu, Xinxiong Chen, and Maosong Sun. 2011a. A simple word trigger method for social tag suggestion. In *Proceedings of EMNLP*, pages 1577–1588.

Zhiyuan Liu, Xinxiong Chen, Yabin Zheng, and Maosong Sun. 2011b. Automatic keyphrase extraction by bridging vocabulary gap. In *Proceedings of CoNLL*, pages 135–144.

Zhiyuan Liu, Xinxiong Chen, and Maosong Sun. 2012. Mining the interests of chinese microbloggers via keyword extraction. *Frontiers of Computer Science*, 6(1):76–87.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press Cambridge.

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

Tomas Mikolov. 2012. *Statistical Language Models Based on Neural Networks*. Ph.D. thesis, Ph. D. thesis, Brno University of Technology.

George A Miller, Claudia Leacock, Randee Tengi, and Ross T Bunker. 1993. A semantic concordance. In *Proceedings of the workshop on HLT*, pages 303–308.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Andriy Mnih and Geoffrey E Hinton. 2008. A scalable hierarchical distributed language model. In *Proceedings of NIPS*, pages 1081–1088.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252.

Roberto Navigli and Mirella Lapata. 2010. An experimental study of graph connectivity for unsupervised word sense disambiguation. *IEEE PAMI*, 32(4):678–692.

Roberto Navigli and Paola Velardi. 2005. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. *IEEE PAMI*, 27(7):1075–1086.

Roberto Navigli, Kenneth C Litkowski, and Orin Hargraves. 2007. Semeval-2007 task 07: Coarse-grained english all-words task. In *Proceedings of SemEval*, pages 30–35.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *CSUR*, 41(2):10.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of EMNLP*.

Simone Paolo Ponzetto and Roberto Navigli. 2009. Large-scale taxonomy mapping for restructuring and integrating wikipedia. In *Proceedings of IJCAI*, volume 9, pages 2083–2088.

Simone Paolo Ponzetto and Roberto Navigli. 2010. Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proceedings of ACL*, pages 1522–1531.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Proceedings of HLT-NAACL*, pages 109–117.

David E Rumelhart, Geoffrey E Hintont, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *CSUR*, 34(1):1–47.

Richard Socher, Cliff C Lin, Andrew Ng, and Chris Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of ICML*, pages 129–136.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of ACL*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. Yago: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203–217.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *Proceedings of COLING*, pages 151–160.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

Reducing Dimensions of Tensors in Type-Driven Distributional Semantics

Tamara Polajnar Luana Făgărăşan Stephen Clark
Computer Laboratory
University of Cambridge
Cambridge, UK
first.last@cl.cam.ac.uk

Abstract

Compositional distributional semantics is a subfield of Computational Linguistics which investigates methods for representing the meanings of phrases and sentences. In this paper, we explore implementations of a framework based on Combinatory Categorial Grammar (CCG), in which words with certain grammatical types have meanings represented by multi-linear maps (i.e. multi-dimensional arrays, or tensors). An obstacle to full implementation of the framework is the size of these tensors. We examine the performance of lower dimensional approximations of transitive verb tensors on a sentence plausibility/selectional preference task. We find that the matrices perform as well as, and sometimes even better than, full tensors, allowing a reduction in the number of parameters needed to model the framework.

1 Introduction

An emerging subfield of computational linguistics is concerned with learning compositional distributional representations of meaning (Mitchell and Lapata, 2008; Baroni and Zamparelli, 2010; Coecke et al., 2010; Grefenstette and Sadrzadeh, 2011; Clarke, 2012; Socher et al., 2012; Clark, 2013). The advantage of such representations lies in their potential to combine the benefits of distributional approachs to word meaning (Schütze, 1998; Turney and Pantel, 2010) with the more traditional compositional methods from formal semantics (Dowty et al., 1981). Distributional representations have the properties of robustness, learnability from data, ease of handling ambiguity, and the ability to represent gradations of meaning; whereas compositional models handle the unbounded nature of natural language, as well as providing established accounts of logical words, quantification, and inference.

One promising approach which attempts to combine elements of compositional and distributional semantics is by Coecke et al. (2010). The underlying idea is to take the type-driven approach from formal semantics — in particular the idea that the meanings of complex grammatical types should be represented as functions — and apply it to distributional representations. Since the mathematics of distributional semantics is provided by linear algebra, a natural set of functions to consider is the set of linear maps. Coecke et al. recognize that there is a natural correspondence from complex grammatical types to tensors (multi-linear maps), so that the meaning of an adjective, for example, is represented by a matrix (a 2nd-order tensor)[1] and the meaning of a transitive verb is represented by a 3rd-order tensor.

Coecke et al. use the grammar of pregroups as the syntactic machinery to construct distributional meaning representations, since both pregroups and vector spaces can be seen as examples of the same abstract structure, which leads to a particularly clean mathematical description of the compositional process. However, the approach applies more generally, for example to other forms of categorial grammar, such as Combinatory Categorial Grammar (Steedman, 2000; Maillard et al., 2014), and also to phrase-structure grammars in a way that a formal linguist would recognize (Baroni et al., 2014). Clark (2013) provides a description of the tensor-based framework aimed more at computational linguists, relying only on the mathematics of multi-linear algebra rather than the category theory used in Coecke et al. (2010). Section 2 repeats some of this description.

A major open question associated with the tensor-based semantic framework is how to learn

[1] This same insight lies behind the work of Baroni and Zamparelli (2010).

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1036–1046,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

the tensors representing the meanings of words with complex types, such as verbs and adjectives. The framework is essentially a compositional framework, providing a recipe for how to combine distributional representations, but leaving open what the underlying vector spaces are and how they can be acquired. One significant challenge is an engineering one: in a wide-coverage grammar, which is able to handle naturally occurring text, there will be a) a large lexicon with many word-category pairs requiring tensor representations; and b) many higher-order tensors with large numbers of parameters which need to be learned. In this paper we take a first step towards learning such representations, by learning tensors for transitive verbs.

One feature of the tensor-based framework is that it allows the meanings of words and phrases with different basic types, for example nouns and sentences, to live in different vector spaces. This means that the sentence space is task specific, and must be defined in advance. For example, to calculate sentence similarity, we would have to learn a vector space where distances between vectors representing the meanings of sentences reflect similarity scores assigned by human annotators.

In this paper we describe an initial investigation into the learning of word meanings with complex syntactic types, together with a simple sentence space. The space we consider is the "plausibility space" described by Clark (2013), together with sentences of the form subject-verb-object. This space is defined to distinguish semantically plausible sentences (e.g. *Animals eat plants*) from implausible ones (e.g. *Animals eat planets*). Plausibility can be either represented as a single continuous variable between 0 and 1, or as a two-dimensional probability distribution over the classes *plausible* ($\top$) and *implausible* ($\bot$). Whether we consider a one- or two-dimensional sentence space depends on the architecture of the logistic regression classifier that is used to learn the verb (Section 3).

We begin with this simple plausibility sentence space to determine if, in fact, the tensor-based representation can be learned to a sufficiently useful degree. Other simple sentence spaces which can perhaps be represented using one or two variables include a "sentence space" for the sentiment analysis task (Socher et al., 2013), where one variable represents positive sentiment and the other negative. We also expect that the insights gained from research on this task can be applied to more complex sentence spaces, for example a semantic similarity space which will require more than two variables.

2 Syntactic Types to Tensors

The syntactic type of a transitive verb in English is $(S\backslash NP)/NP$ (using notation from Steedman (2000)), meaning that a transitive verb is a function which takes an NP argument to the right, an NP argument to the left, and results in a sentence S. Such *categories* with slashes are *complex categories*; S and NP are *basic* or *atomic* categories. Interpreting such categories under the Coecke et al. framework is straightforward. First, for each atomic category there is a corresponding vector space; in this case the sentence space $\mathbf{S}$ and the noun space $\mathbf{N}$.[2] Hence the meaning of a noun or noun phrase, for example *people*, will be a vector in the noun space: $\overrightarrow{people} \in \mathbf{N}$. In order to obtain the meaning of a transitive verb, each slash is replaced with a tensor product operator, so that the meaning of *eat*, for example, is a 3rd-order tensor: $\overline{eat} \in \mathbf{S} \otimes \mathbf{N} \otimes \mathbf{N}$. Just as in the syntactic case, the meaning of a transitive verb is a function (a multi-linear map) which takes two noun vectors as arguments and returns a sentence vector.

Meanings combine using *tensor contraction*, which can be thought of as a multi-linear generalisation of matrix multiplication (Grefenstette, 2013). Consider first the adjective-noun case, for example *black cat*. The syntactic type of *black* is N/N; hence its meaning is a 2nd-order tensor (matrix): $\overline{black} \in \mathbf{N} \otimes \mathbf{N}$. In the syntax, N/N combines with N using the rule of forward application ($N/N \;\; N \Rightarrow N$), which is an instance of function application. Function application is also used in the tensor-based semantics, which, for a matrix and vector argument, corresponds to matrix multiplication.

Figure 1 shows how the syntactic types combine with a transitive verb, and the corresponding tensor-based semantic types. Note that, after the verb has combined with its object NP, the type of the verb phrase is $S\backslash NP$, with a corresponding meaning tensor (matrix) in $\mathbf{S} \otimes \mathbf{N}$. This matrix then combines with the subject vector, through

[2] In practice, for example using the CCG parser of Clark and Curran (2007), there will be additional atomic categories, such as PP, but not many more.

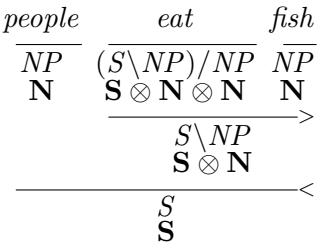


Figure 1: Syntactic reduction and tensor-based semantic types for a transitive verb sentence

matrix multiplication, to give a sentence vector.

In practice, using for example the wide-coverage grammar from CCGbank (Hockenmaier and Steedman, 2007), there will be many types with more than 3 slashes, with corresponding higher-order tensors. For example, a common category for a preposition is the following: $((S\backslash NP)\backslash(S\backslash NP))/NP$, which would be assigned to WITH in *eat* WITH *a fork*. (The way to read the syntactic type is as follows: *with* requires an NP argument to the right – *a fork* in this example – and then a verb phrase to the left – *eat* with type $S\backslash NP$ – resulting in a verb phrase $S\backslash NP$.) The corresponding meaning tensor lives in the tensor space $\mathbf{S}\otimes\mathbf{N}\otimes\mathbf{S}\otimes\mathbf{N}\otimes\mathbf{N}$, i.e. a 5th-order tensor. Categories with even more slashes are not uncommon, for example $((N/N)/(N/N))/((N/N)/(N/N))$. Clearly learning parameters for such tensors is highly challenging, and it is likely that lower dimensional approximations will be required.

3 Methods

In this paper we compare five different methods for modelling the type-driven semantic representation of subject-verb-object sentences. The tensor is a function that encodes the meaning of a verb. It takes two vectors from the K-dimensional noun space as input, and produces a representation of the sentence in the S-dimensional sentence space. In this paper, we consider a *plausibility space* where $\mathbf{S}$ is either a single variable or a two-dimensional space over two classes: *plausible* ($\top$) and *implausible* ($\bot$).

The first method (**Tensor**) follows Krishnamurthy and Mitchell (2013) by learning a tensor as parameters in a softmax classifier. We introduce three related methods (**2Mat**, **SKMat**, **KKMat**), all of which model the verb as a matrix or a pair of matrices (Figure 2). Table 1 gives the number of parameters for each method. **Tensor**, **2Mat**, and **SKMat** all have a two-dimensional $\mathbf{S}$ space, while **KKMat** produces a scalar value. In all of these learning-based methods the derivatives were obtained via the chain rule with respect to each set of parameters and gradient descent was performed using the Adagrad algorithm (Duchi et al., 2011).

	Tensor	**2Mat**	**SKMat**	**KKMat**	**DMat**
V	$2K^2$	$4K$	$2K$	K^2	K^2
Θ	4	8	4	0	0

Table 1: Number of parameters per method.

We also reimplement a distributional method (**DMat**), which was previously used in SVO experiments with the type-driven framework (Grefenstette and Sadrzadeh, 2011). While the other methods are trained as plausibility classifiers, in **DMat** we estimate the class boundary from cosine similarity via training data (see explanation below).

Tensor If subject (n_s) and object (n_o) nouns are K-dimensional vectors and the plausibility vector is S-dimensional with $S = 2$, we can learn the values of the $K \times K \times S$ tensor representing the verb as parameters (V) of a regression algorithm. To represent this space as a distribution over two classes ($\top$,$\bot$) we apply a sigmoid function (σ) to restrict the output to the [0,1] range and the softmax activation function (g) to balance the class probabilities. The full parameter set which needs to be optimised for is $\mathrm{B} = \{\mathrm{V}, \Theta\}$, where $\Theta = \{\theta_\top, \theta_\bot\}$ are the softmax parameters for the two classes. For each verb we optimise the KL-divergence $\mathcal{L}$ between the training labels t^i and classifier predictions using the following regularised objective:

$$O(\mathrm{B}) = \sum_{i=1}^{N} \mathcal{L}\left(t^i, g\left(\sigma\left(h_\mathrm{V}\left(n_s^i, n_o^i\right)\right), \Theta\right)\right) + \frac{\lambda}{2}||\mathrm{B}||^2 \quad (1)$$

where n_s^i and n_o^i are the subject and object of the training instance $i \in N$, and $h_\mathrm{V}\left(n_s^i, n_o^i\right) = (n_s^i)\mathrm{V}(n_o^i)^T$ describes tensor contraction. The function h_V is described diagrammatically in Figure 2-**(a)**, where the verb tensor parameters are drawn as a cube with the subject and object noun vectors as operands on either side. The output is a two-dimensional vector which is then transformed using the sigmoid and softmax functions.

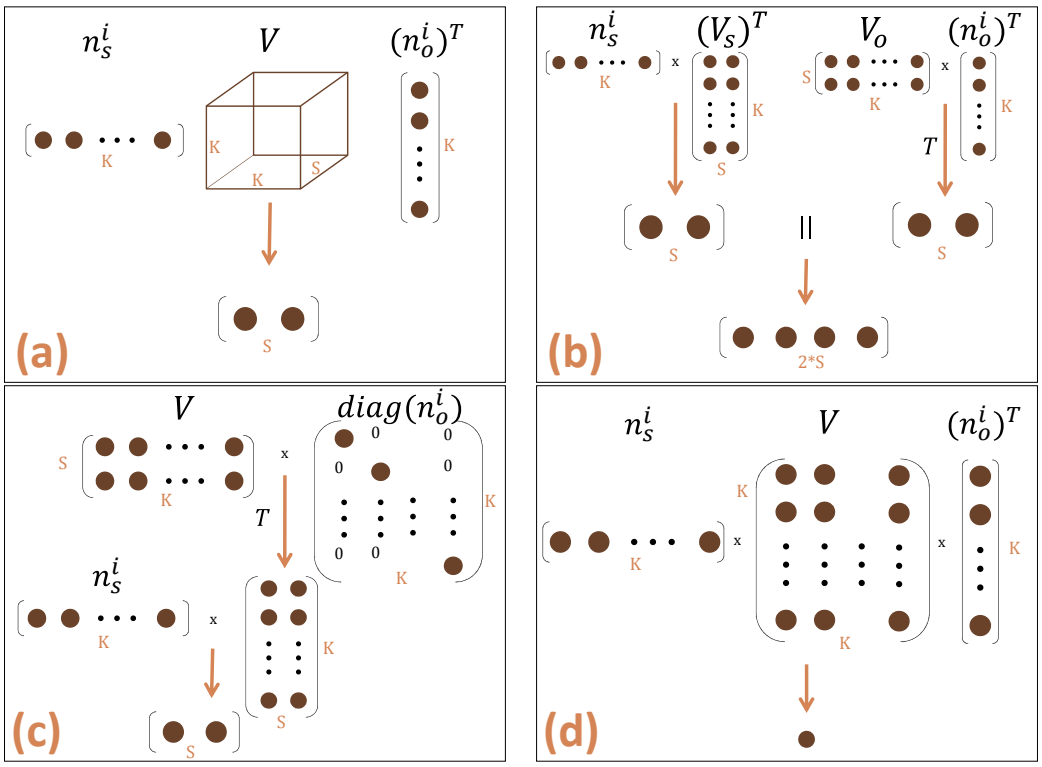


Figure 2: Illustrations of the h_{V} function for the regression-based methods **(a)-Tensor**, **(b)-2Mat**, **(c)-SKMat**, **(d)-KKMat**. The operation in **(a)** is tensor contraction, T denotes transpose, and $\times$ denotes matrix multiplication.

The gold-standard distribution over training labels is defined as $(1, 0)$ or $(0, 1)$, depending on whether the training instance is a positive (plausible) or negative (implausible) example. Tensor contraction is implemented using the Matlab Tensor Toolbox (Bader et al., 2012).

2Mat An alternative approach is to decouple the interaction between the object and subject by learning a pair of $S \times K$ ($S = 2$) matrices (V_s, V_o) for each of the input noun vectors (one matrix for the subject slot of the verb and one for the object slot). The resulting S-vectors are concatenated, after the subject and object nouns have been combined with their matrices, and combined with the softmax component to produce the output distribution. Therefore the objective function is the same as in Equation 1, but h_{V} is defined as:

$$h_{\mathrm{V}}\left(n_s^i, n_o^i\right) = \left((n_s^i)\mathrm{V}_s^T\right) || \left(\mathrm{V}_o(n_o^i)^T\right)^T$$

where $||$ represents vector concatenation. The intention is to test whether we can learn the verb without directly multiplying subject and object features, n_s^i and n_o^j. The function h_{V} is shown in Figure 2-**(b)**, where the verb tensor parameters are drawn as two $2 \times K$ matrices, one of which interacts with the subject and the other with the object noun vector. The output is a four-dimensional vector whose values are then restricted to [0,1] using the sigmoid function and then transformed into a two-dimensional distribution over the classes using the softmax function.

SKMat A third option for generating a sentence vector with $S = 2$ dimensions is to consider the verb as an $S \times K$ matrix. If we transform the object vector into a $K \times K$ matrix with the noun on the diagonal and zeroes elsewhere, we can combine the verb and object to produce a new $S \times K$ matrix, which is encoding the meaning of the verb phrase. We can then complete the sentence reduction by multiplying the subject vector with this verb phrase vector to produce an S-dimensional sentence vector. Formally, we define **SKMat** as:

$$h_{\mathrm{V}}\left(n_s^i, n_o^i\right) = n_s^i \left(\mathrm{V}diag(n_o^i)\right)^T$$

and use it in Equation 1. The function h_{V} is described in Figure 2-**(c)**, where the verb tensor parameters are drawn as a matrix, the subject as a vector, and the object as a diagonal ma-

trix. The graphic demonstrates the two-step combination and the intermediate $S \times K$ verb phrase matrix, as well as the the noun vector product that results in a two-dimensional vector which is then transformed using the sigmoid and softmax functions. Whilst the tensor method captures the interactions between all pairs of context features ($n_{si} \cdot n_{oj}$), **SKMat** only captures the interactions between matching features ($n_{si} \cdot n_{oi}$).

KKMat Given a two-class problem, such as plausibility classification, the softmax implementation is overparameterised because the class membership can be estimated with a single variable. To produce a scalar output, we can learn the parameters for a single $K \times K$ matrix (V) using standard logistic regression with the mean squared error cost function:

$$O(\mathrm{V}) = -\frac{1}{m}\left[\sum_{i=1}^{N} t^i \log h_{\mathrm{V}}\left(n_s^i, n_o^i\right) + (1-t^i)\log h_{\mathrm{V}}\left(n_s^i, n_o^i\right)\right]$$

where $h_{\mathrm{V}}\left(n_s^i, n_o^i\right) = (n_s^i)\mathrm{V}(n_o^i)^T$ and the objective is regularised: $O(\mathrm{V}) + \frac{\lambda}{2}||\mathrm{V}||^2$. This function is shown in Figure 2-**(d)**, where the verb parameters are shown as a matrix, while the subject and object are vectors. The output is a single scalar, which is then transformed with the sigmoid function. Values over 0.5 are considered plausible.

DMat The final method produces a scalar as in **KKMat**, but is distributional and based on corpus counts rather than regression-based. Grefenstette and Sadrzadeh (2011) introduced a corpus-based approach for generating a $K \times K$ matrix for each verb from an average of Kronecker products of the subject and object vectors from the positively labelled subset of the training data. The intuition is that, for example, the matrix for *eat* may have a high value for the contextual topic pair describing animate subjects and edible objects. To determine the plausibility of a new subject-object pair for a particular verb, we calculate the Kronecker product of the subject and object noun vectors for this pair, and compare the resulting matrix with the average verb matrix using cosine similarity.

For label prediction, we calculate the similarity between each of the training data pairs and the learned average matrix. Unlike for **KKmat**, the **class cutoff** is estimated at the break-even point of the receiver operator characteristic (ROC) generated by comparing the training labels with this cosine similarity value. The break-even point is when the true positive rate is equal to the false positive rate. In practice it would be more accurate to estimate the cutoff on a validation dataset, but some of the verbs have so few training instances that this was not possible.

4 Experiments

In order to examine the quality of learning we run several experiments where we compare the different methods. In these experiments we consider the **DMat** method as the baseline. Some of the experiments employ cross-validation, in particular five repetitions of 2-fold cross validation (5x2cv), which has been shown to be statistically more robust than the traditional 10-fold cross validation (Alpaydin, 1999; Ulaş et al., 2012). The results of 5x2cv experiments can be compared using the regular paired t-test, but the specially designed 5x2cv F-test has been proven to produce fewer statistical errors (Ulaş et al., 2012).

The performance was evaluated using the area under the ROC (AUC) and the F_1 measure (based on precision and recall over the plausible class). The AUC evaluates whether a method is ranking positive examples above negative ones, regardless of the class cutoff value. F_1 shows how accurately a method assigns the correct class label. Another way to interpret the results is to consider the AUC as the measure of the quality of the parameters in the verb matrix or tensor, while the F-score indicates how well the softmax, the sigmoid, and the **DMat** cutoff algorithm are estimating class participation.

Ex-1. In the first experiment, we compare the different transitive verb representations by running 5x2cv experiments on ten verbs chosen to cover a range of concreteness and frequency values (Section 4.2).

Ex-2. In the initial experiments we found that some models had low performance, so we applied the column normalisation technique, which is often used with regression learning to standardise the numerical range of features:

$$\vec{x} := \frac{\vec{x} - \min(\vec{x})}{\max(\vec{x}) - \min(\vec{x})} \qquad (2)$$

This preserves the relative values of features between training samples, while moving the values to the [0,1] range.

Ex-3. There are varying numbers of training examples for each of the verbs, so we repeated the 5x2cv with datasets of 52 training points for each verb, since this is the size of the smallest dataset of the verb CENSOR. The points were randomly sampled from the datasets used in the first experiment. Finally, the four verbs with the largest datasets were used to examine how the performance of the methods changes as the amount of training data increases. The 4,000 training samples were randomised and half were used for testing. We sampled between 10 and 1000 training triples from the other half (Figure 4).

4.1 Noun vectors

Distributional semantic models (Turney and Pantel, 2010) encode word meaning in a vector format by counting co-occurrences with other words within a specified context window. We constructed the vectors from the October 2013 dump of Wikipedia articles, which was tokenised using the Stanford NLP tools[3], lemmatised with the Morpha lemmatiser (Minnen et al., 2001), and parsed with the C&C parser (Clark and Curran, 2007). In this paper we use sentence boundaries to define context windows and the top 10,000 most frequent lemmatised words in the whole corpus (excluding stopwords) as context words. The raw co-occurrence counts are re-weighted using the standard tTest weighting scheme (Curran, 2004), where $f_{w_i c_j}$ is the number of times target noun w_i occurs with context word c_j:

$$tTest(\vec{w_i}, c_j) = \frac{p(w_i, c_j) - p(w_i)p(c_j)}{\sqrt{p(w_i)p(c_j)}} \quad (3)$$

where $p(w_i) = \frac{\sum_j f_{w_i c_j}}{\sum_k \sum_l f_{w_k c_l}}$, $p(c_j) = \frac{\sum_i f_{w_i c_j}}{\sum_k \sum_l f_{w_k c_l}}$, and $p(w_i, c_j) = \frac{f_{w_i c_j}}{\sum_k \sum_l f_{w_k c_l}}$.

Using all 10,000 context words would result in a large number of parameters for each verb tensor, and so we apply singular value decomposition (SVD) (Turney and Pantel, 2010) with 40 latent dimensions to the target-context word matrix. We use context selection (with $N = 140$) and row normalisation as described in Polajnar and Clark (2014) to markedly improve the performance of SVD on smaller dimensions (K) and enable us to train the verb tensors using very low-dimensional noun vectors. Performance of the noun vectors was measured on standard word similarity datasets and the results were comparable to those reported by Polajnar and Clark (2014).

[3]http://nlp.stanford.edu/software/index.shtml

Verb	Concreteness	# of Positive	Frequency
APPLY	2.5	5618	47361762
CENSOR	3	26	278525
COMB	5	164	644447
DEPOSE	2.5	118	874463
EAT	4.44	5067	26396728
IDEALIZE	1.17	99	485580
INCUBATE	3.5	82	833621
JUSTIFY	1.45	5636	10517616
REDUCE	2	26917	40336784
WIPE	4	1090	6348595

Table 2: The 10 chosen verbs together with their concreteness scores. The number of positive SVO examples was capped at 2000. **Frequency** is the frequency of the verb in the GSN corpus.

4.2 Training data

In order to generate training data we made use of two large corpora: the Google Syntactic N-grams (GSN) (Goldberg and Orwant, 2013) and the Wikipedia October 2013 dump. We first chose ten transitive verbs with different concreteness scores (Brysbaert et al., 2013) and frequencies, in order to obtain a variety of verb types. Then the positive (plausible) SVO examples were extracted from the GSN corpus. More precisely, we collected all distinct syntactic trigrams of the form *nsubj ROOT dobj*, where the root of the phrase was one of our target verbs. We lemmatised the words using the NLTK[4] lemmatiser and filtered these examples to retain only the ones that contain nouns that also occur in Wikipedia, obtaining the counts reported in Table 2.

For every positive training example, we constructed a negative (implausible) one by replacing both the subject and the object with a *confounder*, using a standard technique from the selectional preference literature (Chambers and Jurafsky, 2010). A confounder was generated by choosing a random noun from the same frequency bucket as the original noun.[5] Frequency buckets of size 10 were constructed by collecting noun frequency counts from the Wikipedia corpus. For ex-

[4]http://nltk.org/

[5]Note that the random selection of the confounder could result in a plausible negative example by chance, but manual inspection of a subset of the data suggests this happens infrequently for those verbs which select strongly for their arguments, but more often for those verbs that don't.

Verb	Tensor	DMat	KKMat	SKMat	2Mat	Tensor	DMat	KKMat	SKMat	2Mat
	AUC					F_1				
APPLY	85.68†	81.46‡	88.88†‡	68.02	**88.92**†‡	79.27	64.00	**81.24**‡	54.06	80.80‡
CENSOR	79.40	**85.54**	80.55	78.52	83.19	70.66	47.93	**73.52**	37.86	71.07
COMB	89.41	85.65	88.38	69.20†‡	**89.56**	81.15	45.02	81.38	39.67	**82.36**
DEPOSE	92.70	**94.44**	93.12	84.47†	93.20	84.60	54.77	84.79	43.79	**86.15**
EAT	94.62	93.81	95.17	67.92	**95.88**‡	88.91	52.45	88.83	56.22	**89.95**
IDEALIZE	69.56	**75.84**	72.46	61.19	70.23	66.53	48.28	**68.39**	31.03	67.43
INCUBATE	89.33	85.53	88.61	70.59	**91.40**	80.30	50.84	80.90	31.99	**84.55**
JUSTIFY	85.27†	88.70‡	89.97‡	73.56	**90.10**‡	79.73	73.71	81.10	54.09	**82.52**
REDUCE	96.13	95.48	96.69†	79.32	**97.21**	91.24	71.24‡	87.46	76.67‡	**92.22**
WIPE	85.19	84.47	**87.84**†	64.93†‡	81.29	78.57	47.62	80.65	39.50	**78.90**
MEAN	86.93	87.29	**88.37**	71.96	88.30	80.30	55.79	81.03	46.69	**81.79**

Table 3: The best AUC and F_1 results for all the verbs, where † denotes statistical significance compared to **DMat** and ‡ denotes significance compared to **Tensor** according to the 5x2cv F-test with $p < 0.05$.

ample, for the plausible triple *animal* EAT *plant*, we generate the implausible triple *mountain* EAT *product*. Some verbs were well represented in the corpus, so we used up to the top 2,000 most frequent triples for training.

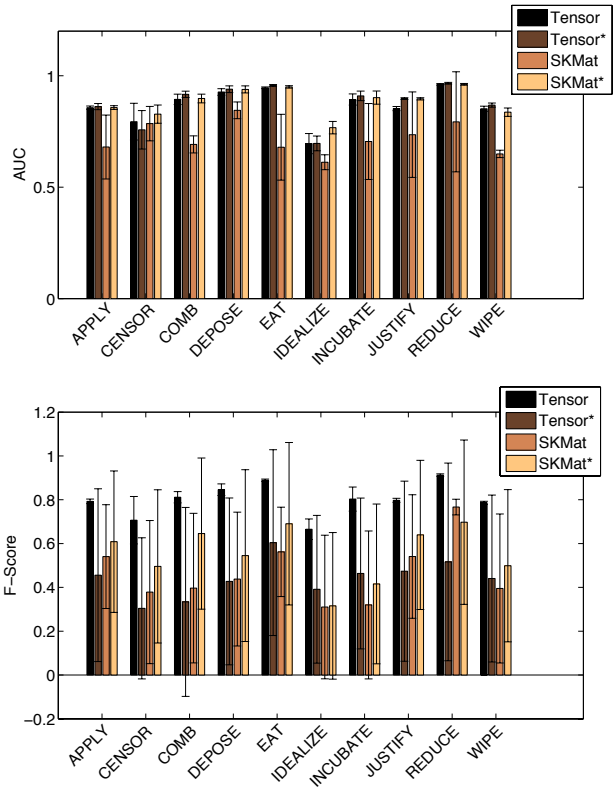


Figure 3: The effect of column normalisation (*) on **Tensor** and **SKMat**. Top table shows AUC and the bottom F_1-score, while the error bars indicate standard deviation.

5 Results

The results from **Ex-1** are summarised in Table 3. We can see that linear regression can lead to models that are able to distinguish between plausible and implausible SVO triples. The **Tensor** method outperforms **DMat**, which was previously shown to produce reasonable verb representations in related experiments (Grefenstette and Sadrzadeh, 2011). **2Mat** and **KKMat**, in turn, outperform **Tensor** demonstrating that it is possible to learn lower dimensional approximations of the tensor-based framework. **2Mat** is an appropriate approximation for functions with two inputs and a sentence space of any dimensionality, while **KKMat** is only appropriate for a single valued sentence space, such as the plausibility or sentiment space. Due to method variance and dataset size there are very few AUC results that are significantly better than **DMat** and even fewer that outperform **Tensor**. All methods perform poorly on the verb IDEALIZE, probably because it has the lowest concreteness value and is in one of the smallest datasets. This verb is also particularly difficult because it does not select strongly for either its subject or object, and so some of the pseudo-negative examples are in fact somewhat plausible (e.g. *town* IDEALIZE *authority* or *child* IDEALIZE *racehorse*). In general, this would indicate that more concrete verbs are easier to learn, as they have a clearer pattern of preferred property types, but there is no distinct correlation.

The results of the normalisation experiments (**Ex-2**) are shown in Table 4. We can see that the **SKMat** method, which performed poorly in **Ex-1** notably improves with normalisation. **Tensor** AUC scores also improve through normalisation, but the F-scores decrease. The rest of the methods, and in particular **DMat** are negatively affected by column normalisation. The results from **Ex-1** and **Ex-2** for **SKMat** and **Tensor** are summarised in

Verb	Tensor	DMat	KKMat	SKMat	2Mat	Tensor	DMat	KKMat	SKMat	2Mat
	AUC					F_1				
APPLY	**86.16**†	48.63‡	82.63†‡	85.73†	85.65†	45.57	46.99	46.17	60.86	**76.60**†
CENSOR	75.74	71.20	78.00	**82.77**	78.64	30.43	55.16	**65.19**	49.59	44.22
COMB	**91.67**†	62.42‡	90.85†	89.79†	91.42†	33.37	61.05	**71.20**	64.56	75.96
DEPOSE	**93.96**†	54.93‡	93.56†	93.87†	93.81†	42.73	39.71	**73.07**	54.51	56.54
EAT	**95.64**†	47.68‡	92.92†	94.99†‡	94.76†	60.42	47.42	58.80	69.05	**87.44**†
IDEALIZE	69.64	55.98	72.20†‡	**76.71**†‡	71.85†	39.14	49.16	41.75	31.57	**50.59**
INCUBATE	**90.97**†	61.31‡	89.69†	90.19†	90.05†	46.35	53.33	**70.45**	41.57	63.61
JUSTIFY	**89.76**†	54.87‡	87.26†‡	89.64†	89.05†	47.38	51.40	41.91	63.96	**80.55**†
REDUCE	**96.63**†	59.58‡	94.99†‡	96.14†	96.53†	51.63	54.27	69.18	69.76	**90.77**†
WIPE	**86.82**†	58.02‡	84.18†	83.65†	86.02†	44.04	55.19	47.84	49.89	**75.80**
MEAN	87.90	57.66	86.83	**88.55**	87.98	44.31	51.57	58.76	55.73	**70.41**

Table 4: The best AUC and F_1 results for all the verbs with normalised vectors, where † denotes statistical significance compared to **DMat** and ‡ denotes significance compared to **Tensor** according to the 5x2cv F-test with $p < 0.05$.

Figure 3. This figure also shows that AUC values have much lower variance, but that high variance in F-score leads to results that are not statistically significant.

When considering the size of the datasets (**Ex-3**), it would seem from Table 5 that **2Mat** is able to learn from less data than **DMat** or **Tensor**. While this may be true over a 5x2cv experiment on small data, Figure 4 shows that this view may be overly simplistic and that different training examples can influence learning. Analysis of errors shows that the baseline method mostly generates false negative errors (i.e. predicting implausible when the gold standard label is plausible). In contrast, **Tensor** produces almost equal numbers of false positives and false negatives, but sometimes produces false negatives with low frequency nouns (e.g. *bourgeoisie* IDEALIZE *work*), presumably because there is not enough information in the noun vector to decide on the correct class. It also produces some false positive errors when either of the nouns is plausible (but the triple is implausible), which would suggest results may be improved by training with data where only one noun is confounded or by treating negative data as possibly positive (Lee and Liu, 2003).

Verb	Tensor	DMat	2Mat
APPLY	**95.76**	86.50	86.31
CENSOR	82.97	**84.09**	77.79
COMB	90.13	92.93	**95.18**
DEPOSE	92.41	91.27	**95.61**
EAT	**99.64**	98.25	99.58
IDEALIZE	75.03	76.68	**88.98**
INCUBATE	91.10	87.20	**96.42**
JUSTIFY	88.96	**88.99**	87.31
REDUCE	**100.0**	99.87	99.46
WIPE	**97.20**	91.63	96.36
MEAN	91.52	89.94	**92.50**

Table 5: Results show average of 5x2cv AUC on small data (26 positive + 26 negative per verb). None of the results are significant.

6 Discussion

Current methods which derive distributed representations for phrases, for example the work of Socher et al. (2012), typically use only matrix representations, and also assume that words, phrases and sentences all live in the same vector space. The tensor-based semantic framework is more flexible, in that it allows different spaces for different grammatical types, which results from it being tied more closely to a type-driven syntactic description; however, this flexibility comes at a cost, since there are many more paramaters to learn.

Various communities are beginning to recognize the additional power that tensor representations can provide, through the capturing of interactions that are difficult to represent with vectors and matrices (see e.g. (Ranzato et al., 2010; Sutskever et al., 2009; Van de Cruys et al., 2012)). Hierarchical recursive structures in language potentially represent a large number of such interactions – the obvious example for this paper being the interaction between a transitive verb's subject and object – and present a significant challenge for machine learning.

This paper is a practical extension of the work in Krishnamurthy and Mitchell (2013), which introduced learning of CCG-based function tensors with logistic regression on a compositional semantics task, but was implemented as a proof-of-concept with vectors of length 2 and on small, manually created datasets based on propositional

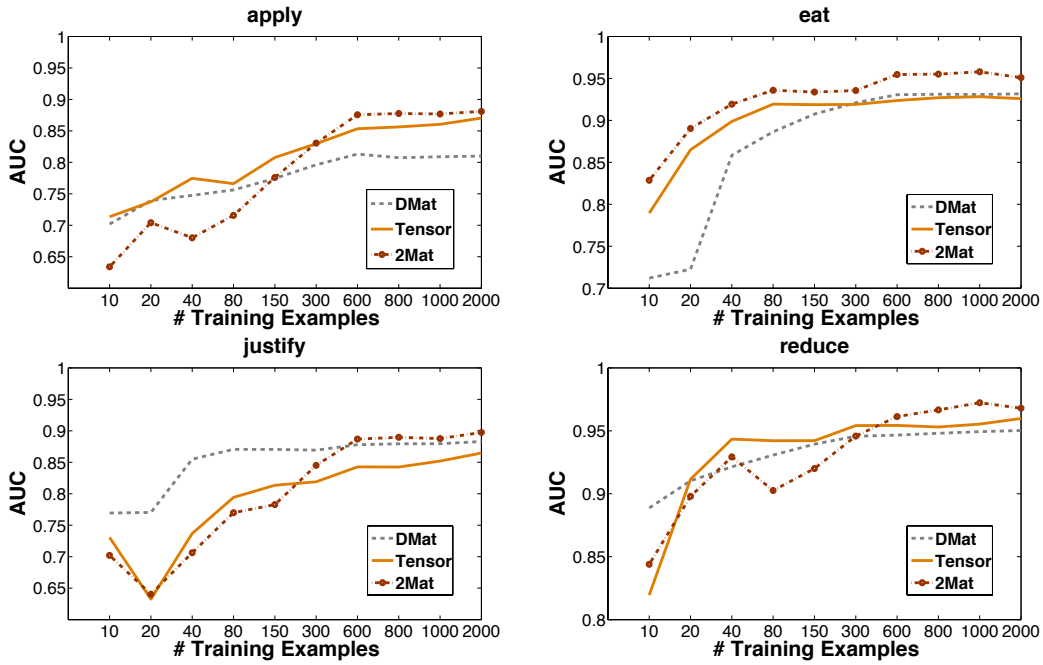


Figure 4: Comparison of **DMat**, **Tensor**, and **2Mat** methods as the number of training instances increases.

logic examples. Here, we go beyond this by learning tensors using corpus data and by deriving several different matrix representations for the verb in the subject-verb-object (SVO) sentence.

This work can also be thought of as applying neural network learning techniques to the classic problem of selectional preference acquisition, since the design of the pseudo-disambiguation experiments is taken from the literature on selectional preferences (Clark and Weir, 2002; Chambers and Jurafsky, 2010). We do not compare directly with methods from this literature, e.g. those based on WordNet (Resnik, 1996; Clark and Weir, 2002) or topic modelling techniques (Seaghdha, 2010), since our goal in this paper is not to extend the state-of-the-art in that area, but rather to use selectional preference acquisition as a test bed for the tensor-based semantic framework.

7 Conclusion

In this paper we introduced three dimensionally reduced representations of the transitive verb tensor defined in the type-driven framework for compositional distributional semantics (Coecke et al., 2010). In a comprehensive experiment on ten different verbs we find no significant difference between the full tensor representation and the reduced representations. The **SKMat** and **2Mat** representations have the lowest number of parameters and offer a promising avenue of research for more complex sentence structures and sentence spaces. **KKMat** and **DMat** also had high scores on some verbs, but these representations are applicable only in spaces where a single-value output is appropriate.

In experiments where we varied the amount of training data, we found that in general more concrete verbs can learn from less data. Low concreteness verbs require particular care with dataset design, since some of the seemingly random examples can be plausible. This problem may be circumvented by using semi-supervised learning techniques.

We also found that simple numerical techniques, such as column normalisation, can markedly alter the values and quality of learning. On our data, column normalisation has a side-effect of removing the negative values that were introduced by the use of tTest weighting measure. The use of the PPMI weighting scheme and non-negative matrix factorisation (NMF) (Grefenstette et al., 2013; Van de Cruys, 2010) could lead to a similar effect, and should be investigated. Further numerical techniques for improving the estimation of the class decision boundary, and consequently the F-score, will also constitute future work.

References

Ethem Alpaydin. 1999. Combined 5x2 CV F-test for comparing supervised classification learning algorithms. *Neural Computation*, 11(8):1885–1892, November.

Brett W. Bader, Tamara G. Kolda, et al. 2012. Matlab tensor toolbox version 2.5. Available online, Jan.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*, pages 1183–1193, Cambridge, MA.

Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. 2014. Frege in space: A program for compositional distributional semantics. *Linguistic Issues in Language Technology*, 9:5–110.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2013. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior research methods*, pages 1–8.

Nathanael Chambers and Dan Jurafsky. 2010. Improving the use of pseudo-words for evaluating selectional preferences. In *Proceedings of ACL 2010*, Uppsala, Sweden.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Stephen Clark and David Weir. 2002. Class-based probability estimation using a semantic hierarchy. *Computational Linguistics*, 28(2):187–206.

Stephen Clark. 2013. Type-driven syntax and semantics for composing meaning vectors. In Chris Heunen, Mehrnoosh Sadrzadeh, and Edward Grefenstette, editors, *Quantum Physics and Linguistics: A Compositional, Diagrammatic Discourse*, pages 359–377. Oxford University Press.

Daoud Clarke. 2012. A context-theoretic framework for compositionality in distributional semantics. *Computational Linguistics*, 38(1):41–71.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. In J. van Bentham, M. Moortgat, and W. Buszkowski, editors, *Linguistic Analysis (Lambek Festschrift)*, volume 36, pages 345–384.

James R. Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh.

David R. Dowty, Robert E. Wall, and Stanley Peters. 1981. *Introduction to Montague Semantics*. Dordrecht.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of English books. In *Second Joint Conference on Lexical and Computational Semantics*, pages 241–247, Atlanta,Georgia.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404, Edinburgh, Scotland, UK, July.

Edward Grefenstette, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. 2013. Multi-step regression learning for compositional distributional semantics. *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)*.

Edward Grefenstette. 2013. *Category-Theoretic Quantitative Compositional Distributional Models of Natural Language Semantics*. Ph.D. thesis, University of Oxford.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Jayant Krishnamurthy and Tom M Mitchell. 2013. Vector space semantic parsing: A framework for compositional vector space models. In *Proceedings of the 2013 ACL Workshop on Continuous Vector Space Models and their Compositionality*, Sofia, Bulgaria.

Wee Sun Lee and Bing Liu. 2003. Learning with positive and unlabeled examples using weighted logistic regression. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*.

Jean Maillard, Stephen Clark, and Edward Grefenstette. 2014. A type-driven tensor-based semantics for CCG. In *Proceedings of the EACL 2014 Type Theory and Natural Language Semantics Workshop (TTNLS)*, Gothenburg, Sweden.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08*, pages 236–244, Columbus, OH.

Tamara Polajnar and Stephen Clark. 2014. Improving distributional semantic vectors through context selection and normalisation. In *14th Conference of the European Chapter of the Association for Computational Linguistics, EACL'14*, Gothenburg, Sweden.

M. Ranzato, A. Krizhevsky, and G. E. Hinton. 2010. Factored 3-way restricted boltzmann machines for modeling natural images. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, Sardinia, Italy.

Philip Resnik. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, 61:127–159.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–124.

Diarmuid O Seaghdha. 2010. Latent variable models of selectional preference. In *Proceedings of ACL 2010*, Uppsala, Sweden.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1201–1211, Jeju, Korea.

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

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

I. Sutskever, R. Salakhutdinov, and J. B. Tenenbaum. 2009. Modelling relational data using bayesian clustered tensor factorization. In *Proceedings of Advances in Neural Information Processing Systems (NIPS 2009)*, Vancouver, Canada.

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

Aydın Ulaş, Olcay Taner Yıldız, and Ethem Alpaydın. 2012. Cost-conscious comparison of supervised learning algorithms over multiple data sets. *Pattern Recognition*, 45(4):1772–1781, April.

Tim Van de Cruys, Laura Rimell, Thierry Poibeau, and Anna Korhonen. 2012. Multi-way tensor factorization for unsupervised lexical acquisition. In *Proceedings of COLING 2012*, Mumbai, India.

Tim Van de Cruys. 2010. A non-negative tensor factorization model for selectional preference induction. *Journal of Natural Language Engineering*, 16(4):417–437.

An Etymological Approach to Cross-Language Orthographic Similarity. Application on Romanian

Alina Maria Ciobanu, Liviu P. Dinu
Faculty of Mathematics and Computer Science, University of Bucharest
Center for Computational Linguistics, University of Bucharest
alina.ciobanu@my.fmi.unibuc.ro, ldinu@fmi.unibuc.ro

Abstract

In this paper we propose a computational method for determining the orthographic similarity between Romanian and related languages. We account for etymons and cognates and we investigate not only the number of related words, but also their forms, quantifying orthographic similarities. The method we propose is adaptable to any language, as far as resources are available.

1 Introduction

Language relatedness and language change across space and time are two of the main questions of the historical and comparative linguistics (Rama and Borin, 2014). Many comparative methods have been used to establish relationships between languages, to determine language families and to reconstruct their proto-languages (Durie and Ross, 1996). If grouping of languages in linguistic families is generally accepted, the relationships between languages belonging to the same family are periodically investigated. In spite of the fact that linguistic literature abounds in claims of classification of natural languages, the degrees of similarity between languages are far from being certain. In many situations, the similarity of natural languages is a fairly vague notion, both linguists and non-linguists having intuitions about which languages are more similar to which others. McMahon and McMahon (2003) and Rama and Borin (2014) note that the computational historical linguistics did not receive much attention until the beginning of the 1990s, and argue for the necessity of development of quantitative and computational methods in this field.

1.1 Related Work

According to Campbell (2003), the methods based on comparisons of cognate lists and sound correspondences are the most popular approaches employed for establishing relationships between languages. Barbançon et al. (2013) emphasize the variety of computational methods used in this field, and state that the differences in datasets and approaches cause difficulties in the evaluation of the results regarding the reconstruction of the phylogenetic tree of languages. Linguistic phylogeny reconstruction proves especially useful in historical and comparative linguistics, as it enables the analysis of language evolution. Ringe et al. (2002) propose a computational method for evolutionary tree reconstruction based on a "perfect phylogeny" algorithm; using a Bayesian phylogeographic approach, Alekseyenko et al. (2012), continuing the work of Atkinson et al. (2005), model the expansion of the Indo-European language family and find support for the hypothesis which places its homeland in Anatolia; Atkinson and Gray (2006) analyze language divergence dates and argue for the usage of computational phylogenetic methods in the question of Indo-European age and origins. Using modified versions of Swadesh's lists[1], Dyen et al. (1992) investigate the classification of Indo-European languages by applying a lexicostatistical method.

The similarity of languages is interesting not only for historical and comparative linguistics, but for machine translation and language acquisition as well. Scannell (2006) and Hajič et al. (2000) argue for the possibility of obtaining a better translation quality using simple methods for very closely related languages. Koppel and Ordan (2011) study the impact of the distance between languages on the translation product and conclude that it is directly correlated with the ability to distinguish translations from a given source language from non-translated text. Some genetically related languages are so similar to each other, that

[1] http://www.wordgumbo.com/ie/cmp/iedata.txt

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1047–1058,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

speakers of such languages are able to communicate without prior instruction (Gooskens, 2007). Gooskens et al. (2008) analyze several phonetic and lexical predictors and their conclusion is that lexical similarity can be seen as a predictor of language intelligibility. The impact of language similarities in the process of second language acquisition is argued by the contrastive analysis hypothesis, which claims that where similarities between the first and the second language occur, the acquisition would be easier compared with the situation in which there were differences between the two languages (Benati and VanPatten, 2011).

1.2 Our Approach

Although there are multiple aspects that are relevant in the study of language relatedness, such as the orthographic, phonetic, syntactic and semantic differences, in this paper we focus only on the orthographic similarity. The orthographic approach relies on the idea that sound changes leave traces in the orthography, and alphabetic character correspondences represent, to a fairly large extent, sound correspondences (Delmestri and Cristianini, 2010).

In this paper we propose an orthographic similarity method focused on etymons (direct sources of the words in a foreign language) and cognates (words in different languages having the same etymology and a common ancestor). In a broadly accepted sense, the higher the similarity degree between two languages, the closer they are.

One of our motivations is that when people encounter a language for the first time in written form, it is most likely that they can distinguish and individualize words which resemble words from their native language. These words are probably either inherited from their mother tongue (etymons), or have a common ancestor with the words in their language (cognates).

Our first goal is, given a corpus C, to automatically detect etymons and cognates. In Section 2 we propose a dictionary-based approach to automatically extract related words, and a method for computing the orthographic similarity of natural languages. Most of the traditional approaches in this field focus either on etymology detection or on cognate identification, most of them reporting results only on small sets of cognate pairs (usually manually determined lists of about 200 cognates, for which the cognate judgments are made by human experts (Rama and Borin, 2014)). Our approach implies a detailed investigation which accounts not only for the number of related words, as it is usually done in lexicostatistics (where the relationships between languages are determined based on the percentage of related words), but also for their forms, quantifying orthographic similarities. We employ three string similarity metrics for a finer-grained analysis, as related words in different languages do not have identical forms and their partial similarity implies different degrees of recognition and comprehensibility. For example, the Romanian word *lună* (*moon*) is closer to its Latin etymon *luna* than the word *bătrân* (*old*) to its etymon *veteranus*, and the Romanian word *vânt* (*wind*) is closer to its French cognate pair *vent* than the word *castel* (*castle*) to its cognate pair *château*.

In this paper we investigate the orthographic similarity between Romanian and related languages. Romanian is a Romance language, belonging to the Italic branch of the Indo-European language family, and is of particular interest regarding its geographic setting. It is surrounded by Slavic languages and its relationship with the big Romance kernel was difficult. Besides general typological comparisons that can be made between any two or more languages, Romanian can be studied based on comparisons of genetic and geographical nature, participating in numerous areally-based similarities that define the Balkan convergence area. Joseph (1999) states that, regarding the genetic relationships, Romanian can be studied in the context of those languages most closely related to it and that the well-studied Romance languages enable comparisons that might not be possible otherwise, within less well-documented families of languages. The position of Romanian within the Romance family is controversial (McMahon and McMahon, 2003): either marginal or more integrated within the group, depending on the versions of the cognate lists that are used in the analysis.

In Section 3.1 we apply our method on Romanian in different stages of its evolution, running our experiments on high-volume corpora from three historical periods: the period approximately between 1642 and 1743, the second half of 19th century (1870 - 1889), and the present period. In Section 3.2 we make use of a fourth corpus, Europarl, with a double goal: on the one hand, to check if degrees of similarity between Romanian

and other languages in the present period are consistent across two different corpora, and on the other hand, to investigate whether there are differences between the overall degrees of similarity obtained for the entire corpus and those obtained in various experiments at sentence level. The conclusions of our paper are outlined in Section 4.

2 Methodology and Algorithm

In this section we introduce a technique for determining the orthographic similarity of languages. In order to obtain accurate results, we investigate both etymons and cognates. First, we automatically identify etymons and cognates, then we measure the distances between related words, and finally we compute the overall degrees of similarity between pairs of languages. We also applied this method for investigating the mutual intelligibility of the Romance languages, and preliminary results are presented in (Ciobanu and Dinu, 2014b).

2.1 Similarity Method

Let $C = \{w_1, w_2, ..., w_{N_{words}}\}$ be a corpus in L_1 and let L_2 be a related language. We assume, without any loss of generality, that the elements of C are ordered such that $C_L = \{w_1, w_2, ..., w_{N_{lingua}}\}$ is the subset of C containing all the words that have an etymon or a cognate pair in L_2. We use the following notations: N_{words} is the number of token words in C, N_{lingua} is the number of token words in C_L, λ is the empty string and x_i is the etymon or cognate pair of w_i in L_2. Given a string distance Δ, we define the distance between L_1 and L_2 (non-metric distance), with frequency support from corpus C, as follows:

$$\Delta(L_1, L_2) = 1 - \frac{N_{lingua}}{N_{words}} + \frac{\sum_{i=1}^{N_{lingua}} \Delta(w_i, x_i)}{N_{words}} \quad (1)$$

Hence, the similarity between languages L_1 and L_2 is defined as follows:

$$Sim(L_1, L_2) = 1 - \Delta(L_1, L_2) \quad (2)$$

2.2 Algorithm

We present here the algorithm based on linguistic relationships detection and string similarity methods for determining the orthographic similarity between languages, with frequency support from corpora in the source language. This algorithm, represented in Figure 2, is applicable to any language. After a preprocessing phase, which is detailed in Subsection 2.2.1, we analyze words and begin by identifying their etymologies.

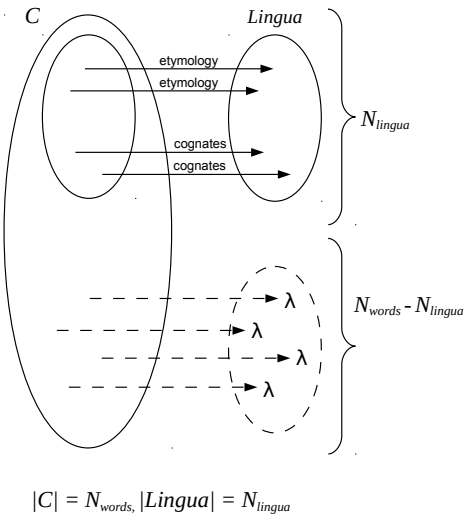


Figure 1: Schema for determining the orthographic similarity between related languages with frequency support from corpus *C*.

Corpus	#words		#stop words		#lemmas
	token	type	token	type	type
Parliament	22,469,290	162,399	14,451,178	214	40,065
Eminescu	870,828	65,742	565,396	212	21,456
Chronicles	253,786	28,936	170,582	193	8,189
RVR	2,464	2,464	124	124	2,252

Table 1: Statistics for the Romanian datasets.

2.2.1 Preprocessing

Given a corpus C, we start by preprocessing the text.

Step 1. Data Cleaning. We perform basic word segmentation, using whitespace and punctuation marks as delimiters and we lower-case all words. We remove from the datasets tokens that are irrelevant for our investigation, such as dates, numbers and non-textual annotations marked by non-alphanumeric characters.

Step 2. Stop Words Removal. We focus on analyzing word content and, in order to obtain relevant results, we remove stop words from the datasets. We use the lists of stop words for Romanian provided by the Apache Lucene[2] text search engine library. In Table 1 we list the total number of stop words from each corpus.

[2]http://lucene.apache.org

Step 3. Lemmatization. We use lemmas for identifying words' definitions in dictionaries and for computing adequate distances between words and their cognates or etymons. We use the Dexonline[3] machine-readable dictionary to lemmatize Romanian words.

Step 4. Diacritics Removal. Many words have undergone transformations by the augmentation of language-specific diacritics when entering a new language. From an orthographic perspective, the resemblance of words is higher between words without diacritics than between words with diacritics. For example, the orthographic distance is higher for the Romanian word *amiciţie* (*friendship*) and its French cognate pair *amitié* than for their corresponding forms without diacritics, *amicitie* and *amitie*. For this reason, in this step of our procedure we create two versions of each dataset, with and without diacritics, in order to further investigate the influence of the diacritics on the cross-language orthographic similarity. In Romanian, 5 diacritics are used today: *ă, î, â, ş, ţ*.

2.2.2 Relationships Identification

Step 1. Etymology Detection. For most words, etymological dictionaries offer a unique etymology, but when more options are possible for explaining a word's etymology (there are words whose etymology was and remains difficult to ascertain), dictionaries may provide multiple alternatives. For example, the Romanian word *parlament* (*parliament*) has a double etymology: French (with the etymon *parlement*) and Italian (with the etymon *parlamento*). We account for all the given etymological hypotheses, enabling our method to provide more accurate results.

For determining words' etymologies we use the Dexonline machine-readable dictionary, which is an aggregation of over 30 Romanian dictionaries. By parsing its definitions, we are able to automatically extract information regarding words' etymologies and etymons. The most frequently used pattern is shown below.

```
<abbr class="abbrev"
    title="limba language_name">
    language_abbreviation </abbr>
<b> origin_word </b>
```

As an example, we provide below an excerpt from a Dexonline entry which uses this pattern to specify the etymology of the Romanian word *capitol* (*chapter*), which has double etymology: Latin (with the etymon *capitulum*) and Italian (with the etymon *capitolo*).

```
<b> CAPÍTOL </b>
<abbr class="abbrev"
    title="limba italiana"> it. </abbr>
<b> capitolo </b>
<abbr class="abbrev"
    title="limba latina"> lat. </abbr>
<b> capitulum </b>
```

Step 2. Cognate Identification. Cognates are words in different languages having the same etymology and a common ancestor. The methods for cognate detection proposed so far are mostly based on orthographic/phonetic and semantic similarities (Kondrak, 2001; Frunza et al., 2005), but the term "cognates" is often used with a somewhat different meaning, denoting words with high orthographic/phonetic and cross-lingual meaning similarity, the condition of common etymology being left aside. We focus on etymology and we introduce an automatic strategy for detecting pairs of

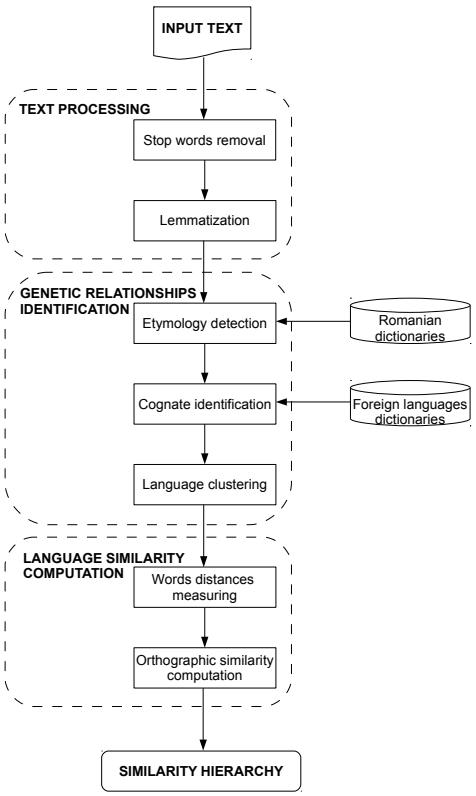


Figure 2: Algorithm for determining the orthographic similarity between related languages.

[3] http://dexonline.ro

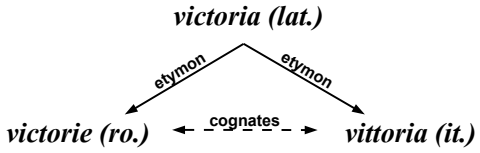


Figure 3: Word-etymon and cognate pairs.

cognates between two given languages, which enables the identification of all cognate pairs for the studied corpus.

Considering a set of words in a given language *L*, to identify the cognate pairs between *L* and a related language *L'*, we first determine the etymologies of the given words. Then we translate in *L'* all words without *L'* etymology. We consider cognate candidates pairs formed of input words and their translations. Using electronic dictionaries, we extract etymology-related information for the translated words. To identify cognates, we compare, for each pair of candidates, the etymologies and the etymons. If they match, we identify the words as being cognates.

In our previous work (Ciobanu and Dinu, 2014a) we applied this method on a Romanian dictionary, while here we extract cognates from Romanian corpora. We identify cognate pairs between Romanian and six other languages: Italian, French, Spanish, Portuguese, Turkish and English. We use electronic dictionaries[4] to extract etymology-related information and Google Translate[5] to translate Romanian words. We are restricted in our investigation by the available resources, but we plan to extend our method to other related languages as well. We selected these six languages for the following reason: the first four in our list are Romance languages, and our intuition is that there are numerous words in these languages which share a common ancestor with Romanian words. We investigate the cognate pairs for Turkish because many French words were imported in both Romanian and Turkish in the 19th century, and we believe that accounting for Romanian-Turkish cognates would provide a more accurate result for the similarity of these languages. As for English, we decided to investigate the cognate pairs for this language in order to analyze to what extent the influence of English on Romanian increases across time. In Table 2 we report the number of Romanian words having an etymon or a cognate pair in the six related languages.

Step 3. Evaluation. In order to evaluate our automatic method for extracting etymology-related information and for detecting related words, we excerpt a sample of 500 words for each of the considered languages (Romanian, French, Italian, Spanish, Portuguese, Turkish and English). The samples are drawn using a proportionate stratification sampling method with regard to the length of the lemmas in our datasets. We manually determine the etymologies of the words in the samples, and we compare these results with the automatically obtained etymologies. We compute the accuracy for etymology extraction for each language, and we obtain the following results: 95.8% accuracy for Romanian, 97.8% for Italian, 96.8% for French, 96.6% for Spanish, 97.0% for Portuguese, 96.0% for Turkish and, finally, 97.2% for English.

Language	Relationship	Corpus			
		Parliament	Eminescu	Chronicles	RVR
French	cognates	192,275	13,074	3,139	43
	etymons	15,665,865	484,668	89,946	1,203
Italian	cognates	1,660,588	40,491	2,743	100
	etymons	9,234,710	348,948	77,633	957
Spanish	cognates	4,616,528	119,627	9,942	355
	etymons	4,411,707	212,106	65,336	482
Portuguese	cognates	4,378,354	115,309	15,755	324
	etymons	3,477,285	156,908	55,991	435
Turkish	cognates	1,401,569	33,070	2,332	113
	etymons	331,863	24,115	11,985	69
English	cognates	4,347,302	146,377	21,966	296
	etymons	625,596	17,328	6,799	56

Table 2: Number of Romanian token words having etymons or cognate pairs in related languages.

2.2.3 Linguistic Distances

Various approaches have been previously employed for assessing the orthographic distance or similarity between related words. Their performance has been investigated and compared (Frunza et al., 2005; Rama and Borin, 2014), but a clear conclusion cannot be drawn with respect to which method is the most appropriate for a given task. We employ three metrics to determine the orthographic similarity between related words. In Subsection 3.1.2 we investigate to what extent the similarity scores computed with each of these metrics differ, and whether the differences are statistically significant. We use the following metrics:

[4] Italian: http://www.sapere.it/sapere/dizionari
French: http://www.cnrtl.fr
Spanish: http://lema.rae.es/drae
Portuguese: http://www.infopedia.pt/lingua-portuguesa
Turkish: http://www.nisanyansozluk.com
English: http://www.collinsdictionary.com

[5] http://translate.google.com

- **LCSR:** The longest common subsequence ratio (Melamed, 1995) is the longest common subsequence of two strings u and v divided by the length of the longer string. We subtract this value from 1, in order to obtain the distance between two words.

- **EDIT:** The edit distance (Levenshtein, 1965) counts the minimum number of operations (insertion, deletion and substitution) required to transform one string into another. We use a normalized version of the edit distance, dividing it by the length of the longer string.

- **RD:** The rank distance (Dinu and Dinu, 2005) is used to measure the similarity between two ranked lists. A ranking of a set of n objects can be represented as a permutation of the integers $1, 2, ..., n$. Let S be a set of ranking results, $\sigma \in S$. $\sigma(i)$ represents the rank of object i in the ranking result σ. The rank distance is computed as: $RD(\sigma, \tau) = \sum_{i=1}^{n} |\sigma(i) - \tau(i)|$. The ranks of the elements are assigned from bottom up, i.e. from n to 1, using the Borda count method (de Borda, 1781). The elements which do not occur in any of the rankings receive the rank 0. To extend the rank distance to strings, we index each occurrence of a given letter a with a_k, where k is the number of its previous occurrences, and then apply the rank distance on the new indexed strings, which become rankings in this situation. In order to normalize it, we divide the rank distance by the maximum possible distance between two strings u and v (Dinu and Sgarro, 2006): $\Delta_{max}(u, v) = |u|(|u|+1)/2+|v|(|v|+1)/2$.

3 Experiments and Results

In this section we present the results obtained by applying our method for determining orthographic similarity on Romanian datasets.

To our knowledge, only basic lexicostatistical methods (generally based on different dictionaries or versions of the representative vocabulary of Romanian) which compute the percentage of words with a given etymology have been applied for determining the relationships between Romanian and related languages. Because of the difficulty of setting the bounds between the basic lexicon and the remaining words, Graur (1968) uses in his experiments three concentric versions of the basic Romanian lexicon. Dinu (1996) reevaluates the etymology detection for the three versions of the basic Romanian lexicon and reclassifies the lexical material. He argues against grouping together all the words with Slavic origins, without differentiation between Old Slavic and languages such as Bulgarian, Russian, Ukrainian and Polish. Sala (1988) builds a version of the representative vocabulary of Romanian comprising 2588 words, which we use in our experiments as well.

3.1 Romanian Evolution

We apply our similarity method on high-volume Romanian corpora from three distinct historical periods of time, with different cultural, economical, political and social contexts. In Table 1 we report statistics for these corpora and for the basic Romanian lexicon.

3.1.1 Data

The first corpus consists of the transcription of the parliamentary debates held in the Romanian Parliament from 1996 to 2007 (Grozea, 2012). The second corpus consists of the publishing works of Mihai Eminescu (Eminescu, 1980 1985), the leading Romanian poet. His works provide an insightful description of the period between 1870 and 1889, with respect to its cultural, economical, social and political aspects, including some major events in the Romanian history. Many researchers consider that Eminescu had a crucial influence on Romanian, his contribution to modern language development being highly appreciated. The third corpus dates back to the period approximately between 1642 and 1743, the beginning period of the Romanian writing. Miron Costin, Grigore Ureche and Ion Neculce are Romanian chroniclers whose main works follow one another in creating one of the most detailed and valuable descriptions of Moldavia in that period, "Letopiseţul Ţării Moldovei". Along with them, Dimitrie Cantermir contributed to the early development of the Romanian writing, having written what is considered to be the first attempt at a socio-political novel ("Istoria Ieroglifică", 1703-1705). Their chronicles account for social, cultural, economical and political events with the purpose of recreating historical periods of time. We also use the basic Romanian lexicon (Sala, 1988), abbreviated RVR, for our experiments. The Dexonline machine-readable dictionary, which we use for determining the etymologies for the Romanian words, aggregates defini-

Language	Parliament					Eminescu					Chronicles					RVR				
	%words	D		ND		%words	D		ND		%words	D		ND		%words	D		ND	
French	70.6	45.5	46.0	48.3	48.8	57.2	35.2	36.1	37.2	38.2	36.7	20.3	21.1	22.3	23.1	50.6	30.3	31.4	32.2	33.3
Latin	63.7	40.2		42.0		59.9	34.6		36.6		44.9	24.2		25.7		56.5	34.0		37.3	
Italian	48.5	28.1	33.4	29.1	34.5	44.7	26.9	30.2	27.9	31.2	31.7	19.6	20.3	20.7	21.4	41.4	23.4	26.2	25.2	28.0
Spanish	40.2	9.2	24.9	10.7	27.0	38.1	10.9	21.2	12.9	23.7	29.7	11.9	15.1	13.9	17.2	32.5	9.0	19.5	9.9	21.0
Portuguese	35.0	8.3	22.1	9.5	24.0	31.3	9.6	18.5	11.3	21.0	28.3	12.2	16.3	13.9	18.4	29.3	8.6	17.4	9.4	18.9
English	22.1	2.2	14.0	2.2	14.2	18.8	1.1	9.9	1.2	10.1	11.3	1.3	5.9	1.3	6.2	14.3	1.6	10.3	1.6	10.4
Provencal	17.7	9.6		9.8		20.7	11.3		11.6		21.8	13.0		13.4		16.8	9.7		10.5	
German	9.2	5.8		5.9		6.9	4.5		4.6		4.9	2.4		2.4		10.2	6.3		6.6	
Turkish	7.7	0.9	5.4	0.9	5.6	6.6	1.7	4.5	1.7	4.7	5.6	2.9	3.7	3.1	3.9	7.4	1.6	5.0	1.8	5.3
Russian	5.9	3.7		4.0		6.5	4.0		4.4		7.5	4.3		4.9		9.0	5.4		6.2	
Catalan	5.9	3.3		3.4		9.0	4.8		5.1		11.2	5.9		6.4		8.4	4.6		4.9	
Greek	4.8	2.9		3.0		6.0	3.6		3.7		4.5	2.6		2.7		4.6	2.5		2.6	
Albanian	4.8	2.6		3.0		6.7	3.7		4.0		9.1	4.9		5.3		8.4	4.2		4.8	
Bulgarian	4.0	2.6		3.0		7.4	4.7		5.5		10.6	6.8		7.8		11.8	7.2		8.4	
Slavic	4.9	2.3		2.5		6.6	3.4		3.8		12.1	6.5		7.7		9.8	5.0		5.7	
Old Slavic	3.8	2.2		2.7		6.1	3.3		4.3		11.9	6.8		8.7		9.5	5.2		6.0	
Hungarian	2.9	1.8		2.0		5.1	2.9		3.3		7.5	4.3		4.7		7.4	3.7		4.6	
Ruthenian	2.4	1.6		2.0		4.7	3.0		3.7		6.0	3.7		4.4		4.5	2.4		3.0	
Serbian	2.6	1.4		1.6		5.8	3.0		3.4		8.9	5.0		5.5		8.6	5.2		6.0	
Sardinian	1.7	1.0		1.0		3.3	1.7		1.8		4.0	2.0		2.1		2.6	1.4		1.5	

Table 3: Results for the Romanian datasets. In the *D* and *ND* columns we provide the average degrees of similarity for the datasets with and without diacritics. For languages for which we determine cognate pairs (besides etymons), we report both versions of the results, before and after cognate identification. In the *%words* column we provide the percentage of words having an etymon or a cognate pair in each language. The results are ordered according to the ranking of similarity for the corpus comprising the parliamentary debates after identifying cognates and with diacritics included.

tions from over 30 dictionaries ranging from 1927 to the present time and contains archaisms and obsolete words (which are marked accordingly); therefore, we are able to identify etymologies for words in all used corpora.

3.1.2 Results

In this subsection we present and analyze the main results drawn from our research. In Table 3 we list the output of our method for each corpus, with and without diacritics[6]. We report the similarity between Romanian and related languages, providing the average value of the three metrics used. In the *%words* column we provide, for each corpus, the percentage of words having an etymon or a cognate pair in a given language (the typical measure used in lexicostatistical comparison, i.e., the 0 distance function). The results for the Romanian datasets are plotted in Figure 4 and Figure 5.

Cognate Influence. Table 3 and Figure 4 present the gain obtained by cognate analysis. Accounting for cognates leads to an increase of similarity between Romanian and Spanish and Portuguese with almost 300%, and between Romanian and Italian with almost 20%. Another spectacular increase of closeness is for Turkish, which draws closer to Romanian with more than 500% by using the cognate gain. The degree of similarity is not given by the contribution of words inherited in Romanian from Turkish (about 1%), but by the pairs of shared cognates. Both Romanian and Turkish borrowed words from French massively towards the end of the 19th century. Thus, most pairs of Romanian-Turkish cognates have common French ancestors, and words in Romanian and Turkish which resemble are actually loans from the same French words. We also notice a significant increase in similarity between Romanian and English in the modern period. This increase is natural and probably arises for the similarity between English and most of the other languages as well. We notice that this increase is due preponderantly to the cognate pairs. Most of the Romanian-English cognates have a Romance common ancestor (78.4% Latin, 4.2% French, 3.4% Italian), and 11.8% have a Greek common ancestor, counted at lemma level on the corpus comprising the parliamentary debates.

Romanian Evolution. Some significant results can be observed in the evolution of Romanian: the degrees of similarity between Romanian and all the Romance languages has increased significantly from the Chronicles period until today. Besides them, German is the only language to which Romanian drew closer (a possible explanation might be the fact that, after the establishment of Germans

[6]The complete ranking of similarity is available online at http://nlp.unibuc.ro/resources/rosim.pdf.

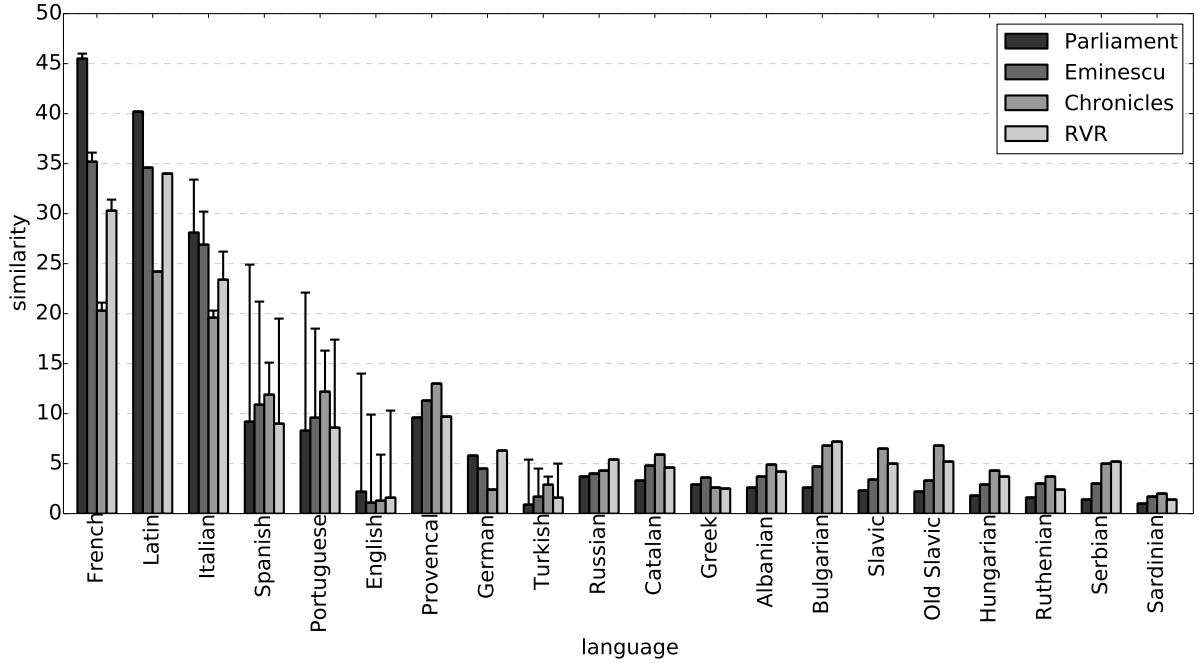


Figure 4: Degrees of similarity for the Romanian datasets. For French, Italian, Spanish, Portuguese, Turkish and English, the values obtained after the cognate identification phase are also plotted.

in Banat and Transylvania, many German words entered the basic Romanian lexicon). On the contrary, the similarity between Romanian and almost all the Slavic languages decreased in the same period. Russian is the only Slavic language which preserved its degree of similarity with Romanian (being, in the 2000s, the only Slavic language among the top 10 most closely related languages with Romanian, on the ninth position, with a degree of similarity of less than 4%). In the 18th and 19th centuries, the transition to the Latin alphabet and the desire to restore Romanian's Latin appearance contributed to the decrease of the Slavic influence (Gheţie, 1978). In fact, all Slavic influences in the 2000s sum up to 8.9%, in contrast with Latin influences, reaching 61.8%. Greek is the only language which reaches its peak regarding the similarity with Romanian in the 19th century (due to the brief Phanariot dominance in the 19th century). Therefore, Romanian preserved its Latin character all along, and the influence of the non-Latin languages on Romanian (overestimated in some works) was in fact not so significant. This fact supports Darwin's theory (Darwin, 1859), which states that the genealogy of languages is consistent with the genealogy of the nations (analyzed based on DNA similarity).

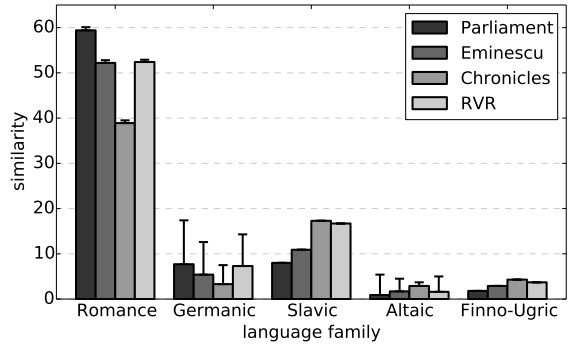


Figure 5: Degrees of similarity for the language families. Iranian and Baltic have a degree of similarity of less than 0.5.

Orthographic Metrics. In order to compare the similarity scores computed with the three metrics used, we conduct hypothesis tests (Sheskin, 2003) to determine whether the differences between the results obtained with each metric are statistically significant. We extract a sample of 5,000 words and we compute the pairwise differences between the similarity scores. Using the R v3.1.0 software environment for statistical computing (R Core Team, 2014), we perform the one-way ANOVA F-test, with the null hypothesis $\mathcal{H}_0$: $\mu_{EDIT} = \mu_{LCSR} = \mu_{RD}$ (where μ_Δ is the mean of the val-

ues computed with the Δ metric) and the alternative hypothesis $\mathcal{H}_a$: *not all* $\mu_{EDIT}, \mu_{LCSR}, \mu_{RD}$ *are equal*. Since the p-value of 2.88e-05 is much smaller than the 0.05 significance level, we have very strong evidence to reject the null hypothesis that the mean computed values for the three metrics are all equal. Further, we perform post-hoc comparisons applying pairwise t-tests with Bonferonni correction for the p-value, in order to analyze the differences between the metrics. For each pair of metrics, $p \ll 0.05$. The differences are statistically significant, but we notice that they are small. Applying a two-sampled t-test, we obtain a [0.012, 0.015] confidence interval for the mean difference between EDIT and LCSR, [0.015, 0.018] for EDIT and RD, and [0.001, 0.003] for RD and LCSR, at 95% confidence level. Moreover, computing Spearman's rank correlation coefficient for the rankings obtained by each metric for each dataset, we observe a very high correlation between them ($\rho > 0.98$ for each pair of variables). Thus, we conclude that reporting the average of the three metrics is relevant for our experiments, as differences are small and do not influence the ranking.

3.2 Europarl Experiments

We continue our investigation regarding the similarity of natural languages with two additional experiments. First, we want to see if degrees of similarity between Romanian and other languages in the present period are consistent across two different corpora. In the second experiment we are interested to see if there are differences between the overall degrees of similarity obtained for the entire corpus (the bag-of-words model) and those obtained in various experiments at sentence level. Our main corpus is Europarl (Koehn, 2005). More specifically, we use the portions larger than 2KB collected between 2007 and 2011 from the Romanian subcorpus of Europarl. The corpus is tokenized and sentence-aligned in 21 languages. For preprocessing this corpus, we discard all the transcribers' descriptions of the parliamentary sessions (such as "The President interrupted the speaker" or "The session was suspended at 19:30 and resumed at 21:00").

Exp. #1. In a first step, we apply the methodology described in Section 2 on the entire Europarl corpus for Romanian, using a bag-of-words model for the entire corpus, in which we account for the overall frequencies of the words. In this experiment, as in the previous ones, we cannot detect outliers, i.e., sentences which are unbalanced regarding the etymologies of the comprised words. For this reason, we conduct a second experiment which addresses this potential issue.

Exp. #2. We determine sentence-level orthographic similarity and we aggregate the results to compute the average values for the related languages. In this second experiment, we apply the methodology described in Section 2 for each sentence in the Europarl corpus for Romanian. For each sentence we obtain a ranking of related languages and, in order to obtain a ranking of similarity for the entire corpus, we compute the average degrees of similarity: for each related language, we sum up the degrees of similarity for all the sentences and divide this value by the number of sentences in the corpus.

Exp. #3. Because the interpretation of statistics derived from datasets that include outliers may be misleading, we compute the standard quartiles *Q1, Q2* and *Q3* (Sheskin, 2003) with regard to the length of the sentences. We use the interquartile range $IQR = Q3 - Q1$ to find outliers in the data. We consider outliers the observations that fall below the lower fence $LF = Q1 - 1.5(IQR)$ or above the upper fence $UF = Q3 + 1.5(IQR)$. We apply our methodology again only for the sentences having the length in the $[LF, UF]$ range.

Language	Exp. #1	Exp. #2	Exp. #3	Exp. #4
French	53.1	52.1	52.1	52.8
Latin	44.1	43.6	43.6	44.0
Italian	40.6	39.9	39.9	40.2
Portuguese	33.6	32.9	32.8	33.2
Spanish	27.6	27.3	27.3	26.8
English	16.0	15.7	15.7	15.1
Provencal	10.0	10.1	10.1	9.3
Turkish	6.3	6.2	6.1	5.7
German	5.9	5.8	5.8	5.3
Greek	4.4	4.3	4.3	3.8
Russian	4.2	4.1	4.1	3.6
Catalan	4.1	4.2	4.2	3.5
Old Slavic	3.1	3.2	3.2	2.7
Albanian	3.0	3.1	3.1	2.5
Bulgarian	2.9	2.9	2.9	2.2
Slavic	2.6	2.5	2.5	1.9
Hungarian	2.5	2.4	2.4	1.7
Ruthenian	2.1	2.1	2.1	1.3
Serbian	1.6	1.5	1.5	0.7
Sardinian	1.2	1.2	1.3	0.1

Table 4: Results for Europarl on the entire corpus (Exp. #1) and at sentence level (Exp. #2 - #4).

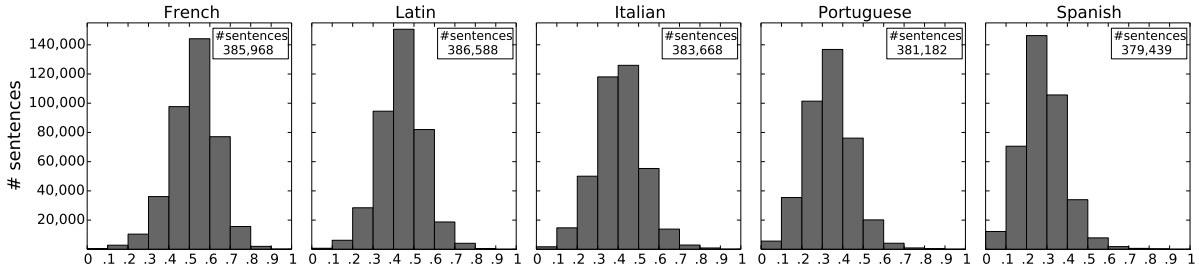


Figure 6: Distribution of the Romanian sentences in Europarl based on their similarity with the top 5 ranked languages. The OX axis represents the degree of similarity normalized to [0,1].

Exp. #4. As a last experiment, for each language L we consider as observations the degrees of similarity between Romanian and L, we discard outliers and we compute the average value of the observations inside the $[LF, UF]$ range. For each language, we determine the distribution of the sentences according to their similarity with the given language (the histograms for the top 5 languages are presented in Figure 6).

In Figure 7 we report the top 20 languages in the ranking of similarity for Europarl, emphasizing the gain obtained by identifying cognates. In Table 4 we report the similarity scores for the top 20 languages in the rankings of similarity for all the 4 experiments: overall similarity for the entire Europarl corpus (Exp. #1), sentence-level similarity (Exp. #2), similarity for the sentences having the length in the $[LF, UF]$ range (Exp. #3), and similarity for the sentences having the similarity between Romanian and each related language in the $[LF, UF]$ range (Exp. #4).

Some remarks are immediate. We observe that the differences between the values obtained for the entire corpus (Exp. #1) and those obtained in various experiments at sentence level (Exp. #2 - #4) are very small (an exception is Exp. #4, for languages whose degrees of similarity with Romanian are of less than 10%). We test the bag-of-words model on two corpora from the same period (the Parliament corpus and Europarl – in Exp. #1) and we notice that the results are consistent across different corpora (0.98 Spearman's ρ). The only significant difference is for Portuguese, which is closer to Romanian as measured on Europarl than on the Parliament corpus.

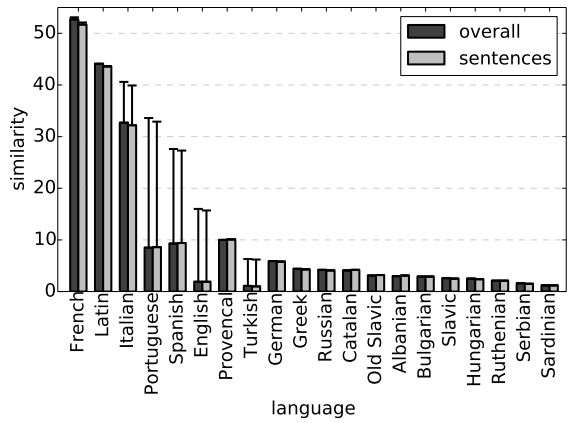


Figure 7: Degrees of similarity for Europarl.

4 Conclusions and Future Work

In this paper we proposed a computational method for determining cross-language orthographic similarity, with application on Romanian. We investigated etymons and cognates and we conducted a fine-grained analysis of the orthographic similarity between Romanian and related languages. Our results provide a new insight into the classification and evolution of Romanian. We plan to apply our similarity method on a corpus of spoken language, and to extend our analysis to other languages as well, as we gain access to available resources. We further intend to combine our orthographic approach with syntactic and semantic evidence for a wider perspective on language similarity.

Acknowledgements

We thank the anonymous reviewers for their helpful and constructive comments. We thank Raluca Vasilache for the help with evaluating the automatic method for detecting related words. The contribution of the authors to this paper is equal. Research supported by CNCS UEFISCDI, project number PNII-ID-PCE-2011-3-0959.

References

Alexander V. Alekseyenko, Quentin D. Atkinson, Remco Bouckaert, Alexei J. Drummond, Michael Dunn, Russell D. Gray, Simon J. Greenhill, Philippe Lemey, and Marc A. Suchard. 2012. Mapping the Origins and Expansion of the Indo-European Language Family. *Science*, 337(6097):957–960.

Quentin D. Atkinson and Russell D. Gray. 2006. How Old is the Indo-European Language Family? Illumination or More Moths to the Flame? In Peter Forster and Colin Renfrew, editors, *Phylogenetic Methods and the Prehistory of Languages*, chapter 8, pages 91–109. McDonald Institute for Archaeological Research.

Quentin D. Atkinson, Russell D. Gray, Geoff K. Nicholls, and David J. Welch. 2005. From Words to Dates: Water into Wine, Mathemagic or Phylogenetic Inference? *Transactions of the Philological Society*, 103(2):193–219.

Francois Barbançon, Steven N. Evans, Luay Nakhleh, Don Ringe, and Tandy Warnow. 2013. An Experimental Study Comparing Linguistic Phylogenetic Reconstruction Methods. *Diachronica*, 30(2):143 – 170.

Alessandro G. Benati and Bill VanPatten. 2011. Key Terms in Second Language Acquisition. *International Journal of Applied Linguistics*, 21(2):270–273.

Lyle Campbell. 2003. How to Show Languages are Related: Methods for Distant Genetic Relationship. In Brian D. Joseph and Richard W. Janda, editors, *The Handbook of Historical Linguistics*. Blackwell.

Alina Maria Ciobanu and Liviu P. Dinu. 2014a. Building a Dataset of Multilingual Cognates for the Romanian Lexicon. In *Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014*, pages 1038–1043.

Alina Maria Ciobanu and Liviu P. Dinu. 2014b. On the Romance Languages Mutual Intelligibility. In *Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014*, pages 3313–3318.

Charles Robert Darwin. 1859. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. John Murray.

Jean-Charles de Borda. 1781. *Mémoire sur les Élections au Scrutin*. Histoire de l'Académie Royale des Sciences.

Antonella Delmestri and Nello Cristianini. 2010. String Similarity Measures and PAM-like Matrices for Cognate Identification. *Bucharest Working Papers in Linguistics*, 12(2):71–82.

Anca Dinu and Liviu P. Dinu. 2005. On the Syllabic Similarities of Romance Languages. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing, CICLing 2005*, pages 785–788.

Liviu P. Dinu and Andrea Sgarro. 2006. A Low-Complexity Distance for DNA Strings. *Fundam. Inform.*, 73(3):361–372.

Mihai Dinu. 1996. *Personalitatea Limbii Române*. Cartea Românească.

Mark Durie and Malcolm Ross. 1996. *The Comparative Method Reviewed: Regularity and Irregularity in Language Change*. Oxford University Press.

Isidore Dyen, Joseph B. Kruskal, and Paul Black. 1992. An Indoeuropean Classification: a Lexicostatistical Experiment. *Transactions of the Americal Philosophical Society*, 82(5):1–132.

Mihai Eminescu. 1980-1985. *Opere. Vol IX-XIII. Publicistică*. Editura Academiei Române.

Oana Frunza, Diana Inkpen, and Grzegorz Kondrak. 2005. Automatic Identification of Cognates and False Friends in French and English. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing, RANLP 2005*, pages 251–257.

Ion Gheţie. 1978. *Istoria Limbii Române Literare. Privire Sintetică*. Editura Ştiinţifică şi Enciclopedică.

Charlotte Gooskens, Wilbert Heeringa, and Karin Beijering. 2008. Phonetic and Lexical Predictors of Intelligibility. *International Journal of Humanities and Arts Computing*, 2(1-2):63–81.

Charlotte Gooskens. 2007. The Contribution of Linguistic Factors to the Intelligibility of Closely Related Languages. *Journal of Multilingual and Multicultural Development*, 28(6):445.

Alexandru Graur. 1968. *Tendinţele Actuale ale Limbii Române*. Editura Ştiinţifică.

Cristian Grozea. 2012. Experiments and Results with Diacritics Restoration in Romanian. In *Proceedings of the 15th International Conference on Text, Speech and Dialogue, TSD 2012*, pages 199–206.

Jan Hajič, Jan Hric, and Vladislav Kubon. 2000. Machine Translation of Very Close Languages. In *Proceedings of the 6th Conference on Applied Natural Language Processing, ANLC 2000*, pages 7–12.

Brian D. Joseph. 1999. Romanian and the Balkans: Some Comparative Perspectives. In Sheila Embleton, John E. Joseph, and Hans-Joseph Niederehe, editors, *The Emergence of the Modern Language Sciences*. John Benjamins Publishing Company.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the 10th Machine Translation Summit, AAMT 2005*, pages 79–86.

Grzegorz Kondrak. 2001. Identifying Cognates by Phonetic and Semantic Similarity. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies, NAACL 2001*, pages 1–8.

Moshe Koppel and Noam Ordan. 2011. Translationese and Its Dialects. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, HLT 2011*, pages 1318–1326.

Vladimir I. Levenshtein. 1965. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics Doklady*, 10:707–710.

April McMahon and Robert McMahon. 2003. Finding Families: Quantitative Methods in Language Classification. *Transactions of the Philological Society*, 101(1):7–55.

Dan Melamed. 1995. Automatic Evaluation and Uniform Filter Cascades for Inducing N-Best Translation Lexicons. In *Proceedings of the 3rd Workshop on Very Large Corpora*, pages 184–198.

R Core Team, 2014. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Taraka Rama and Lars Borin. 2014. Comparative Evaluation of String Similarity Measures for Automatic Language Classification. In George K. Mikros and Ján Macutek, editors, *Sequences in Language and Text*. De Gruyter Mouton.

Don Ringe, Ann Taylor, and Tandy Warnow. 2002. Indo-European and Computational Cladistics. *Transactions of the Philological Society*, 100(1):59–129.

Marius Sala. 1988. *Vocabularul Reprezentativ al Limbilor Romanice*. Editura Academiei.

Kevin Scannell. 2006. Machine Translation for Closely Related Language Pairs. In *Proceedings of the Workshop on Strategies for Developing Machine Translation for Minority Languages*, pages 103–107.

David J Sheskin. 2003. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman and Hall/CRC Press.

Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space

Arvind Neelakantan*, **Jeevan Shankar***, **Alexandre Passos, Andrew McCallum**
Department of Computer Science
University of Massachusetts, Amherst
Amherst, MA, 01003
{arvind,jshankar,apassos,mccallum}@cs.umass.edu

Abstract

There is rising interest in vector-space word embeddings and their use in NLP, especially given recent methods for their fast estimation at very large scale. Nearly all this work, however, assumes a single vector per word type—ignoring polysemy and thus jeopardizing their usefulness for downstream tasks. We present an extension to the Skip-gram model that efficiently learns multiple embeddings per word type. It differs from recent related work by jointly performing word sense discrimination and embedding learning, by non-parametrically estimating the number of senses per word type, and by its efficiency and scalability. We present new state-of-the-art results in the word similarity in context task and demonstrate its scalability by training with one machine on a corpus of nearly 1 billion tokens in less than 6 hours.

1 Introduction

Representing words by dense, real-valued vector embeddings, also commonly called "distributed representations," helps address the curse of dimensionality and improve generalization because they can place near each other words having similar semantic and syntactic roles. This has been shown dramatically in state-of-the-art results on language modeling (Bengio et al, 2003; Mnih and Hinton, 2007) as well as improvements in other natural language processing tasks (Collobert and Weston, 2008; Turian et al, 2010). Substantial benefit arises when embeddings can be trained on large volumes of data. Hence the recent considerable interest in the CBOW and Skip-gram models of Mikolov et al (2013a); Mikolov et al (2013b)—relatively simple log-linear models that can be trained to produce high-quality word embeddings on the entirety of English Wikipedia text in less than half a day on one machine.

There is rising enthusiasm for applying these models to improve accuracy in natural language processing, much like Brown clusters (Brown et al, 1992) have become common input features for many tasks, such as named entity extraction (Miller et al, 2004; Ratinov and Roth, 2009) and parsing (Koo et al, 2008; Täckström et al, 2012). In comparison to Brown clusters, the vector embeddings have the advantages of substantially better scalability in their training, and intriguing potential for their continuous and multi-dimensional interrelations. In fact, Passos et al (2014) present new state-of-the-art results in CoNLL 2003 named entity extraction by directly inputting continuous vector embeddings obtained by a version of Skip-gram that injects supervision with lexicons. Similarly Bansal et al (2014) show results in dependency parsing using Skip-gram embeddings. They have also recently been applied to machine translation (Zou et al, 2013; Mikolov et al, 2013c).

A notable deficiency in this prior work is that each word type (*e.g.* the word string plant) has only one vector representation—polysemy and hononymy are ignored. This results in the word plant having an embedding that is approximately the average of its different contextual semantics relating to biology, placement, manufacturing and power generation. In moderately high-dimensional spaces a vector can be relatively "close" to multiple regions at a time, but this does not negate the unfortunate influence of the triangle inequality[2] here: words that are not synonyms but are synonymous with different senses of the same word will be pulled together. For example, pollen and refinery will be inappropriately pulled to a dis-

*The first two authors contributed equally to this paper.

[2]For distance d, $d(a,c) \leq d(a,b) + d(b,c)$.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1059–1069,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

tance not more than the sum of the distances plant–pollen and plant–refinery. Fitting the constraints of legitimate continuous gradations of semantics are challenge enough without the additional encumbrance of these illegitimate triangle inequalities.

Discovering embeddings for multiple senses per word type is the focus of work by Reisinger and Mooney (2010a) and Huang et al (2012). They both pre-cluster the contexts of a word type's tokens into discriminated senses, use the clusters to re-label the corpus' tokens according to sense, and then learn embeddings for these re-labeled words. The second paper improves upon the first by employing an earlier pass of non-discriminated embedding learning to obtain vectors used to represent the contexts. Note that by pre-clustering, these methods lose the opportunity to jointly learn the sense-discriminated vectors and the clustering. Other weaknesses include their fixed number of sense per word type, and the computational expense of the two-step process—the Huang et al (2012) method took one week of computation to learn multiple embeddings for a 6,000 subset of the 30,000 vocabulary on a corpus containing close to billion tokens.[3]

This paper presents a new method for learning vector-space embeddings for multiple senses per word type, designed to provide several advantages over previous approaches. (1) Sense-discriminated vectors are learned jointly with the assignment of token contexts to senses; thus we can use the emerging sense representation to more accurately perform the clustering. (2) A non-parametric variant of our method automatically discovers a varying number of senses per word type. (3) Efficient online joint training makes it fast and scalable. We refer to our method as *Multiple-sense Skip-gram*, or *MSSG*, and its non-parametric counterpart as *NP-MSSG*.

Our method builds on the Skip-gram model (Mikolov et al, 2013a), but maintains multiple vectors per word type. During online training with a particular token, we use the average of its context words' vectors to select the token's sense that is closest, and perform a gradient update on that sense. In the non-parametric version of our method, we build on *facility location* (Meyerson, 2001): a new cluster is created with probability proportional to the distance from the context to the nearest sense.

We present experimental results demonstrating the benefits of our approach. We show qualitative improvements over single-sense Skip-gram and Huang et al (2012), comparing against word neighbors from our parametric and non-parametric methods. We present quantitative results in three tasks. On both the SCWS and WordSim353 data sets our methods surpass the previous state-of-the-art. The Google Analogy task is not especially well-suited for word-sense evaluation since its lack of context makes selecting the sense difficult; however our method dramatically outperforms Huang et al (2012) on this task. Finally we also demonstrate scalabilty, learning multiple senses, training on nearly a billion tokens in less than 6 hours—a 27x improvement on Huang et al.

2 Related Work

Much prior work has focused on learning vector representations of words; here we will describe only those most relevant to understanding this paper. Our work is based on neural language models, proposed by Bengio et al (2003), which extend the traditional idea of n-gram language models by replacing the conditional probability table with a neural network, representing each word token by a small vector instead of an indicator variable, and estimating the parameters of the neural network and these vectors jointly. Since the Bengio et al (2003) model is quite expensive to train, much research has focused on optimizing it. Collobert and Weston (2008) replaces the max-likelihood character of the model with a max-margin approach, where the network is encouraged to score the correct n-grams higher than randomly chosen incorrect n-grams. Mnih and Hinton (2007) replaces the global normalization of the Bengio model with a tree-structured probability distribution, and also considers multiple positions for each word in the tree.

More relevantly, Mikolov et al (2013a) and Mikolov et al (2013b) propose extremely computationally efficient log-linear neural language models by removing the hidden layers of the neural networks and training from larger context windows with very aggressive subsampling. The goal of the models in Mikolov et al (2013a) and Mikolov et al (2013b) is not so much obtaining a low-perplexity language model as learning word representations which will be useful in

[3] Personal communication with authors Eric H. Huang and Richard Socher.

downstream tasks. Neural networks or log-linear models also do not appear to be necessary to learn high-quality word embeddings, as Dhillon and Ungar (2011) estimate word vector representations using Canonical Correlation Analysis (CCA).

Word vector representations or embeddings have been used in various NLP tasks such as named entity recognition (Neelakantan and Collins, 2014; Passos et al, 2014; Turian et al, 2010), dependency parsing (Bansal et al, 2014), chunking (Turian et al, 2010; Dhillon and Ungar, 2011), sentiment analysis (Maas et al, 2011), paraphrase detection (Socher et al, 2011) and learning representations of paragraphs and documents (Le and Mikolov, 2014). The word clusters obtained from Brown clustering (Brown et al, 1992) have similarly been used as features in named entity recognition (Miller et al, 2004; Ratinov and Roth, 2009) and dependency parsing (Koo et al, 2008), among other tasks.

There is considerably less prior work on learning multiple vector representations for the same word type. Reisinger and Mooney (2010a) introduce a method for constructing multiple sparse, high-dimensional vector representations of words. Huang et al (2012) extends this approach incorporating global document context to learn multiple dense, low-dimensional embeddings by using recursive neural networks. Both the methods perform word sense discrimination as a preprocessing step by clustering contexts for each word type, making training more expensive. While methods such as those described in Dhillon and Ungar (2011) and Reddy et al (2011) use token-specific representations of words as part of the learning algorithm, the final outputs are still one-to-one mappings between word types and word embeddings.

3 Background: Skip-gram model

The Skip-gram model learns word embeddings such that they are useful in predicting the surrounding words in a sentence. In the Skip-gram model, $v(w) \in R^d$ is the vector representation of the word $w \in W$, where W is the words vocabulary and d is the embedding dimensionality.

Given a pair of words (w_t, c), the probability that the word c is observed in the context of word w_t is given by,

$$P(D = 1|v(w_t), v(c)) = \frac{1}{1 + e^{-v(w_t)^T v(c)}} \quad (1)$$

The probability of not observing word c in the context of w_t is given by,

$$P(D = 0|v(w_t), v(c)) = 1 - P(D = 1|v(w_t), v(c))$$

Given a training set containing the sequence of word types $w_1, w_2, \ldots, w_T$, the word embeddings are learned by maximizing the following objective function:

$$J(\theta) = \sum_{(w_t, c_t) \in D^+} \sum_{c \in c_t} \log P(D = 1|v(w_t), v(c)) + \sum_{(w_t, c'_t) \in D^-} \sum_{c' \in c'_t} \log P(D = 0|v(w_t), v(c'))$$

where w_t is the t^{th} word in the training set, c_t is the set of observed context words of word w_t and c'_t is the set of randomly sampled, noisy context words for the word w_t. D^+ consists of the set of all observed word-context pairs (w_t, c_t) $(t = 1, 2 \ldots, T)$. D^- consists of pairs (w_t, c'_t) $(t = 1, 2 \ldots, T)$ where c'_t is the set of randomly sampled, noisy context words for the word w_t.

For each training word w_t, the set of context words $c_t = \{w_{t-R_t}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+R_t}\}$ includes R_t words to the left and right of the given word as shown in Figure 1. R_t is the window size considered for the word w_t uniformly randomly sampled from the set $\{1, 2, \ldots, N\}$, where N is the maximum context window size.

The set of noisy context words c'_t for the word w_t is constructed by randomly sampling S noisy context words for each word in the context c_t. The noisy context words are randomly sampled from the following distribution,

$$P(w) = \frac{p_{unigram}(w)^{3/4}}{Z} \quad (2)$$

where $p_{unigram}(w)$ is the unigram distribution of the words and Z is the normalization constant.

4 Multi-Sense Skip-gram (MSSG) model

To extend the Skip-gram model to learn multiple embeddings per word we follow previous work (Huang et al, 2012; Reisinger and Mooney, 2010a)

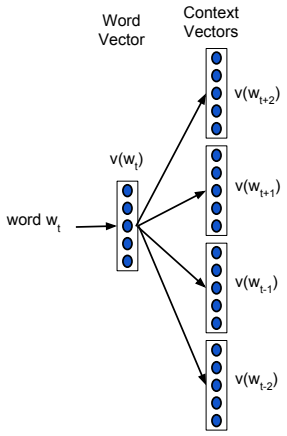


Figure 1: Architecture of the Skip-gram model with window size $R_t = 2$. Context c_t of word w_t consists of $w_{t-1}, w_{t-2}, w_{t+1}, w_{t+2}$.

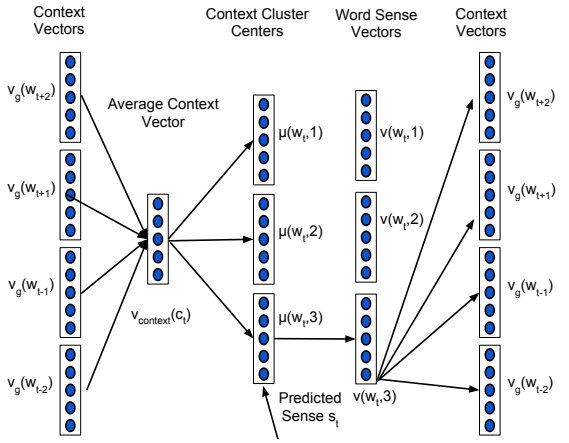


Figure 2: Architecture of Multi-Sense Skip-gram (MSSG) model with window size $R_t = 2$ and $K = 3$. Context c_t of word w_t consists of $w_{t-1}, w_{t-2}, w_{t+1}, w_{t+2}$. The sense is predicted by finding the cluster center of the context that is closest to the average of the context vectors.

and let each sense of word have its own embedding, and induce the senses by clustering the embeddings of the context words around each token. The vector representation of the context is the average of its context words' vectors. For every word type, we maintain clusters of its contexts and the sense of a word token is predicted as the cluster that is closest to its context representation. After predicting the sense of a word token, we perform a gradient update on the embedding of that sense. The crucial difference from previous approaches is that word sense discrimination and learning embeddings are performed jointly by predicting the sense of the word using the current parameter estimates.

In the MSSG model, each word $w \in W$ is associated with a global vector $v_g(w)$ and each sense of the word has an embedding (sense vector) $v_s(w,k)$ $(k = 1, 2, \ldots, K)$ and a context cluster with center $\mu(w,k)$ $(k = 1, 2, \ldots, K)$. The K sense vectors and the global vectors are of dimension d and K is a hyperparameter.

Consider the word w_t and let $c_t = \{w_{t-R_t}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+R_t}\}$ be the set of observed context words. The vector representation of the context is defined as the average of the global vector representation of the words in the context. Let $v_{context}(c_t) = \frac{1}{2*R_t} \sum_{c \in c_t} v_g(c)$ be the vector representation of the context c_t. We use the global vectors of the context words instead of its sense vectors to avoid the computational complexity associated with predicting the sense of the context words. We predict s_t, the sense of word w_t when observed with context c_t as the context cluster membership of the vector $v_{context}(c_t)$ as shown in Figure 2. More formally,

$$s_t = \underset{k=1,2,\ldots,K}{\arg\max}\ sim(\mu(w_t,k), v_{context}(c_t)) \quad (3)$$

The hard cluster assignment is similar to the k-means algorithm. The cluster center is the average of the vector representations of all the contexts which belong to that cluster. For sim we use cosine similarity in our experiments.

Here, the probability that the word c is observed in the context of word w_t given the sense of the word w_t is,

$$\begin{aligned} P(D=1|s_t, &v_s(w_t,1), \ldots, v_s(w_t,K), v_g(c)) \\ &= P(D=1|v_s(w_t,s_t), v_g(c)) \\ &= \frac{1}{1+e^{-v_s(w_t,s_t)^T v_g(c)}} \end{aligned}$$

The probability of not observing word c in the context of w_t given the sense of the word w_t is,

$$\begin{aligned} P(D=0|s_t, &v_s(w_t,1), \ldots, v_s(w_t,K), v_g(c)) \\ &= P(D=0|v_s(w_t,s_t), v_g(c)) \\ &= 1 - P(D=1|v_s(w_t,s_t), v_g(c)) \end{aligned}$$

Given a training set containing the sequence of word types $w_1, w_2, ..., w_T$, the word embeddings are learned by maximizing the following objective

Algorithm 1 Training Algorithm of MSSG model

Input: $w_1, w_2, ..., w_T$, d, K, N.
Initialize $v_s(w,k)$ and $v_g(w)$, $\forall w \in W, k \in \{1,\ldots,K\}$ randomly, $\mu(w,k)$ $\forall w \in W, k \in \{1,\ldots,K\}$ to 0.
for $t = 1, 2, \ldots, T$ **do**
    $R_t \sim \{1,\ldots,N\}$
    $c_t = \{w_{t-R_t},\ldots,w_{t-1},w_{t+1},\ldots,w_{t+R_t}\}$
    $v_{context}(c_t) = \frac{1}{2*R_t}\sum_{c\in c_t} v_g(c)$
    $s_t = \arg\max_{k=1,2,...,K}$ $\{$ $sim(\mu(w_t,k), v_{context}(c_t))\}$
    Update context cluster center $\mu(w_t, s_t)$ since context c_t is added to context cluster s_t of word w_t.
    $c'_t = Noisy\_Samples(c_t)$
    Gradient update on $v_s(w_t,s_t)$, global vectors of words in c_t and c'_t.
end for
Output: $v_s(w,k)$, $v_g(w)$ and context cluster centers $\mu(w,k)$, $\forall w \in W, k \in \{1,\ldots,K\}$

function:

$$J(\theta) = \sum_{(w_t,c_t)\in D^+}\sum_{c\in c_t}\log P(D=1|v_s(w_t,s_t),v_g(c)) + \sum_{(w_t,c'_t)\in D^-}\sum_{c'\in c'_t}\log P(D=0|v_s(w_t,s_t),v_g(c'))$$

where w_t is the t^{th} word in the sequence, c_t is the set of observed context words and c'_t is the set of noisy context words for the word w_t. D^+ and D^- are constructed in the same way as in the Skip-gram model.

After predicting the sense of word w_t, we update the embedding of the predicted sense for the word w_t ($v_s(w_t,s_t)$), the global vector of the words in the context and the global vector of the randomly sampled, noisy context words. The context cluster center of cluster s_t for the word w_t ($\mu(w_t,s_t)$) is updated since context c_t is added to the cluster s_t.

5 Non-Parametric MSSG model (NP-MSSG)

The MSSG model learns a fixed number of senses per word type. In this section, we describe a non-parametric version of MSSG, the NP-MSSG model, which learns varying number of senses per word type. Our approach is closely related to the online non-parametric clustering procedure described in Meyerson (2001). We create a new cluster (sense) for a word type with probability proportional to the distance of its context to the nearest cluster (sense).

Each word $w \in W$ is associated with sense vectors, context clusters and a global vector $v_g(w)$ as in the MSSG model. The number of senses for a word is unknown and is learned during training. Initially, the words do not have sense vectors and context clusters. We create the first sense vector and context cluster for each word on its first occurrence in the training data. After creating the first context cluster for a word, a new context cluster and a sense vector are created online during training when the word is observed with a context were the similarity between the vector representation of the context with every existing cluster center of the word is less than λ, where λ is a hyperparameter of the model.

Consider the word w_t and let $c_t = \{w_{t-R_t},\ldots,w_{t-1},w_{t+1},\ldots,w_{t+R_t}\}$ be the set of observed context words. The vector representation of the context is defined as the average of the global vector representation of the words in the context. Let $v_{context}(c_t) = \frac{1}{2*R_t}\sum_{c\in c_t} v_g(c)$ be the vector representation of the context c_t. Let $k(w_t)$ be the number of context clusters or the number of senses currently associated with word w_t. s_t, the sense of word w_t when $k(w_t) > 0$ is given by

$$s_t = \begin{cases} k(w_t)+1, & \text{if } \max_{k=1,2,...,k(w_t)}\{sim \\ & (\mu(w_t,k), v_{context}(c_t))\} < \lambda \\ k_{max}, & \text{otherwise} \end{cases} \quad (4)$$

where $\mu(w_t,k)$ is the cluster center of the k^{th} cluster of word w_t and $k_{max} = \arg\max_{k=1,2,...,k(w_t)} sim(\mu(w_t,k), v_{context}(c_t))$.

The cluster center is the average of the vector representations of all the contexts which belong to that cluster. If $s_t = k(w_t)+1$, a new context cluster and a new sense vector are created for the word w_t.

The NP-MSSG model and the MSSG model described previously differ only in the way word sense discrimination is performed. The objective function and the probabilistic model associated with observing a (word, context) pair given the sense of the word remain the same.

Model	Time (in hours)
Huang et al	168
MSSG 50d	1
MSSG-300d	6
NP-MSSG-50d	1.83
NP-MSSG-300d	5
Skip-gram-50d	0.33
Skip-gram-300d	1.5

Table 1: Training Time Results. First five model reported in the table are capable of learning multiple embeddings for each word and Skip-gram is capable of learning only single embedding for each word.

6 Experiments

To evaluate our algorithms we train embeddings using the same corpus and vocabulary as used in Huang et al (2012), which is the April 2010 snapshot of the Wikipedia corpus (Shaoul and Westbury, 2010). It contains approximately 2 million articles and 990 million tokens. In all our experiments we remove all the words with less than 20 occurrences and use a maximum context window (N) of length 5 (5 words before and after the word occurrence). We fix the number of senses (K) to be 3 for the MSSG model unless otherwise specified. Our hyperparameter values were selected by a small amount of manual exploration on a validation set. In NP-MSSG we set λ to -0.5. The Skip-gram model, MSSG and NP-MSSG models sample one noisy context word (S) for each of the observed context words. We train our models using AdaGrad stochastic gradient decent (Duchi et al, 2011) with initial learning rate set to 0.025. Similarly to Huang et al (2012), we don't use a regularization penalty.

Below we describe qualitative results, displaying the embeddings and the nearest neighbors of each word sense, and quantitative experiments in two benchmark word similarity tasks.

Table 1 shows time to train our models, compared with other models from previous work. All these times are from single-machine implementations running on similar-sized corpora. We see that our model shows significant improvement in the training time over the model in Huang et al (2012), being within well within an order-of-magnitude of the training time for Skip-gram models.

APPLE	
Skip-gram	blackberry, macintosh, acorn, pear, plum
MSSG	pear, honey, pumpkin, potato, nut microsoft, activision, sony, retail, gamestop macintosh, pc, ibm, iigs, chipsets
NP-MSSG	apricot, blackberry, cabbage, blackberries, pear microsoft, ibm, wordperfect, amiga, trs-80
FOX	
Skip-gram	abc, nbc, soapnet, espn, kttv
MSSG	beaver, wolf, moose, otter, swan nbc, espn, cbs, ctv, pbs dexter, myers, sawyer, kelly, griffith
NP-MSSG	rabbit, squirrel, wolf, badger, stoat cbs,abc, nbc, wnyw, abc-tv
NET	
Skip-gram	profit, dividends, pegged, profits, nets
MSSG	snap, sideline, ball, game-trying, scoring negative, offset, constant, hence, potential pre-tax, billion, revenue, annualized, us$
NP-MSSG	negative, total, transfer, minimizes, loop pre-tax, taxable, per, billion, us$, income ball, yard, fouled, bounced, 50-yard wnet, tvontorio, cable, tv, tv-5
ROCK	
Skip-gram	glam, indie, punk, band, pop
MSSG	rocks, basalt, boulders, sand, quartzite alternative, progressive, roll, indie, blues-rock rocks, pine, rocky, butte, deer
NP-MSSG	granite, basalt, outcropping, rocks, quartzite alternative, indie, pop/rock, rock/metal, blues-rock
RUN	
Skip-gram	running, ran, runs, afoul, amok
MSSG	running, stretch, ran, pinch-hit, runs operated , running, runs, operate, managed running, runs, operate, drivers, configure
NP-MSSG	two-run, walk-off, runs, three-runs, starts operated, runs, serviced, links, walk running, operating, ran, go, configure re-election, reelection, re-elect, unseat, term-limited helmed, longest-running, mtv, promoted, produced

Table 2: Nearest neighbors of each sense of each word, by cosine similarity, for different algorithms. Note that the different senses closely correspond to intuitions regarding the senses of the given word types.

6.1 Nearest Neighbors

Table 2 shows qualitatively the results of discovering multiple senses by presenting the nearest neighbors associated with various embeddings. The nearest neighbors of a word are computed by comparing the cosine similarity between the embedding for each sense of the word and the context embeddings of all other words in the vocabulary. Note that each of the discovered senses are indeed semantically coherent, and that a reasonable number of senses are created by the non-parametric method. Table 3 shows the nearest neighbors of the word plant for Skip-gram, MSSG , NP-MSSG and Haung's model (Huang et al, 2012).

Skip-gram	plants, flowering, weed, fungus, biomass
MS -SG	plants, tubers, soil, seed, biomass refinery, reactor, coal-fired, factory, smelter asteraceae, fabaceae, arecaceae, lamiaceae, ericaceae
NP MS -SG	plants, seeds, pollen, fungal, fungus factory, manufacturing, refinery, bottling, steel fabaceae, legume, asteraceae, apiaceae, flowering power, coal-fired, hydro-power, hydroelectric, refinery
Hua -ng et al	insect, capable, food, solanaceous, subsurface robust, belong, pitcher, comprises, eagles food, animal, catching, catch, ecology, fly seafood, equipment, oil, dairy, manufacturer facility, expansion, corporation, camp, co. treatment, skin, mechanism, sugar, drug facility, theater, platform, structure, storage natural, blast, energy, hurl, power matter, physical, certain, expression, agents vine, mute, chalcedony, quandong, excrete

Table 3: Nearest Neighbors of the word *plant* for different models. We see that the discovered senses in both our models are more semantically coherent than Huang et al (2012) and NP-MSSG is able to learn reasonable number of senses.

6.2 Word Similarity

We evaluate our embeddings on two related datasets: the WordSim-353 (Finkelstein et al, 2001) dataset and the Contextual Word Similarities (SCWS) dataset Huang et al (2012).

WordSim-353 is a standard dataset for evaluating word vector representations. It consists of a list of pairs of word types, the similarity of which is rated in an integral scale from 1 to 10. Pairs include both monosemic and polysemic words. These scores to each word pairs are given without any contextual information, which makes them tricky to interpret.

To overcome this issue, Stanford's Contextual Word Similarities (SCWS) dataset was developed by Huang et al (2012). The dataset consists of 2003 word pairs and their sentential contexts. It consists of 1328 noun-noun pairs, 399 verb-verb pairs, 140 verb-noun, 97 adjective-adjective, 30 noun-adjective, 9 verb-adjective, and 241 same-word pairs. We evaluate and compare our embeddings on both WordSim-353 and SCWS word similarity corpus.

Since it is not trivial to deal with multiple embeddings per word, we consider the following similarity measures between words w and w' given their respective contexts c and c', where $P(w, c, k)$ is the probability that w takes the k^{th} sense given the context c, and $d(v_s(w, i), v_s(w', j))$ is the similarity measure between the given embeddings $v_s(w, i)$ and $v_s(w', j)$.

The avgSim metric,

$$\mathrm{avgSim}(w, w') = \frac{1}{K^2} \sum_{i=1}^{K} \sum_{j=1}^{K} d\left(v_s(w, i), v_s(w', j)\right),$$

computes the average similarity over all embeddings for each word, ignoring information from the context.

To address this, the $\mathrm{avgSimC}$ metric,

$$\mathrm{avgSimC}(w, w') = \sum_{j=1}^{K} \sum_{i=1}^{K} P(w, c, i) P(w', c', j) \times d\left(v_s(w, i), v_s(w', j)\right)$$

weighs the similarity between each pair of senses by how well does each sense fit the context at hand.

The $\mathrm{globalSim}$ metric uses each word's global context vector, ignoring the many senses:

$$\mathrm{globalSim}(w, w') = d\left(v_g(w), v_g(w')\right).$$

Finally, $\mathrm{localSim}$ metric selects a single sense for each word based independently on its context and computes the similarity by

$$\mathrm{localSim}(w, w') = d\left(v_s(w, k), v_s(w', k')\right),$$

where $k = \arg\max_i P(w, c, i)$ and $k' = \arg\max_j P(w', c', j)$ and $P(w, c, i)$ is the probability that w takes the i^{th} sense given context c. The probability of being in a cluster is calculated as the inverse of the cosine distance to the cluster center (Huang et al, 2012).

We report the Spearman correlation between a model's similarity scores and the human judgements in the datasets.

Table 5 shows the results on WordSim-353 task. C&W refers to the language model by Collobert and Weston (2008) and HLBL model is the method described in Mnih and Hinton (2007). On WordSim-353 task, we see that our model performs significantly better than the previous neural network model for learning multi-representations per word (Huang et al, 2012). Among the methods that learn low-dimensional and dense representations, our model performs slightly better than Skip-gram. Table 4 shows the results for the SCWS task. In this task, when the words are

Model	globalSim	avgSim	avgSimC	localSim
TF-IDF	26.3	-	-	-
Collobort & Weston-50d	57.0	-	-	-
Skip-gram-50d	63.4	-	-	-
Skip-gram-300d	65.2	-	-	-
Pruned TF-IDF	62.5	60.4	60.5	-
Huang et al-50d	58.6	62.8	65.7	26.1
MSSG-50d	62.1	64.2	66.9	49.17
MSSG-300d	65.3	67.2	**69.3**	57.26
NP-MSSG-50d	62.3	64.0	66.1	50.27
NP-MSSG-300d	**65.5**	**67.3**	69.1	**59.80**

Table 4: Experimental results in the SCWS task. The numbers are Spearmans correlation $\rho \times 100$ between each model's similarity judgments and the human judgments, in context. First three models learn only a single embedding per model and hence, avgSim, avgSimC and localSim are not reported for these models, as they'd be identical to globalSim. Both our parametric and non-parametric models outperform the baseline models, and our best model achieves a score of 69.3 in this task. NP-MSSG achieves the best results when globalSim, avgSim and localSim similarity measures are used. The best results according to each metric are in bold face.

Model	$\rho \times 100$
HLBL	33.2
C&W	55.3
Skip-gram-300d	70.4
Huang et al-G	22.8
Huang et al-M	64.2
MSSG 50d-G	60.6
MSSG 50d-M	63.2
MSSG 300d-G	69.2
MSSG 300d-M	**70.9**
NP-MSSG 50d-G	61.5
NP-MSSG 50d-M	62.4
NP-MSSG 300d-G	69.1
NP-MSSG 300d-M	68.6
Pruned TF-IDF	73.4
ESA	75
Tiered TF-IDF	76.9

Table 5: Results on the WordSim-353 dataset. The table shows the Spearmans correlation ρ between the model's similarities and human judgments. G indicates the globalSim similarity measure and M indicates avgSim measure.The best results among models that learn low-dimensional and dense representations are in bold face. Pruned TF-IDF (Reisinger and Mooney, 2010a), ESA (Gabrilovich and Markovitch, 2007) and Tiered TF-IDF (Reisinger and Mooney, 2010b) construct spare, high-dimensional representations.

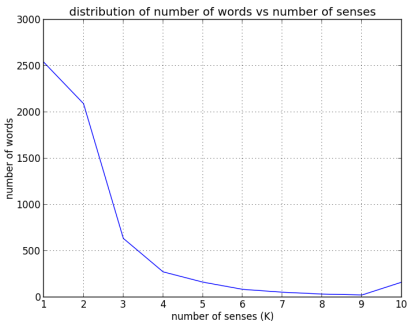
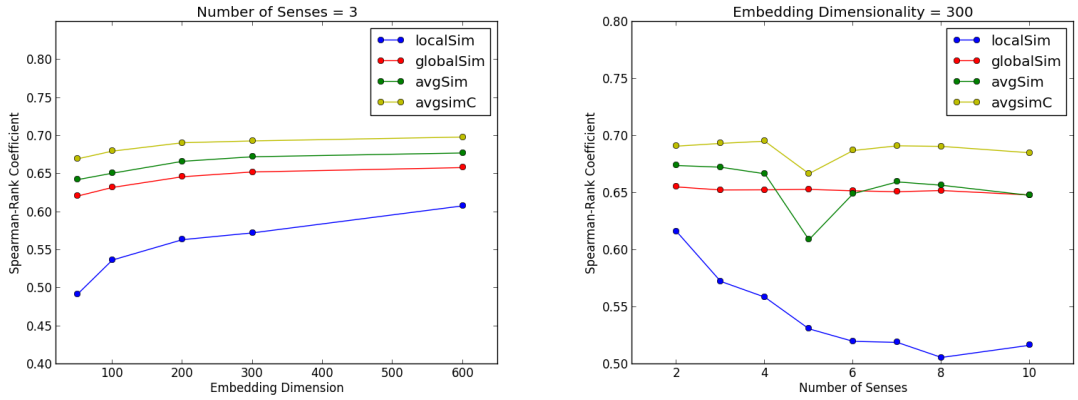


Figure 3: The plot shows the distribution of number of senses learned per word type in NP-MSSG model

given with their context, our model achieves new state-of-the-art results on SCWS as shown in the Table-4. The previous state-of-art model (Huang et al, 2012) on this task achieves 65.7% using the avgSimC measure, while the MSSG model achieves the best score of 69.3% on this task. The results on the other metrics are similar. For a fixed embedding dimension, the model by Huang et al (2012) has more parameters than our model since it uses a hidden layer. The results show that our model performs better than Huang et al (2012) even when both the models use 50 dimensional vectors and the performance of our model improves as we increase the number of dimensions to 300.

We evaluate the models in a word analogy task



(a) (b)

Figure 4: Figures (a) and (b) show the effect of varying embedding dimensionality and number of senses respectively of the MSSG Model on the SCWS task.

Model	Task	Sim	$\rho \times 100$
Skip-gram	WS-353	globalSim	70.4
MSSG	WS-353	globalSim	68.4
MSSG	WS-353	avgSim	**71.2**
NP MSSG	WS-353	globalSim	68.3
NP MSSG	WS-353	avgSim	69.66
MSSG	SCWS	localSim	59.3
MSSG	SCWS	globalSim	64.7
MSSG	SCWS	avgSim	67.2
MSSG	SCWS	avgSimC	**69.2**
NP MSSG	SCWS	localSim	60.11
NP MSSG	SCWS	globalSim	65.3
NP MSSG	SCWS	avgSim	67
NP MSSG	SCWS	avgSimC	68.6

Table 6: Experiment results on WordSim-353 and SCWS Task. Multiple Embeddings are learned for top 30,000 most frequent words in the vocabulary. The embedding dimension size is 300 for all the models for this task. The number of senses for MSSG model is 3.

introduced by Mikolov et al (2013a) where both MSSG and NP-MSSG models achieve 64% accuracy compared to 12% accuracy by Huang et al (2012). Skip-gram which is the state-of-art model for this task achieves 67% accuracy.

Figure 3 shows the distribution of number of senses learned per word type in the NP-MSSG model. We learn the multiple embeddings for the same set of approximately 6000 words that were used in Huang et al (2012) for all our experiments to ensure fair comparision. These approximately 6000 words were choosen by Huang et al. mainly from the top 30,00 frequent words in the vocabulary. This selection was likely made to avoid the noise of learning multiple senses for infrequent words. However, our method is robust to noise, which can be seen by the good performance of our model that learns multiple embeddings for the top 30,000 most frequent words. We found that even by learning multiple embeddings for the top 30,000 most frequent words in the vocubulary, MSSG model still achieves state-of-art result on SCWS task with an avgSimC score of 69.2 as shown in Table 6.

7 Conclusion

We present an extension to the Skip-gram model that efficiently learns multiple embeddings per word type. The model jointly performs word sense discrimination and embedding learning, and non-parametrically estimates the number of senses per word type. Our method achieves new state-of-the-art results in the word similarity in context task and learns multiple senses, training on close to billion tokens in less than 6 hours. The global vectors, sense vectors and cluster centers of our model and code for learning them are available at `https://people.cs.umass.edu/~arvind/emnlp2014wordvectors`. In future work we plan to use the multiple embeddings per word type in downstream NLP tasks.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by DARPA under agreement number FA8750-13-2-0020. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. *Tailoring Continuous Word Representations for Dependency Parsing.* Association for Computational Linguistics (ACL).

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. *A neural probabilistic language model.* Journal of Machine Learning Research (JMLR).

Peter F. Brown, Peter V. Desouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. *Class-based N-gram models of natural language* Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. *A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning.* International Conference on Machine learning (ICML).

Paramveer S. Dhillon, Dean Foster, and Lyle Ungar. 2011. *Multi-View Learning of Word Embeddings via CCA.* Advances in Neural Information Processing Systems (NIPS).

John Duchi, Elad Hazan, and Yoram Singer 2011. *Adaptive sub- gradient methods for online learning and stochastic optimization.* Journal of Machine Learning Research (JMLR).

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. *Placing search in context: the concept revisited.* International Conference on World Wide Web (WWW).

Evgeniy Gabrilovich and Shaul Markovitch. 2007. *Computing semantic relatedness using wikipedia-based explicit semantic analysis.* International Joint Conference on Artificial Intelligence (IJCAI).

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. *Improving Word Representations via Global Context and Multiple Word Prototypes*. Association of Computational Linguistics (ACL).

Terry Koo, Xavier Carreras, and Michael Collins. 2008. *Simple Semi-supervised Dependency Parsing.* Association for Computational Linguistics (ACL).

Quoc V. Le and Tomas Mikolov. 2014 *Distributed Representations of Sentences and Documents.* International Conference on Machine Learning (ICML)

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011 *Learning Word Vectors for Sentiment Analysis* Association for Computational Linguistics (ACL)

Adam Meyerson. 2001 *IEEE Symposium on Foundations of Computer Science.* International Conference on Machine Learning (ICML)

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. *Efficient Estimation of Word Representations in Vector Space.* Workshop at International Conference on Learning Representations (ICLR).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. *Distributed Representations of Words and Phrases and their Compositionality.* Advances in Neural Information Processing Systems (NIPS).

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013c. *Exploiting Similarities among Languages for Machine Translation.* arXiv.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. *Name tagging with word clusters and discriminative training.* North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT).

Andriy Mnih and Geoffrey Hinton. 2007. *Three new graphical models for statistical language modelling.* International Conference on Machine learning (ICML).

Arvind Neelakantan and Michael Collins. 2014. *Learning Dictionaries for Named Entity Recognition using Minimal Supervision.* European Chapter of the Association for Computational Linguistics (EACL).

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. *Lexicon Infused Phrase Embeddings for Named Entity Resolution.* Conference on Natural Language Learning (CoNLL).

Lev Ratinov and Dan Roth. 2009. *Design Challenges and Misconceptions in Named Entity Recognition.* Conference on Natural Language Learning (CoNLL).

Siva Reddy, Ioannis P. Klapaftis, and Diana McCarthy. 2011. *Dynamic and Static Prototype Vectors for Semantic Composition.* International Joint Conference on Artificial Intelligence (IJCNLP).

Joseph Reisinger and Raymond J. Mooney. 2010a. *Multi-prototype vector-space models of word meaning.* North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)

Joseph Reisinger and Raymond Mooney. 2010b. *A mixture model with sharing for lexical semantics.* Empirical Methods in Natural Language Processing (EMNLP).

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

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011 *Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection.* Advances in Neural Information Processing Systems (NIPS).

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. *Cross-lingual Word Clusters for Direct Transfer of Linguistic Structure.* North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. *Word Representations: A Simple and General Method for Semi-Supervised Learning.* Association for Computational Linguistics (ACL).

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. *Bilingual Word Embeddings for Phrase-Based Machine Translation.* Empirical Methods in Natural Language Processing.

Tailor knowledge graph for query understanding: linking intent topics by propagation

Shi Zhao
z.s@pku.edu.cn

Yan Zhang
zhy@cis.pku.edu.cn

Department of Machine Intelligence, Peking University, Beijing, China
Key Laboratory on Machine Perception, Ministry of Education, Beijing, China

Abstract

Knowledge graphs are recently used for enriching query representations in an entity-aware way for the rich facts organized around entities in it. However, few of the methods pay attention to non-entity words and clicked websites in queries, which also help conveying user intent. In this paper, we tackle the problem of intent understanding with innovatively representing entity words, refiners and clicked urls as *intent topics* in a unified knowledge graph based framework, in a way to exploit and expand knowledge graph which we call 'tailor'. We collaboratively exploit global knowledge in knowledge graphs and local contexts in query log to initialize intent representation, then propagate the enriched features in a graph consisting of intent topics using an unsupervised algorithm. The experiments prove intent topics with knowledge graph enriched features significantly enhance intent understanding.

1 Introduction

Query understanding is the process of generating a representation which characterizes a user's search intent (Croft et al., 2010), which is of vital importance for information retrieval. However, users are remarkably laconic in describing their information needs due to anomalous state of knowledge (Belkin et al., 1982), resulting in vague and underspecified queries, which makes it especially difficult to understand and locate what they intended for in mountains of web data. The problem is often significantly compounded that people convey their intent rather in a series of behaviors called a search session than a single query, leaving a wealth of clues including query reformulations, page visits, dwell times, etc. What's more, as entities are taking center stage (Yin and Shah, 2010), string-level or phrase-level modeling of intent soon hits the bottleneck, calling for an entity-aware perspective.

Knowledge repositories, better known as knowledge graphs, such as Wikipedia, DBpedia and Freebase, have been recently utilized for enhancing query understanding for the large amounts of world knowledge they've harvested about entities and facts. A widely accepted way to use knowledge graph is tying queries with it by annotating entities in them, also known as entity linking.

However, information need is conveyed through more than entities. Quite a few non-entity words, aka refiners or modifiers, as well as many urls are barely included in knowledge graph, while they play an irreplaceable role in intent understanding. For example, a user may query *toyota*, *volvo* or just enter *car soup, cars for sale* and click *www.carsoup.com*, which should be encoded in a form that we could perceive their closeness in intent. That's why at-a-glance info cards about merely recognized entity in the query are far from enough and previous methods disregarding refiners and urls are too limited to cover queries in majority.

We move one step further to tailor knowledge graph for representing more than entity words. We collect refiners and clicked urls along with entity words and model intents they represent using knowledge graph based features. We use Freebase[1], one of the largest available knowledge graph, in our work and our method can be easily generalized to other knowledge repositories.

We put up an idea of *intent topic* which can be query words or urls, whether mean an entity or not, representing an atomic information need. We identify them with *intent features* by exploiting global knowledge in Freebase and local con-

[1] http://www.freebase.com

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1070–1080,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

texts in query sessions. Notice the new concept here is distinguished from *query intent* or *query facet* in previous literature for it is in a holistic view, not specifically meaning subtopics around a certain query.

Our intuitive observations as follows inspire us to represent intent features with topics and domains in knowledge graph and propagate the enriched features in the intent topic graph.

1) Query words and urls within the same session tend to indicate the same query intent.
2) Intent topics sharing similar query intent often relate to similar topics in knowledge graph.
3) Knowledge graph domains sketch the query intent briefly.

Observation 1 indicates domain coherency within sessions is a good starting point to generate intent features, along with Observation 2 and 3 lay the basis of proximity that the propagation rely on.

To the best of our knowledge, we're the first to represent intent behind entity words, refiners and urls in a unified knowledge graph based framework, in a way to exploit and expand knowledge graph which we call 'tailor'.

Our contributions include:

- An innovative and unified framework to represent intent topics, whether they can directly link to an entity in knowledge graph or not.
- A novel algorithm to generate a specified intent topic graph, which enables learning intent features in an unsupervised propagation method.
- With intent topic graph we can better understand user intent conducting session-based contextualization and potentially find highly-related intent topic.

The rest of the paper is organized as follows. Section 2 tells our methods to map queries to Freebase and initialize intent features. Section 3 is about how we model intent topics in a unified graph and the propagation framework to learn intent features. We provide experiments and analysis in Section 4. Related work and conclusions are presented at the end of the paper.

2 Labeling intent topic nodes with Freebase-enriched features

In Freebase, facts around a certain topic and multi-faceted intents they reflect is more like a global domain distribution, what facet do users exactly intend for is difficult to locate until in a specified context, namely a query session.

We take a line in query log as a *query*, exhibiting an interaction with the search engine, including query words and page clicks. And a sequence of queries with a certain time interval constitute a *session*, completely conveying an information need.

In existing knowledge graph, only a small part of urls are contained in views of web pages beyond number online. Even for query words, we can merely get access to some of them, which we call *entity words* and the rest *refiners*. To avoid misunderstanding, the url intent topics in the following will specially refer to the clicks without directly matched concepts in knowledge graph, otherwise they'll be taken as entity intent topic.

In this section, we propose a framework of knowledge graph enriched representation of intent topics, the following propagation in Section3 bases on it.

2.1 Freebase as a knowledge graph

Freebase has over 39 million concepts, aka topics, about real-world entities like people, places and things stored as *nodes* in a graph. They're linked to each other with annotated egdes named as *property*. These *edges* actually represent *facts*. There are over a billion such facts or relations that make up the graph and they're all available for free. Properties are grouped into types, types are grouped into domains, which gives a broad view of knowledge in addtion to specific topics.

We can tap into Freebase through dump data or API[2]. In our work, we retrieve related Freebase topics with relevance scores for entity words via Freebase search API, which is based on combination of topic's inbound and outbound link counts in Freebase and Wikipedia as well as a popularity score computed by Google, and all the facts about a given topic through Freebase topic API. We use $\mathbb{T} = \{t_1, t_2, ...t_n\}$, $\mathbb{D} = \{d_1, d_2, ...d_n\}$ to denote all Freebase topics and domains used in our work.

2.2 Enriching entities and queries with Freebase

We represent a query's candidate intent topics by three sets, E_q, R_q, C_q, where E_q includes entity words and clicks which have equivalents in Freebase, R_q the refiner words and C_q the rest clicks.

[2]http://developers.google.com/freebase/

Global knowledge in Freebase can directly enrich each e in E_q with Freebase topics represented in vector $\mathbf{t}^e$, for each candidate topic there's a Freebase domain distribution vector $\mathbf{d}^t$. As for the rest in R_q and C_q, they can learn features in later propagation process.

For any topic t_i in $\mathbf{t}^e$, the relevance of entity words e and knowledge graph topic t_i is estimated as follows:

$$\mathbf{t}_i^e = \frac{RelevanceScore(e, t_i)}{\max_{t_j \in \mathbb{T}} RelevanceScore(e, t_j)} \quad (1)$$

And the domain vector $\mathbf{d}^{t_i}$ for t_i is:

$$\mathbf{d}_j^{t_i} = \frac{pr(d_j|t_i)}{\sum_{d_k \in \mathbb{D}} pr(d_k|t_i)} \quad (2)$$

$$pr(d_j|t_i) = \frac{\#\ of\ links\ of\ t_i\ in\ domain\ d_j}{\#\ of\ all\ links\ in\ domain\ d_j} \quad (3)$$

Then we'll get a knowledge graph enriched intent description of the query by combining that of e, r, c.

$$\mathbf{t}_i^q = \sum_{e \in E_q} \mathbf{t}_i^e w_q(e) + \sum_{r \in R_q} \mathbf{t}_i^r w_q(r) + \sum_{c \in C_q} \mathbf{t}_i^c w_q(c) \quad (4)$$

$$w_q(e) = NCount_q(e)\varphi(e) \quad (5)$$

Here $\mathbf{t}^e$ $\mathbf{t}^r$ $\mathbf{t}^c$ correspond to the topic vector of each entity, refiner and click respectively. The weight indicates how dominant it is in conveying intent in the query. It is in proportion to the normalized count as well as each occurrence's quality denoted by $\varphi(e)$. Such as for entity words in Equation (5), the quality $\varphi(e)$ can be estimated with the help of entity linking methods, which describes the probability of e as a candidate reference. That for clicks and refiners will be explained later.

The query's domain feature can be calculated as follows:

$$\mathbf{d}_i^q = \frac{\sum_{t_j \in \mathbb{T}} \mathbf{d}_i^{t_j} \mathbf{t}_j^q}{\sum_{d_k \in \mathbb{D}} \sum_{t_j \in \mathbb{T}} \mathbf{d}_k^{t_j} \mathbf{t}_j^q} \quad (6)$$

It describes the probability of query q in domain d_i, in which $\mathbf{t}_j^q$ can be calculated by Equation (4) and $\mathbf{d}_i^{t_j}$ via facts around topic t_j by Equation (2).

2.3 Contextualized intent depiction of sessions

The aforesaid enriched features we get about queries rely heavily on global knowledge in Freebase, reflecting prior distribution in the feature space. In this part, we derive a contextualized description of session intent in a local view by aggregating all the global knowledge we get about the session's queries. The ambiguity of a single query can be alleviated by looking at the dominant domain within the session.

The intent features $\mathbf{t}^s$ and $\mathbf{d}^s$ of session s can be represented by computations on its query set $Q_s = \{q_1, q_2, ...q_n\}$ with time-order decay.

$$\mathbf{t}_i^s = \frac{\sum_{q \in Q_s} \mathbf{t}_i^q \alpha^{rank(q)}}{\sum_{t_j \in \mathbb{T}} \sum_{q \in Q_s} \mathbf{t}_j^q \alpha^{rank(q)}} \quad (7)$$

where we put an exponential decay controlled by decay factor α. We get domain feature the same way as Equation (6).

We'll put up an unsupervised method of learning knowledge graph based intent representation of refiners and clicks in the following part.

3 Propagating intent features in the intent topic graph

In this section, our idea is to characterize entities, refiners and urls uniformly as intent topics, tailoring knowledge graph to intent topic graph so as to enrich representations by propagation.

3.1 Modeling intent topic graph

As in last section, with $\mathbf{d}^s$ featuring the context, candidate intent topics in sessions can make intent topic nodes now. We use the concept *intent topic* to stress words with local contexts tell a specified information need, thus making a node. Taking entity word *fl* as an example, it can be recognized as the topic *Florida* in Freebase, while the intent behind it can hardly be mapped to a single intent topic, such as *travel* domain in *hollywood fl*, *education* domain in *community college in florida*, and *florida department of health* actually convey intent in *government* domain.

So each intent topic node is identified with its name string and Freebase-enriched intent features $\mathbf{t}$ and $\mathbf{d}$. They're directly linked by co-occurring in the same line in the query log and implicitly related via intent features similarities, so that constitute a large graph $G =< V, E, W >$, where $\forall w \in W$ denotes an explicit edge weight and $\forall v \in V$ an intent topic. With intent topics and their relations modeled in a graph, we can better understand the query space so as to find the intended query faster. We realize it by aggregating massive sessions.

The implicit intent similarity $ISim$ of any node pair n and v can be encoded as follows.

$$ISim_{n,v} = \beta SSim_{n,v} + \gamma DSim_{n,v} + \eta TSim_{n,v} \tag{8}$$

where $SSim$ denotes the names' string similarity, $DSim$ the similarity of their domain feature and $TSim$ the topic vector similarity, with β, γ and η controlling the weight. The parameters may vary due to different scenarios. We just provide a framework of modeling nodes' intent features, which actually mirror their proximity in query intent.

To put it in more details, we use jaccard similarity for name shinglings and cosine similarity for domain and topic vector. As query log induced intent topic graph is of considerable large size, the pair-wise similarity is computationally prohibitive, hence we use Local Sensitive Hash (Indyk and Motwani, 1998) for each similarity metric so as to compute $ISim$ just in candidate set. We use random hyperplane based hash family proposed in (Charikar, 2002) and set the hash code dimension and hash table numbers empirically to ensure the number of nodes falling into each bucket is relatively stable.

3.2 Merging nodes

Although our idea of specifying intent topics by context better models the multi-facets of queries, it obviously also brings a sparse issue. For example, in one session user query *beep lyrics* and click *www.lyricsandsongs.com*, *lyrics* is tagged with the song *beep* and the musician *Pussycat Dolls*, in another scenario *lyrics* occurs with the song *what you know* and url *www.dapslyrics.com*, intents behind these two nodes are so similar that they should come into one, otherwise connections between the two intent-coherent urls may be lost.

To avoid that, we conduct a merge process to integrate nodes with exactly the same names and contexts into one, combing linked nodes and intent features together.

For a set of nearly duplicate nodes ω the calculation of new node's features can be written as:

$$\hat{\mathbf{t}} = \frac{\sum_{u \in \omega} \mathbf{t}^u}{|\omega|} \tag{9}$$

$$\hat{\mathbf{d}} = \frac{\sum_{u \in \omega} \mathbf{d}^u}{|\omega|} \tag{10}$$

In other words, we gather candidate nodes retrieved by LSH and then calculate $ISim$ for them with η setting to 0. Only node pairs with $ISim$ higher than a merge threshold θ can be seen as duplicates. The merge process is summarized in Algorithm 1.

Algorithm 1: Merging similar nodes

Input: $G =< V, E, W >, \beta, \gamma, \eta, \theta$
Output: $\hat{G} =< \hat{V}, \hat{E}, \hat{W} >$
begin
  Initialize $\Omega \leftarrow \varnothing$
  for $v \in V$ **do**
    Find dupset ω_v with $ISim_{\beta,\gamma,\eta}$
    if $\exists u \in V, \omega_u \in \Omega$ *and* $\omega_v \cap \omega_u \neq \varnothing$ **then**
      $\omega_v \leftarrow \omega_v \cup \omega_u$
      Remove ω_u from Ω
    Add ω_v to Ω
  for $\omega \in \Omega$ **do**
    Merge nodes in ω into new node $\hat{v}$
    Update G with replacing nodes in ω with $\hat{v}$

3.3 Label propagation

We utilize knowledge graph induced intent features instead of manually labels as constraints to conduct label propagation(Zhu and Ghahramani, 2002). The idea is that node labels are propagated to nearby nodes via weighted edges until convergence, as highly weighted edges indicate high probability of sharing labels.

Nodes in our work have soft labels, where each dimension of intent features denotes a label, such as a topic or domain of knowledge graph. As described in aforesaid observations, it is intuitively reasonable to propagate on the basis of explicit edges and implicit intent similarities. We illustrate the propagation with topic feature, that of domain feature is similar.

We use matrix $\mathbf{Y}^t \in \mathbb{R}^{|V|*|\mathbb{T}|}$ to denote the intent topic graph's initial topic feature labels, with element $\mathbf{Y}^t_{ik}$ indicating node v_i's relevance to t_k, wherer $t_k \in \mathbb{T}$. $\mathbf{Y}^t$ is initialized based on the results of the feature enriching step in Section 2, with no manually-labelled instances needed in our model. As only part of nodes can directly map to Freebase topics, those are initialized as labelled nodes, then propagate $\mathbf{t}$ to their linked neighbors. The number of unlabelled data is written as u, while that of labelled data l and the total number of nodes N.

The transition matrix T indicates the impact of nodes on each other. Note that here the w_{ij} can be replaced by other similarity measures such as $ISim$ in Section 3.2.

$$T_{ij} = \frac{w_{ij}}{\sum_{k=1}^{N} w_{kj}} \tag{11}$$

Let D denote an $N \times N$ diagonal matrix with $d_{ii} = \sum_j T_{ij}$. Then we can get a normalized version of transition matrix $P = D^{-1}T$.

The normalized transition matrix can be split into 4 sub-matrices.

$$P = \begin{bmatrix} P_{ll} & P_{lu} \\ P_{ul} & P_{uu} \end{bmatrix} \tag{12}$$

At each step, we propagate and clamp the labelled data and repeat until Y converges, the propagation step can be written as:

$$\hat{Y_u} = P_{uu}Y_u + P_{ul}Y_l \tag{13}$$

As is shown in (Zhu and Ghahramani, 2002; Zhu et al., 2003) the solution to the propagation converges to:

$$\hat{Y_u} = (I - P_{uu})^{-1}P_{ul}Y_l \tag{14}$$

3.4 The propagation framework for intent features

We carry the propagation in an iterative process illustrated in Algorithm 2.

Algorithm 2: Intent feature propagation

Input: G, Y_l^t,Y_l^d
Output: $\hat{G}$, $\hat{Y}_u^t$, $\hat{Y}_u^d$
Initialize Y_l^t Y^d with results of Section2
repeat
  Merge similar nodes according to Algorithm 1
  Compute matrix P
  repeat
    $\hat{Y}_u^t = P_{uu}Y_u^t + P_{ul}Y_l^t$
  until *Convergence*;
  Recompute $\hat{P}$ with $\hat{Y}^t$
  repeat
    $\hat{Y}_u^d = \hat{P}_{uu}Y_u^d + \hat{P}_{ul}Y_l^d$
  until *Convergence*;
until *no dups*;

Since intent features include both domain vector and topic vector, we propagate them in an alternating way. At first we label nodes as described in Section 2, though missing refiners' and some urls' intent features, they are just used for initialization. Then we propagate Freebase topic features based on explicit edge weights, so that more nodes in intent topic graph have topic features now. Then fetching the learned topic features, we reinput it into domain feature propagation, which means we recalculate the transition matrix combining the implicit learned $TSim$ into edge weight, then propagate domain vector of labelled nodes through the graph. At each iteration, we first update Y^t, then input it to update Y^d, therefore merge near duplicate intent topics to update the whole graph.

4 Experiments

4.1 Data preparation

4.1.1 Search logs

We use AOL search log data for experiments. It includes 20 million web queries collected covering 500K users over three months in 2006.

Table 1: The query set

# of sessions	35140
# of queries	271127
# of users	21378
# of urls	63019

We preprocess the query log by keeping urls occurring more than 3 times and queries with 2 to 40 characters, then extract sessions considering 25 minutes duration. While user session segmentation can be improved with more sophisticated algorithms, this simple low-cost heuristic performs adequately for our purposes. We then move on to map queries to Freebase and empirically filter sessions that are less entity-centric. We use an annotation tool especially for short text (Ferragina and Scaiella, 2012) called Tagme[3] to recognize entities and observe only 16% of all the queries are exactly an entity itself, which means most of queries do have refiner words to convey information need. To ensure the precision of recognized entities, we set a significant threshold and bottom line threshold , queries should have at least one recognized entity with a likelihood above significant level, and those below bottom line are ignored. They are 0.19 and 0.05 in our work, which may vary with entity recognition method. The normalized

[3]http://tagme.di.unipi.it/

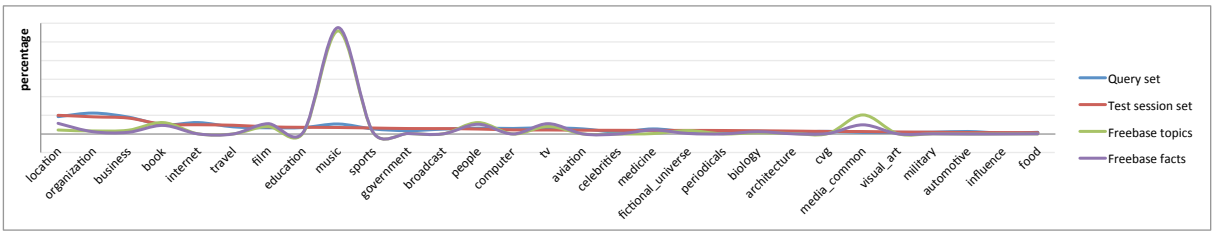


Figure 1: Unbalanced domain distributions in Freebase comparing against query set. Only domains with top proportions are shown.

Table 2: Examples of labelled intent topic nodes with learned feature

Intent topic nodes	Original in Freebase	After propagation	Annotation
travel.yahoo.com	Yahoo! Travel (internet, 0.87), (projects, 0.13)	(location, 0.13), (travel, 0.11), (organization, 0.08), (business, 0.08) ...	Yahoo! Travel offers travel guides, booking and reservation services.
map quest www.mapquest.com	MapQuest (organization, 0.6), (book, 0.4)	(location, 0.13), (organization, 0.09), (travel, 0.09), (automotive, 0.06)...	MapQuest is an American free online web mapping service.

likelihood is used as $w_q(e)$. Then we drop sessions where tagged entity words weight less than refiners as well as the ones with too many entity words spotted indicating disperse intents. For each recognized entity, only Freebase topics with relevance over 0.3 are kept. The query set we finally get is shown in Table 1.

4.1.2 Freebase

To enrich query representations, we collect a subset of Freebase including more than 7 millions facts and 4 millions topics in total which also contain 150 thousand topical equivalent websites, though less than 3% urls in query set are covered.

The facts and entities in Freebase is rather unbalanced across domains especially against that of recoginized entities in query set as shown in Figure 1. Thus the original global knowledge we use about domain distribution may cause bias, which makes *tailoring* necessary for intent understanding.

For both generality and precision, we keep most of Freebase domains except several extreme incomplete ones, instead of retaining a small number of *representative* domains like many researchers do (Li et al., 2013; Yu et al., 2014; Lin et al., 2012). But generality comes at a price that some domains are confusing and mixed used which we then choose to merge, like celebrities and people, periodicals and books, tv and broadcast, etc. We finally keep 50 of all 76 domains.

4.2 Intent topic graph

4.2.1 Building the graph

We leverage both Freebase and search sessions to enrich intent topics. We set α to 0.9 in calculation of session's intent features. After labeling the session log, we roughly make a graph with 335206 intent topic nodes, 119364 of them have been labelled with Freebase topic feature, others only have domain feature. Then we conduct a merge process with β set to 0.7, γ to 0.3 and θ to 0.75 in order to merge nodes with duplicate names and similar contexts. We find 46659 duplicate sets covering 140768 nodes. Then we ignore nodes with few links and rare names to reduce sparsity. Finally we've got a graph of 209351 intent topics to initialize the propagation, including 78932 labelled nodes. The merge and propagation progress get converged in less than 4 rounds.

We'll further evaluate the graph with case study and a session intent understanding task.

4.2.2 Case Study

We demonstrate intent features are good interpretations for query intent, whether they're labelled in Section 2 or learned by propagation in Section 3.

We can see in Table 2 that as nodes' original

Table 3: Examples of unlabelled intent topic nodes with learned feature

Intent topic node	intent features	Annotation	Similarity nodes
www.bnm.com	(The Hertz Corporation, 0.25), (Southwest Florida International Airport, 0.17), (Punta Gorda Airport, 0.13), (Supercar, 0.09), (Sports car, 0.08)... (aviation, 0.23), (business, 0.21), (location, 0.14), (automotive, 0.11)...	Online booking of discount rentals at major airports, worldwide.	www.arac.com www.rentalcars.com www.hertz.com www.alamo.com rent a car cheap rental cars
www.mobtime.com	(Software, 0.18), (Mobile phone, 0.11), (100% Totally Free Ringtones, 0.10), (Motorola, 0.09), (Free Cell, 0.08), (Verizon Wireless, 0.04)... (computer, 0.23), (cvg, 0.21), (music, 0.19), (business, 0.11) ...	MobTime Cell Phone Manager is a PC software to manage or sync mobile phones.	cellphones.about.com cell software cell to pc reviews of cellphone wallpaper

types in Freebase are not proper for describing intent, the intent features they get after propagation tend to be more explainable, such as the travel site often co-occurs with city names, tourist attractions, hotels and so on, thus indicating its intent in travel and location domain.

Table 3 shows examples which have no equivalents in Freebase. Although some of them may be accessible in other ontologies, we only take them as examples to show our propagation method makes it possible to depict intents behind urls and words in a knowledge graph based way while beyond the capacity of knowledge graph.

4.3 Session intent understanding task

4.3.1 Experiment Setup

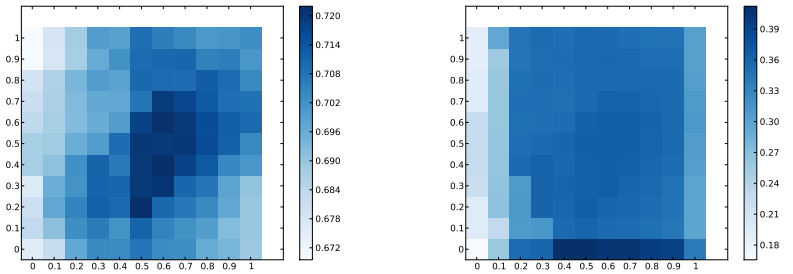
The evaluation of query understanding has long been a challenging task. To judge whether the concepts in query are successfully recognized seems too straightforward, and it can hardly be considered *understanding* the intent until the big idea about what *kind* of topics users emphasis is captured, which can be briefly sketched by distribution across Freebase domains. Also it is difficult to translate results of previous log analysis methods into knowledge graph domain information, thus hardly fit into our evaluation schema. We take popularity-based method as baseline.

We have few choices but to tag ground truth ourselves for intent understanding evaluation.

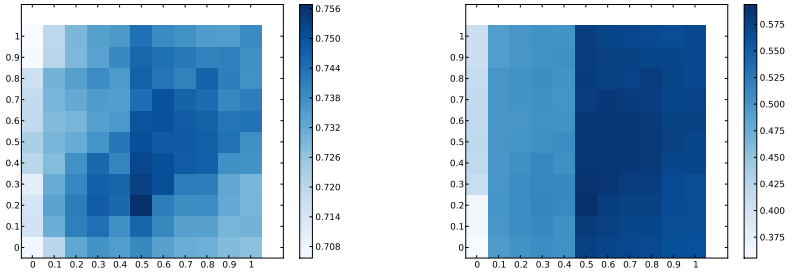
We randomly select 150 sessions as test set, the domain distribution of which agrees with the whole query set as shown in Figure 1. As mastering meanings of all Freebase domains is too challenging, we ask 5 accessors to describe each session's intent broadly with a few natural language terms, then an expert familiar with Freebase schema translates the words into matched Freebase domains. Each test session is tagged by 2 accessors and 1 expert, we choose to use the tags of the cases in which the accessors reached agreement as the gold stantard. For example, if accessors tag session intent as pictures, then experts can translate it into Freebase visual art domain. Each session has 1$\sim$4 tags and 1.6 tags in average. The tags cover 30 domains.

For each session, we derive the local intent domain vector $\mathbf{d}^s$ following the method in Section 2. Here we simply set quality function $\varphi(r)$ to a constant λ^r for all refiners and $\varphi(c)$ to λ^c for all clicks, we'll dive into more specialized weighting method in future work. λ^r and λ^c are parameters to control impact of different kinds of intent topics. Based on whether to exploit global intent features of non-entity words, we compare four variations against one baseline.

- Popularity-based (GP). We use domains' frequency in the query set as a baseline.
- Entity-based (E). We only use entity nodes' original intent features without propagation.
- Entity+Clicks (EC). Both intent features of entity words and clicks are used, controlled with λ^c.
- Entity+Refiners (ER). Intent features of entity words and refiner words are used, refiners' impact is controlled by λ^r.
- Entity+Clicks+Refiners (ECR). All intent topics are combined, controlled by λ^c, λ^r.



(a) $MAP@5$ (b) $GMAP@5$



(c) $MAP@10$ (d) $GMAP@10$

Figure 2: The impact of $\lambda^r$and λ^c on ECR methods in four metrics, with vertical axis indicating λ^r, horizontal axis as λ^c. The first column on the left denotes ER method, while the bottom row the EC method.

4.3.2 Evaluation metrics

We use each approach to rank domains according to its derived weight, then compare it with golden standard set. It can be evaluated using Mean Average Precision (MAP), Geometric MAP and Precision@K. We use GMAP because it is more robust to outliers than the arithmetic mean.

For test set of size N, the MAP and GMAP can be calculated as follows:

$$MAP@k = \frac{1}{N}\sum_{i=1}^{N} AP_i@k \tag{15}$$

$$GMAP@k = \sqrt[N]{\prod_{i=1}^{N} AP_i@k} \tag{16}$$

4.3.3 Results and analysis

We first study impact of parameters λ^r and λ^c, which is shown in Figure 2.

It roughly demonstrates different combinations of parameters' impact on ECR methods, performance is evaluated in four metrics, with deeper color indicating better result.

Best results comes with a λ^c larger than λ^r in all four subfigures. This trend seems more obvious in (d) where right part with larger λ^c get better results. Also, deeper colors around diagonal line in (a) (c) indicate a more balanced combination of refiners and urls are more likely to enhance intent understanding. Thus we conclude clicks has a weak advantage over refiners in improving the result, while combining both with proper parameters can get the best result.

When comparing between MAP and GMAP, we can see while GMAP stays a high value when amplifying the impact of clicks, MAP changes with the variation of λ^r for better or worse. As GMAP is a more robust metric, we can then infer that increasing weight of refiners could bring more outliers, implying refiners' intent features are more susceptible to noise.

Then we use ER with $\lambda^r = 0.5$ as ER^{opt}, EC with $\lambda^c = 0.5$ as EC^{opt} and ECR with $\lambda^r = 0.2, \lambda^c = 0.5$ as ECR^{opt}.

Figure 3 clearly shows the superior performance of our model, especially at top positions. Table 4 shows the detailed comparisons between different methods. We can see our knowledge graph based intent representations perform well in session intent understanding. And refiners' and clicks' intent features which we learn by propagation con-

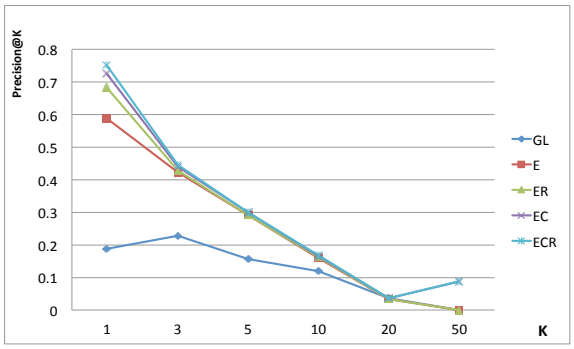


Figure 3: Precision@K results for different approaches, by varying number of k

Table 4: Comparisons among different methods

	K=5		K=10	
	MAP	GMAP	MAP	GMAP
GP	0.177	0.000	0.232	0.002
E	0.676	0.166	0.707	0.355
EC^{opt}	0.708	0.412	0.739	0.579
ER^{opt}	0.688	0.227	0.723	0.421
ECR^{opt}	0.722	0.412	0.756	0.594

tribute a lot to improve naive entity-based method, which do validate an complment effect of their learned intent features.

5 Related Work

5.1 Query intent understanding

Query intent or search intent has been studied intensively from various views.

A popular paradigm is to label several intents for each query, also called facets subgoals and subtopics in the literature, manully or by mining methods and then do classification (Hu et al., 2009; Li et al., 2008) based on that. Manually intent schemas range from 3 top level (Broder, 2002) to fine-grained subcatogories (Rose and Levinson, 2004) and taxonomy (Yin and Shah, 2010). Intent tasks in NTCIR-10 (Sakai et al., 2013) also provide subtopic pools made by accessors.

Another view of intent is more generic, mining or learning search intents without any kind of pre-defined intent category and clustering method is often used. Methods including (Sadikov et al., 2010; Yamamoto et al., 2012; Cheung and Li, 2012) cast intent as represented by a pattern or template consisting of a sequence of semantic concepts or lexical items. (Tan et al., 2012) encode intent in language models, aware of long-lasting interests. (Ren et al., 2014) uses an unsupervised heterogeneous clustering. (Yin and Shah, 2010) capture generic intents around a certain named entities and model their relationships in a tree taxonomy and (Wang et al., 2009) mine broad latent modifiers of intent aspect , which are similar to our motivation, while we model more than intent phrases, but intent topics. We do not split queries into clusters or subtopics relevant to the original query to indicate a intent, but link them in an graph with intent feature similarity, weakly or strongly, in a holistical view.

On the other hand, previous research can be categorized by what kind of resources they rely on. Quite an amount of work leverage query logs (Jiang et al., 2013), including query reformulations (Radlinski et al., 2010), click-through data (Li et al., 2008). There are also works using sponsered data (Yamamoto et al., 2012) and interactive data (Ruotsalo et al., 2013). The new trend of integrating knowledge graph will be discussed next.

5.2 Knowledge graph on intent understanding

Instead of summarizing queries into concepts by clustering, recently there appears a tendency to use concpets from knowledge graph resources. Some researchers manage to build entity graph from queries (Bordino et al., 2013a) (Bordino et al., 2013b; Yu et al., 2014), some in a structure view, interpret quries into knowledge base fit template (Pound et al., 2012; Li et al., 2013). (Pantel et al., 2012) models latent intent to mine entity type distributions. (Ren et al., 2014) utilizes knowledge graph resources in a hetrogeneous view. (Lin et al., 2012) also pays attention to refiners, but restricted to limited domains, while our method is more general.

6 Conclusion

In this paper, we tailor knowledge graph to represent query intent behind entity words, refiners and clicked urls in a unified framework, taking them as intent topic nodes connected in a large graph. We manage to get a contextualized intent depiction exploiting global knowledge in Freebase, then propagate the feature to cover more intent topics. We show in experiments the knowledge graph enriched representation is reasonable and explainable, and the intents feature of refiners and clicks can better enhance intent understanding than methods simply relying on entities.

There are several directions for future work, including using both types and domains in Freebase schema, diving into refiners and looking for a proper weighting method, developing a query recommendation framework based on the intent topic graph and user interest modeling.

Acknowledgments

We sincerely thank all the anonymous reviewers for their valuable comments, which have helped to improve this paper greatly. This work is supported by NSFC with Grant No.61370054, and 973 Program with Grant No.2014CB340405.

References

Nicholas J Belkin, Robert N Oddy, and Helen M Brooks. 1982. Ask for information retrieval: Part i. background and theory. *Journal of documentation*, 38(2):61–71.

Ilaria Bordino, Gianmarco De Francisci Morales, Ingmar Weber, and Francesco Bonchi. 2013a. From machu_picchu to rafting the urubamba river: anticipating information needs via the entity-query graph. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 275–284. ACM.

Ilaria Bordino, Yelena Mejova, and Mounia Lalmas. 2013b. Penguins in sweaters, or serendipitous entity search on user-generated content. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 109–118. ACM.

Andrei Broder. 2002. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, September.

Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM.

Jackie Chi Kit Cheung and Xiao Li. 2012. Sequence clustering and labeling for unsupervised query intent discovery. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 383–392. ACM.

W Bruce Croft, Michael Bendersky, Hang Li, and Gu Xu. 2010. Query representation and understanding workshop. In *SIGIR Forum*, volume 44, pages 48–53.

Paolo Ferragina and Ugo Scaiella. 2012. Fast and accurate annotation of short texts with wikipedia pages. *IEEE software*, 29(1).

Jian Hu, Gang Wang, Fred Lochovsky, Jian-tao Sun, and Zheng Chen. 2009. Understanding user's query intent with wikipedia. In *Proceedings of the 18th international conference on World wide web*, pages 471–480. ACM.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM.

Daxin Jiang, Jian Pei, and Hang Li. 2013. Mining search and browse logs for web search: A survey. *ACM Trans. Intell. Syst. Technol.*, 4(4):57:1–57:37, October.

Xiao Li, Ye-Yi Wang, and Alex Acero. 2008. Learning query intent from regularized click graphs. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 339–346, New York, NY, USA. ACM.

Yanen Li, Bo-June Paul Hsu, and ChengXiang Zhai. 2013. Unsupervised identification of synonymous query intent templates for attribute intents. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, CIKM '13, pages 2029–2038, New York, NY, USA. ACM.

Thomas Lin, Patrick Pantel, Michael Gamon, Anitha Kannan, and Ariel Fuxman. 2012. Active objects: Actions for entity-centric search. In *Proceedings of the 21st international conference on World Wide Web*, pages 589–598. ACM.

Patrick Pantel, Thomas Lin, and Michael Gamon. 2012. Mining entity types from query logs via user intent modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 563–571. Association for Computational Linguistics.

Jeffrey Pound, Alexander K Hudek, Ihab F Ilyas, and Grant Weddell. 2012. Interpreting keyword queries over web knowledge bases. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 305–314. ACM.

Filip Radlinski, Martin Szummer, and Nick Craswell. 2010. Inferring query intent from reformulations and clicks. In *Proceedings of the 19th international conference on World wide web*, pages 1171–1172. ACM.

Xiang Ren, Yujing Wang, Xiao Yu, Jun Yan, Zheng Chen, and Jiawei Han. 2014. Heterogeneous graph-based intent learning with queries, web pages and wikipedia concepts. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 23–32, New York, NY, USA. ACM.

Daniel E. Rose and Danny Levinson. 2004. Understanding user goals in web search. In *Proceedings of the 13th International Conference on World*

Wide Web, WWW '04, pages 13–19, New York, NY, USA. ACM.

Tuukka Ruotsalo, Jaakko Peltonen, Manuel Eugster, Dorota Głowacka, Ksenia Konyushkova, Kumaripaba Athukorala, Ilkka Kosunen, Aki Reijonen, Petri Myllymäki, Giulio Jacucci, et al. 2013. Directing exploratory search with interactive intent modeling. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1759–1764. ACM.

Eldar Sadikov, Jayant Madhavan, Lu Wang, and Alon Halevy. 2010. Clustering query refinements by user intent. In *Proceedings of the 19th international conference on World wide web*, pages 841–850. ACM.

Tetsuya Sakai, Zhicheng Dou, Takehiro Yamamoto, Yiqun Liu, Min Zhang, Ruihua Song, MP Kato, and M Iwata. 2013. Overview of the ntcir-10 intent-2 task. *Proceedings of NTCIR-10*, pages 94–123.

Bin Tan, Yuanhua Lv, and ChengXiang Zhai. 2012. Mining long-lasting exploratory user interests from search history. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1477–1481. ACM.

Xuanhui Wang, Deepayan Chakrabarti, and Kunal Punera. 2009. Mining broad latent query aspects from search sessions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 867–876. ACM.

Takehiro Yamamoto, Tetsuya Sakai, Mayu Iwata, Chen Yu, Ji-Rong Wen, and Katsumi Tanaka. 2012. The wisdom of advertisers: mining subgoals via query clustering. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 505–514. ACM.

Xiaoxin Yin and Sarthak Shah. 2010. Building taxonomy of web search intents for name entity queries. In *Proceedings of the 19th international conference on World wide web*, pages 1001–1010. ACM.

Xiao Yu, Hao Ma, Bo-June (Paul) Hsu, and Jiawei Han. 2014. On building entity recommender systems using user click log and freebase knowledge. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 263–272, New York, NY, USA. ACM.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical report, Technical Report CMU-CALD-02-107, Carnegie Mellon University.

Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919.

Queries as a Source of Lexicalized Commonsense Knowledge

Marius Paşca
Google Inc.
1600 Amphitheatre Parkway
Mountain View, California 94043
mars@google.com

Abstract

The role of Web search queries has been demonstrated in the extraction of attributes of instances and classes, or of sets of related instances and their class labels. This paper explores the acquisition of open-domain commonsense knowledge, usually available as factual knowledge, from Web search queries. Similarly to previous work in open-domain information extraction, knowledge extracted from text - in this case, from queries - takes the form of lexicalized assertions associated with open-domain classes. Experimental results indicate that facts extracted from queries complement, and have competitive accuracy levels relative to, facts extracted from Web documents by previous methods.

1 Introduction

Motivation: Open-domain information extraction methods (Etzioni et al., 2005; Pennacchiotti and Pantel, 2009; Wang and Cohen, 2009; Kozareva and Hovy, 2010; Wu et al., 2012) aim at distilling text into knowledge assertions about classes, instances and relations among them (Etzioni et al., 2011). Ideally, the assertions would complement or expand upon knowledge available in popular, human-created resources such as Wikipedia (Remy, 2002) and Freebase (Bollacker et al., 2008), reducing costs and scalability issues associated with manual editing, curation and maintenance of knowledge.

Candidate knowledge assertions extracted from text for various instances and classes (Banko et al., 2007; Cafarella et al., 2008; Wu and Weld, 2010) must satisfy several constraints in order to be useful. First, their boundaries must be correctly identified within the larger context (e.g., a document sentence) from which they are extracted. In practice, this is a challenge with arbitrary Web documents, where even instances and class labels that are complex nouns, and thus still shorter than candidate assertions, are difficult to precisely detect and pick out from surrounding text (Downey et al., 2007). This causes the extraction of assertions like *companies* may *"be in the process"*, *hurricanes* may *"run from june"*, or *video games* may *"make people"* (Fader et al., 2011). Second, the assertions must be correctly associated with their corresponding instance or class. In practice, tagging and parsing errors over documents of arbitrary quality may cause the extracted assertions to be associated with the wrong instances or classes. Examples are *video games* may *"watch movies"*, or *video games* may *"read a book"*. Third, the assertions, even if true, must refer to relevant properties or facts, rather than to statements of little or no practical interest to anyone. In practice, relevant properties may be difficult to distinguish from uninteresting statements in Web documents. Consequently, assertions extracted from Web documents include the facts that *companies* may *"say in a statement"*, or that *hurricanes* may *"be just around the corner"* or may *"be in effect"*.

Contributions: This paper explores the use of Web search queries, as opposed to Web documents, as a textual source from which knowledge pertaining to open-domain classes can be extracted. Previous explorations of the role of queries in information extraction include the acquisition of attributes of instances (Alfonseca et al., 2010) and of classes (Van Durme and Paşca, 2008); the acquisition of sets of related

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1081–1091,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

instances (Sekine and Suzuki, 2007; Jain and Pennacchiotti, 2010) and their class labels (Van Durme and Paşca, 2008; Pantel et al., 2012); the disambiguation of instances mentioned in queries relative to entries in external knowledge repositories (Pantel and Fuxman, 2011) and its application in query expansion (Dalton et al., 2014); and the extraction of the most salient of the instances mentioned in a given Web document (Gamon et al., 2013). In comparison, this paper shows that queries also lend themselves to the acquisition of factual knowledge beyond attributes, like the facts that *companies* may *"buy back stock"*, *hurricanes* may *"need warm water"*, and *video games* may *"come out on tuesdays"*.

To extract knowledge assertions for diverse classes of interest to Web users, the method applies simple extraction patterns to queries. The presence of the source queries, from which the assertions are extracted, is in itself deemed evidence that the Web users who submitted the queries find the assertions to be relevant and not just random statements. Experimental results indicate that knowledge assertions extracted from queries complement, and have competitive accuracy levels relative to, knowledge extracted from Web documents by previous methods.

2 Extraction from Queries

Queries as Knowledge: Users tend to formulate their Web search queries based on knowledge that they already possess at the time of the search (Paşca, 2007). Therefore, search queries play two roles simultaneously: in addition to requesting new information, they indirectly convey knowledge in the process.

A fact corresponds to a property that, together with other properties, help define the semantics of the class and its interaction with other classes. The extraction of factual knowledge from queries starts from the intuition that, if a fact F is relevant for a class C, then users are likely to ask for various aspects of the fact F, in the context of the class C. If *companies* may *"pay dividends"* or *"get audited"*, and such properties are relatively prominent for *companies*, then users eventually submit queries to inquire about the facts.

Often, queries will be simple concatenations of keywords: *"companies pay dividends"* or perhaps *"company dividends"*, *"audit companies"*. Since there are no restrictions on the linguistic structure

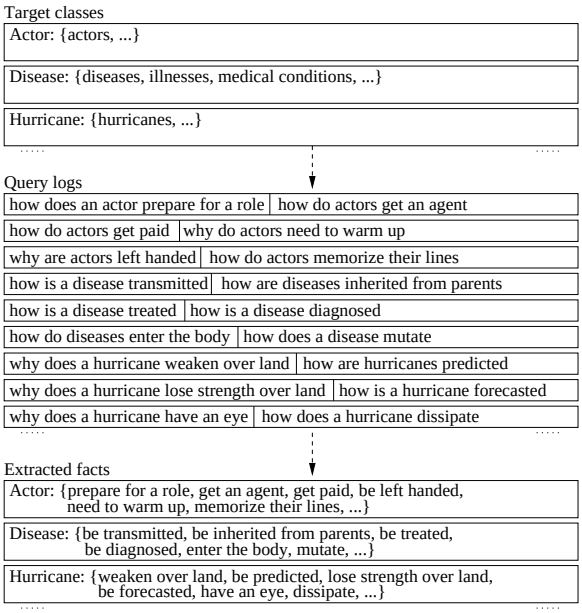


Figure 1: Overview of extraction of knowledge from Web search queries

of keyword-based queries, extracting facts from such queries would be difficult. But if queries are restricted to fact-seeking questions, the expected format of the questions makes it easier to identify the likely boundaries of the class and the fact mentioned in the queries. Queries such as *"why does a (company)$_C$ (pay dividends)$_F$"* and *"how do (companies)$_C$ (get audited)$_F$"*, follow the linguistic structure, even if minimal, imposed by formulating the query as a question. This allows one to approximate the location of the class C, possibly towards the beginning of the query; the start of the fact F, possibly as the verb immediately following the class; and the end of the fact, which possibly coincides with the end of the query.

Acquisition from Queries: The extraction method proposed in this paper takes as input a set of target classes, each of which is available as a set of class descriptors, i.e., phrases that describe the class. It also has access to a set of anonymized queries. As illustrated in Figure 1, the method selects queries that contain a class descriptor and what is deemed to be likely a fact. It outputs ranked lists of facts for each class. The extraction consists in several stages: 1) the selection of a subset of queries that refer to a class in a form that suggests the queries inquire about a fact of the class; 2) the extraction of facts, from query fragments that describe the property of interest to users submitting the queries; and 3) the aggregation and

ranking of facts of a class.

Extraction Patterns: In order to determine whether a query contains a fact for a class, the query is matched against the extraction patterns from Table 1.

The use of targeted patterns in relation extraction has been suggested before (Hearst, 1992; Fader et al., 2011; Mesquita et al., 2013). Specifically, in (Tokunaga et al., 2005), the patterns *"A of D"* or *"what is the A of D"* extract noun-phrase A attributes from queries and documents, for phrase descriptors D of the class. In our case, the patterns are constructed such that they match questions that likely inquire about the reason why, or manner in which, a relevant fact F may hold for a class C. For example, the first pattern from Table 1 matches the queries *"why does a company pay dividends"* and *"why do video games come out on tuesdays"*. These queries seek explanations for why certain properties may hold for *companies* and *video games* respectively.

A class C can be mentioned in queries through lexicalized, phrase descriptors D that capture its meaning. The descriptors D of the class C may be available as non-disambiguated items, i.e., as strings (*companies*, *firms*, *businesses*, *video games*); or as disambiguated items, that is, as pointers to knowledge base entries with a disambiguated meaning (*Company*, *Video Game*). In the first case, the matching of a query fragment, on one hand, to the portion of an extraction pattern corresponding to the class C, on the other hand, consists in simple string matching with one of the descriptors D specified for C. In the second case, the matching requires that the disambiguation of the query fragment, in the context of the query, matches the desired disambiguated meaning of C from the pattern. The subset of queries matching any of the extraction patterns, for any descriptor D of a class C, are the queries that contribute to extracting facts of the class C.

If a pattern from Table 1 employs a form of the auxiliary verb *"be"*, the extracted facts are modified by having the verb *"be"* inserted at their beginning. For example, the fact *"be stored sideways"* is extracted from the query *"why is wine stored sideways"*. In all patterns, the candidate fact is required to start with a verb that acts as the predicate of the query.

Ranking of Facts: Facts of a class C are aggregated from facts of individual class descriptors D.

Extraction Pattern → Examples of Matched Queries
why [does|did|do] [a|an|the|<nothing>] D F → why does a (company)$_D$ (pay dividends)$_F$ → why do (planes)$_D$ (take longer to fly west than east)$_F$ → why do (video games)$_D$ (come out on tuesdays)$_F$
why [is|was|were] [a|an|the|<nothing>] D F → why are (cars)$_D$ (made of steel)$_F$ → why is a (newspaper)$_D$ (written in columns)$_F$ → why is (wine)$_D$ (stored sideways)$_F$
how [does|did|do] [a|an|the|<nothing>] D F → how does a (company)$_D$ (use financial statements)$_F$ → how does (food)$_D$ (get absorbed)$_F$ → how do (stadiums)$_D$ (get cleaned)$_F$
how [is|was|were] [a|an|the|<nothing>] D F → how are (hurricanes)$_D$ (predicted)$_F$ → how is a (treaty)$_D$ (ratified)$_F$ → how is a (cell phone)$_D$ (unlocked)$_F$

Table 1: The extraction patterns match queries likely to inquire about facts of a class (D=a phrase acting as a class descriptor; F=a sequence of tokens whose first token is the head verb of the query)

A fact F is deemed more relevant for C if the fact is extracted for more of the descriptors D of the class C, and for fewer descriptors D that do not belong to the class C. Concretely, the score of a fact for a class is the lower bound of the Wilson score interval (Brown et al., 2001):

$$Score(F, C) = LowBound(Wilson(N_+, N_-))$$

where:

- the number of positive observations N_+ is the number of queries for which the fact A is extracted for some descriptor D of the class C, $|\{Query(D, A)\}_{D \in C}|$; and
- the number of negative observations N_- is the number of queries for which the fact F is extracted for some descriptors D outside of the class C, $|\{Query(D, A)\}_{D \notin C}|$.

The scores are internally computed at 95% confidence. Facts of each class are ranked in decreasing order of their scores. In case of ties, facts are ranked in decreasing order of the frequency sum of the source queries from which the facts are extracted.

3 Experimental Setting

Textual Data Sources: The experiments rely on a random sample of around 1 billion fully-anonymized Web search queries in English. The sample is drawn from queries submitted to a general-purpose Web search engine. Each query is available independently from other queries, and is accompanied by its frequency of occurrence in

Target Class (class descriptors to be looked up in queries)	
Actor (actors)	Mountain (mountains)
Aircraft (planes)	Movie (movies)
Award (awards)	NationalPark (national parks)
Battle (battles)	NbaTeam (nba teams)
Car (cars)	Newspaper (newspapers)
CartoonChar (cartoon characters)	Painter (painters)
CellPhone (cell phones)	ProgLanguage (programming languages)
ChemicalElem (elements)	Religion (religions)
City (cities)	River (rivers)
Company (companies)	SearchEngine (search engines)
Country (countries)	SkyBody (celestial bodies)
Currency (currencies)	Skyscraper (skyscrapers)
DigitalCamera (digital cameras)	SoccerClub (soccer teams)
Disease (diseases)	SportEvent (sport events)
Drug (drugs)	Stadium (stadiums)
Empire (empires)	TerroristGroup (terrorist groups)
Flower (flowers)	Treaty (treaties)
Food (foods)	University (universities)
Holiday (holidays)	VideoGame (video games)
Hurricane (hurricanes)	Wine (wines)

Table 2: Set of 40 target classes used in the evaluation of extracted facts

the query logs.

Target Classes: Table 2 shows the set of 40 target classes for evaluating the extracted facts. Similar evaluation strategies were followed in previous work (Paşca, 2007). As illustrated earlier in Figure 1, a target class consists in a small set of phrase descriptors. The phrase descriptors are selected such that they best approximate the meaning of the class. In general, the descriptors can be selected and expanded with any strategy from any source. One such possible source might be synonym sets from WordNet (Fellbaum, 1998). Following a stricter strategy, the sets of descriptors in our experiments contain only one phrase each, manually selected to match the target class. Examples are the sets of phrase descriptors {*actors*} for the class *Actor* and {*nba teams*} for *NbaTeam*. The occurrence of a descriptor (*nba teams*) in a query (*"how do nba teams make money"*) is deemed equivalent to a mention of the corresponding class (*NbaTeam*) in that query. Each set of descriptors of a class is then expanded (not shown in Table 2), to also include the singular forms of the descriptors (e.g., *nba team* for *nba teams*). Further inclusion of additional descriptors would increase the coverage of the extracted facts.

Experimental Runs: The baseline run R_D is the extraction method introduced in (Fader et al., 2011). The method produces triples of an instance or a class, a text fragment capturing a fact, and another instance or class. In these experiments, the second and third elements of each triple are concatenated together, giving pairs of an instance or a class, and a fact applying to it. The baseline run is applied to around 500 million Web documents in English. [1] In addition to the baseline run, the method introduced in this paper constitutes the second experimental run R_Q. Facts extracted by the two experimental runs are directly comparable: both are text snippets extracted from the respective sources of text - documents in the case of R_D, or queries in the case of R_Q.

Parameter Settings: Queries that match any of the extraction patterns from Table 1 are syntactically parsed (Petrov et al., 2010), in order to verify that the first token of an extracted fact is the head verb of the query. Extracted facts that do not satisfy the constraint are discarded. A positive side effect of doing so is to avoid extraction from some of the particularly subjective queries. For example, facts extracted from the queries *"why is (A) evil"* or *"why is (B) ugly"*, where *(A)* and *(B)* are the name of a company and actress respectively, are discarded.

4 Evaluation Results

Accuracy: The measurement of recall requires knowledge of the complete set of items (in our case, facts) to be extracted. Unfortunately, this number is often unavailable in information extraction tasks in general (Hasegawa et al., 2004), and fact extraction in particular. Indeed, the manual enumeration of all facts of each target class, to measure recall, is unfeasible. Therefore, the evaluation focuses on the assessment of accuracy.

Following evaluation methodology from prior work (Paşca, 2007), the top 50 facts, from a ranked lists extracted for each target class, are manually assigned correctness labels. A fact is marked as *vital*, if it must be present among representative facts of the class; *okay*, if it provides useful but non-essential information; and *wrong*, if it is incorrect (Paşca, 2007). For example, the facts *"run on kerosene"*, *"be delayed"* and *"fly wiki"* are annotated as *vital*, *okay* and *wrong* respectively for the class *Aircraft*. To compute the precision score

[1] At the time when the experiments were conducted, the facts were extracted by the baseline run from English documents in the ClueWeb collection, and were accessible at http://reverb.cs.washington.edu.

Target Class: Sample of Extracted Facts (with Source Queries)	Target Class: Sample of Extracted Facts (with Source Queries)
Actor (may): prepare for a role (how does an actor prepare for a role), get an agent (how do actors get an agent), do love scenes (how do actors do love scenes), get paid (how do actors get paid), be left handed (why are actors left handed), need to warm up (why do actors need to warm up)	Car (may): backfire (why does a car backfire), burn oil (why do cars burn oil), pull to the right (why do cars pull to the right), pull to the left (why does a car pull to the left), catch on fire (how does a car catch on fire), run hot (why do cars run hot), get repossessed (why do cars get repossessed)
Company (may): buy back stock (how does a company buy back stock), go public (why does a company go public), buy back shares (why do companies buy back shares), incorporate in delaware (why do companies incorporate in delaware), pay dividends (why does a company pay dividends), merge (how do companies merge)	Disease (may): be transmitted (how is a disease transmitted), be inherited from parents (how are diseases inherited from parents), affect natural selection (how do diseases affect natural selection), be treated (how is a disease treated), affect the conquest of the americas (how did diseases affect the conquest of the americas), be diagnosed (how is a disease diagnosed)
Hurricane (may): weaken over land (why does a hurricane weaken over land), be predicted (how are hurricanes predicted), lose strength over land (why does a hurricane lose strength over land), have an eye (why does a hurricane have an eye), be forecasted (how is a hurricane forecasted), dissipate (how does a hurricane dissipate), lose strength (how do hurricanes lose strength)	NbaTeam (may): make money (how does an nba team make money), communicate to win (how does an nba team communicate to win), want expiring contracts (why do nba teams want expiring contracts), make the playoffs (how do nba teams make the playoffs), get their names (how do nba teams get their names), do sign and trades (why do nba teams do sign and trades), lose money (how do nba teams lose money)

Table 3: Examples of facts extracted for various classes by run R_Q

Class	Precision						Class	Precision					
	@10		@20		@50			@10		@20		@50	
	R_D	R_Q	R_D	R_Q	R_D	R_Q		R_D	R_Q	R_D	R_Q	R_D	R_Q
Actor	0.60	0.85	0.57	0.85	0.60	0.83	Mountain	0.20	0.75	0.10	0.72	0.05	0.55
Aircraft	0.50	0.95	0.42	0.87	0.47	0.81	Movie	0.40	0.20	0.37	0.20	0.40	0.32
Award	0.50	0.25	0.45	0.25	0.52	0.23	NationalPark	0.40	0.70	0.32	0.72	0.30	0.69
Battle	0.25	0.45	0.42	0.46	0.38	0.44	NbaTeam	0.60	0.75	0.42	0.80	0.20	0.77
Car	0.55	0.80	0.62	0.82	0.52	0.75	Newspaper	0.25	0.80	0.32	0.55	0.44	0.59
CartoonChar	0.25	0.60	0.22	0.57	0.18	0.55	Painter	0.30	0.75	0.40	0.65	0.42	0.61
CellPhone	0.75	0.90	0.75	0.82	0.55	0.82	ProgLanguage	0.20	0.75	0.25	0.72	0.25	0.70
ChemicalElem	0.45	0.90	0.45	0.72	0.54	0.72	Religion	0.10	0.80	0.30	0.70	0.13	0.69
City	0.30	0.80	0.27	0.67	0.27	0.63	River	0.65	0.95	0.70	0.87	0.54	0.57
Company	0.60	0.95	0.57	0.95	0.53	0.91	SearchEngine	0.40	0.70	0.37	0.65	0.38	0.64
Country	0.30	0.85	0.25	0.90	0.20	0.83	SkyBody	0.55	0.00	0.32	0.00	0.28	0.00
Currency	0.40	0.90	0.25	0.85	0.22	0.73	Skyscraper	0.45	0.85	0.37	0.77	0.24	0.78
DigitalCamera	0.30	0.90	0.35	0.85	0.42	0.77	SoccerClub	0.35	0.15	0.37	0.33	0.41	0.31
Disease	0.55	0.90	0.60	0.70	0.64	0.60	SportEvent	0.30	0.00	0.27	0.00	0.32	0.00
Drug	0.20	0.95	0.30	0.87	0.40	0.78	Stadium	0.50	0.85	0.50	0.77	0.47	0.75
Empire	0.15	0.45	0.12	0.52	0.23	0.49	TerroristGroup	0.90	0.55	0.70	0.55	0.55	0.53
Flower	0.60	0.90	0.50	0.80	0.48	0.78	Treaty	1.00	0.75	0.90	0.75	0.77	0.59
Food	0.65	0.80	0.55	0.85	0.43	0.85	University	0.10	0.95	0.05	0.92	0.10	0.70
Holiday	0.30	0.25	0.17	0.22	0.19	0.14	VideoGame	0.20	0.90	0.25	0.85	0.28	0.77
Hurricane	0.40	0.80	0.37	0.77	0.32	0.73	Wine	0.70	1.00	0.60	0.87	0.56	0.70
							Average-Class	0.43	0.71	0.40	0.67	0.38	0.63

Table 4: Relative accuracy of facts extracted from documents in run R_D, vs. facts extracted from queries in run R_Q

over a set of facts, the correctness labels are converted to numeric values: *vital* to 1.0, *okay* to 0.5, and *wrong* to 0.0. Precision is the sum of the correctness values of the facts, divided by the number of facts. Table 3 shows a sample of facts extracted from queries by run R_Q, which are judged to be *vital* or *okay*.

Table 4 provides a comparison of precision at ranks 10, 20 and 50, for each of the 40 target classes and as an average over all target classes. The scores vary from one class to another and between the two runs, for example 0.22 (R_D) and 0.73 (R_Q) for the class *Currency* at rank 50, but 0.77 (R_D) and 0.59 (R_Q) for *Treaty*. Run R_Q fails to extract any facts for two of the target classes, *SkyBody* and *SportEvent*. Therefore, it receives no credit for those classes during the computation of precision.

Over all target classes, run R_Q is superior to run R_D, with relative precision boosts of 65% (0.71 vs. 0.43) at rank 10, 67% at rank 20, and 65% at rank 50. The results show that facts extracted from

Run: [Ranked Facts Extracted from Text for a Sample of Classes]
Class: Actor (may):
R_D: [do a great job, get the part, play their roles, play their parts, play their characters, be on a theatre, die aged 81, be all great, deliver their lines, portray their characters, take on a role, be best known for his role, play the role of god, be people, give great performances, bring the characters to life, wear a mask, be the one, have chemistry, turn director, read the script, ..]
R_Q: [prepare for a role, get an agent, do love scenes, get paid, be left handed, need to warm up, get started, get paid so much, memorize their lines, get ripped so fast, remember their lines, make themselves cry, learn their lines, jump out of a window in times square, lose weight so fast, play dead, be paid, kiss, remember lines, memorize lines, get discovered, get paid for movies, go uncredited, say break a leg, get their start, have perfect skin, become actors, ..]
Class: Car (may):
R_D: [get a tax write-off, can be more competitive than airline rates, be in good condition, be first for second hand cars, be in the shop, relocate to a usa firm, be in motion, come to a stop, hire companies, be in great shape, be for sale, hire service from spain, ride home, be on fire, use the autos.com, come to a halt, catch fire, be on road, be on display, go on sale, hit a tree, be available for delivery, stop in front, be a necessity, go off the road, pull out in front, hire services, run out of gas, ..]
R_Q: [backfire, burn oil, save ostriches from extinction, pull to the right, pull to the left, catch on fire, run hot, sputter, get repossessed, have a top speed, be called a car, have gears, get impounded, be called cars, go to auction, called whip, made of steel, get hot in the sun, shake at high speed, changed america, totaled, cut out, cut off while driving, fail emissions, protect from lightning, run rich, lose oil, become electrically charged, cut off, flip over, know tire pressure, have a maximum speed, require premium gas, shake at high speeds, stall out, cause acid rain, fog up, get stuck in park, need an oil change, ..]
Class: Company (may):
R_D: [say in a statement, specialize in local moves, be in the process, go out of business, have been in business, be in business, do business, file for bankruptcy, make money, be on track, say in a press release, be a place, have cut back on health insurance, state in a press release, be on the verge, save money, be in talks, have helped thousands of consumers, reduce costs, go bust, be in the midst, say in a release, be founded in 1999, be in trouble, be founded in 2000, be losing money, ..]
R_Q: [buy back stock, go public, buy back shares, incorporate in delaware, pay dividends, merge, go global, go international, use financial statements, verify education, expand internationally, go green, verify employment, need a website, choose to form as a corporation, do market research, go private, diversify, go into administration, get on angies list, pay dividend, struck off, buy back their shares, get audited, need a mission statement, repurchase common stock, spin off, get listed on the nyse, create value, distribute dividends, need a strategic plan, ..]
Class: Mountain (may):
R_D: [spot fever, meet the sea, be covered with snow, be covered in snow, be the place, come into view, be on fire, be fun, fly fishing, be volcano, be moved out of their places, enjoy the exhilaration, meet the ocean, be available for hire, keep their secrets, win the mwc in 2010, ..]
R_Q: [affect rainfall, affect the climate of an area, affect climate, be measured, be formed, be created, be made, grow, affect weather, have snow on top, affect solar radiation, affect temperature, be formed ks2, affect the weather, be built, affect people, look blue, tops cold, affect neighboring climates, be formed video, help shape the development of greek civilization, be made for kids, occur, affect the climate, be formed, be formed wikipedia, have roots, affect precipitation, exist, affect life on earth, be formed kids, float in avatar, erode, have snow on the top, affect the political character of greece, help rain form, ..]

Table 5: Comparative top facts extracted for a sample of classes from documents (R_D) or queries (R_Q)

queries have higher levels of accuracy.

Facts from Documents vs. Queries: Table 5 compares the top facts extracted by the two experimental runs for a sample of target classes. Most commonly, erroneous facts are extracted by run R_D due to the extraction of relatively uninteresting properties (a *Company* may *"say in a statement"* or *"be in the process"*). Other errors in R_D are caused by wrong boundary detection of facts within documents (a *Company* may *"be in the midst"*), or by the association of a fact with the wrong instance or class (a *Car* may *"hire companies"* or *"hire services"*).

As for facts extracted by run R_Q, they are sometimes too informal, due to the more conversational nature of queries when compared to documents. Queries may suggest that a *Car* may *"know tire pressure"*. Occasionally, similarly to facts from documents, they have wrong boundaries (a *Mountain* may *"be made for kids"* or *"be formed wikipedia"*); and they may correspond to less interesting, or too specific, properties (a *Company* may *"incorporate in delaware"*). Lastly, queries may appear to be questions, but occasionally they really are not. An example is the query *"why did the actor jump out of the window in times square"*, which may refer to a joke. When such queries match one of the extraction patterns, they produce wrong facts. Overall, Table 5 corroborates the scores from Table 4. It suggests that a) facts extracted by either R_D or R_Q still need refinement, before they can capture essential characteristics of the respective classes and nothing else; and b) facts extracted in run R_Q have higher quality than facts extracted in run R_D. Indeed, because fact-seeking queries inquire about the value (or reason, or manner) of some relations of an instance, the facts themselves tend to be more relevant than facts extracted from arbitrary document sentences.

An issue related to facts extracted from text

is their ability to capture the kind of "obvious" commonsense knowledge (Zang et al., 2013) that would be essential for machine-driven reasoning. If it is obvious that *"teachers give lectures"*, how likely is it for such information to be explicitly stated in documents or, even more interestingly, inquired about in queries? Anecdotal evidence gathered during experimentation suggests that queries do produce many commonsense facts, perhaps even surprisingly so given that a) queries tend to be shorter and grammatically simpler than document sentences; and b) the patterns in Table 1 are relatively more restrictive than the patterns used in (Fader et al., 2011). Indeed, the patterns in Table 1, when applied to queries like *"why do teachers give homework"*, *"why do teachers give grades"*, actually produce commonsense knowledge that *teachers* give *homework*, *grades* (to their students). In fact, the quality of equivalent facts extracted from documents in (Fader et al., 2011) may be lower. Concretely, facts extracted in (Fader et al., 2011) state that what *teachers* give is *students*, *class*, *homework* and *feedback*, in this order. The first two of these extractions are errors, likely caused by the incorrect detection of complex entities and their inter-dependencies in document sentences (Downey et al., 2007).

A necessary condition for the usefulness of extracted facts is that the source text contain consistent, true information. But both documents and queries may contain contradictory or false information, whether due to unsupported conjectures, unintended errors or systematic campaigns that fall under the scope of adversarial information retrieval (Castillo and Davison, 2011). The phenomena potentially affect prior work on Web-based open-domain extraction, and potentially affect the quality of facts extracted from queries in this paper. For example, facts extracted from queries like *"why do companies like obamacare"* and *"why do companies hate obamacare"* would be inconsistent, if not incorrect.

Occasionally, facts extracted from the two text sources refer to the same properties. For example, a *VideoGame* may *"be good for the hand-eye coordination"*, according to documents; and may *"improve hand eye coordination"*, according to queries. Nevertheless, facts derived from queries likely serve as a complement, rather than replacement, of facts from documents. In particular, facts extracted from queries make no attempt to isolate the value of the respective properties, whereas facts extracted from documents usually do.

Stricter Comparison of Data Sources: In the experiments described so far, distinct sets of patterns are applied in the experimental runs to documents vs. queries. More precisely, run R_D applies the patterns introduced in (Fader et al., 2011) to document sentences, whereas run R_Q the patterns shown in Table 1 to queries. To more accurately gauge the role of queries vs. documents in extracting facts from unstructured text, additional experiments isolate the effect of extracting facts from different types of data sources. For this purpose, the same set of patterns from Table 1 is matched against the sentences from around 500 million Web documents. The patterns are applied to document sentences converted to lowercase, similarly to how they are applied to queries. This corresponds to a new experimental run R_{DS}, which employs the same patterns as the earlier run R_Q but runs over document sentences instead of queries.

As an average over the target classes, the precision of facts extracted by run R_{DS} is 0.50, 0.47 and 0.44 at ranks 10, 20 and 50 respectively. Two conclusions can be drawn from comparing these scores with the average scores from the earlier Table 4. First, the average precision of run R_{DS} is higher than for run R_D. In other words, when extracting from document sentences in R_{DS} and R_D, the patterns proposed in our method give fewer and more accurate facts than the patterns from (Fader et al., 2011). Second, although R_{DS} is more accurate than R_D, it is less accurate than run R_Q. Note that, among the top 50 facts extracted for each target class by runs R_{DS} and R_Q, an average of 13% of the facts are extracted by both runs. There are several phenomena contributing to the difference in precision. While inherently noisy, queries tend to be more compact, and therefore more focused. In comparison, document sentences matching the patterns are often more convoluted (e.g., *"who do cities keep building stadiums despite study after study showing they do not make money"*, or *"how does a company go from low associate satisfaction to #15 on the fortune 100 best list in the midst of a crippling recession"*). Furthermore, both queries and sentences may not be useful questions from which relevant facts can be extracted, even when they match the extraction patterns. However, anecdotal evidence suggests

that this happens more frequently with document sentences than with queries. Examples include document sentences extracted from sites aggregating jokes (*"why did the cell phone ask to see the psychologist"*). The results confirm that queries represent an intriguing resource for fact extraction, providing a useful complement to document sentences for the purpose of extracting facts.

Quantitative Results: From the set of queries used as input in run R_Q, 3.8% of all queries start with *why* or *how*. In turn, 13.6% of them match one of the extraction patterns from Table 1, and therefore produce a candidate fact in R_Q. In the case of run R_{DS}, 18.7% of the document sentences that start with *why* or *how* match one of the patterns from Table 1.

Choice of Extraction Patterns: The sets of patterns sometimes employed in relation extraction from documents (Hearst, 1992) occasionally benefit from the addition of new patterns, or the refinement into more specific patterns (Kozareva et al., 2008). Similarly, the set of patterns proposed in Table 1, which targets the extraction of facts from queries, is neither exhaustive nor final. Other patterns beyond *why* and *how* may prove useful, whether they rely on relatively less frequent *when* and *where* queries, or extract relations containing underspecified arguments from *who* or *what* queries.

When applied to queries in run R_Q, the *how* patterns from Table 1 match 3.3 times more queries than the *why* patterns.

In separate experiments, *why* vs. *how* patterns from Table 1 are temporarily disabled. The ratio of facts extracted on average per target class in run R_Q diminishes from 100% (with both patterns) to 30% (with *why* only) or 70% (with *how* only). Overall, no difference in accuracy is observed over facts extracted by *why* vs. *how* patterns.

Choice of Phrase Descriptors: A separate experiment investigates the impact of expanding the sets of phrase descriptors associated with each target class. Among many possible strategies, each set of phrase descriptors associated with a target class is expanded automatically, using WordNet and distributional similarities. For this purpose, for each target class, the set of synonyms and hyponyms of all senses, if any, available in WordNet for each phrase descriptor is intersected with the set of the 50 most distributionally similar phrases, if any, available for each phrase descriptor. The original set of phrase descriptors of each target class is then expanded, to include the phrases from the intersected set, if any.

A repository of distributionally similar phrases is collected in advance following (Lin and Wu, 2009; Pantel et al., 2009), from a sample of around 200 million Web documents. Their intersection with phrases collected from WordNet aims at reducing the noise associated with expansion solely from either source. For example, for the class *Actor*, the set of phrases {*player*, *worker*, *heavy*, *plant*, *actress*, *comedian*, *film star*, ..} is collected from WordNet for the descriptor *actors*. The set is intersected with the set of phrases {*film stars*, *performers*, *comedians*, *actresses*, ..} most distributionally similar to *actors*. Examples of sets of phrase descriptors after expansion are {*actors*, *actresses*, *comedians*, *players*, *film stars*, ..}, for the class *Actor*; and {*battles*, *naval battles*, *fights*, *skirmishes*, *struggles*, ..}, for *Battle*.

On average, the sets of phrase descriptors associated with each target class contains 2 vs. 11 phrases, before vs. after expansion. Some of the sets of phrase descriptors, such as for the target classes *CartoonChar* and *DigitalCamera*, remain unchanged after expansion. As expected, expansion may introduce noisy phrase descriptors, such as *players* for *Actor*, or *diets* for *Food*. The presence of noisy phrase descriptors lowers the precision of the extracted facts. After expansion, the precision scores of R_Q, as an average over all target classes, become smaller by 6% (0.71 vs. 0.67), at rank 10; 6% (0.67 vs. 0.63), at rank 20; and 7% (0.63 vs. 0.59), at rank 50. Expansion also affects relative coverage, increasing the average number of facts extracted by R_Q per target class by more than twice (i.e., by a factor of 2.6).

Redundant Facts: Due to lexical variation in the source text fragments, some of the extracted facts may be near-duplicates of one another. In general, the phenomenon affects facts extracted from text by previous methods (Van Durme and Paşca, 2008; Etzioni et al., 2011; Fader et al., 2011). In particular, it affects facts extracted from both documents or queries in our experiments. For example, the facts extracted from documents for *Actor* include *"play their roles"*, *"play their parts"*, *"play their characters"* and *"portrayed their characters"*. Separately, the facts *"memorize their lines"*, *"remember their lines"* and *"learn their lines"* are extracted from queries for the class

Actor. The automatic detection of equivalent facts would increase the usefulness of facts extracted from text in general, and of facts extracted by the method presented here in particular.

5 Related Work

A variety of methods address the more general task of acquisition of open-domain relations from text, e.g., (Banko et al., 2007; Carlson et al., 2010; Wu and Weld, 2010; Fader et al., 2011; Lao et al., 2011; Mausam et al., 2012; Lopez de Lacalle and Lapata, 2013). In general, relations extracted from document sentences (e.g., *"Claude Monet was born in Paris"*) are tuples of an argument (*claude monet*), a text fragment acting as the lexicalized relation (*was born in*), and another argument (*paris*) (cf. (Banko et al., 2007; Fader et al., 2011; Mausam et al., 2012)). For convenience, the relation and second argument may be concatenated into a fact applying to the first argument, as in *"was born in paris"* for *claude monet*. Relatively shallow tools like part of speech taggers, or more complex tools like semantic taggers (Van Durme et al., 2008; Van Durme et al., 2009) can be employed in order to extract relations from document sentences. The former choice scales better to Web documents of arbitrary quality, whereas the latter could be more accurate over high-quality documents such as news articles (Mesquita et al., 2013). In both cases, document sentences mentioning an instance or a class may refer to properties of the instance that people other than the author of the document are less likely to inquire about. Consequently, even top-ranked extracted relations occasionally include less informative ones, such as *"come into view"* for *mount rainier*, *"be on the table"* for *madeira wine*, or *"allow for features"* for *javascript* (Fader et al., 2011).

Data available within Web documents, from which relations are extracted in previous work, includes unstructured (Banko et al., 2007; Fader et al., 2011), structured (Raju et al., 2008) and semi-structured text (Yoshinaga and Torisawa, 2007; Pasupat and Liang, 2014), layout formatting tags (Wong et al., 2008), itemized lists or tables (Cafarella et al., 2008). Another source is human-compiled resources (Wu and Weld, 2010) including infoboxes and category labels (Nastase and Strube, 2008; Hoffart et al., 2013; Wang et al., 2013; Flati et al., 2014) in Wikipedia, or topics and relations in Freebase (Weston et al., 2013; Yao and Van Durme, 2014).

Whether Web search queries are a useful textual data source for open-domain information extraction has been investigated in several tasks. Examples are collecting unlabeled sets of similar instances (Jain and Pennacchiotti, 2010), extracting attributes of instances (Alfonseca et al., 2010; Paşca, 2014), identifying mentions in queries of instances defined in a manually-created resource (Pantel et al., 2012), and extracting the most salient of the instances mentioned within Web documents (Gamon et al., 2013).

Other previous work shares the intuition that the submission of Web search queries is influenced by, and indicative of, various relations. Relations are loosely defined, either by approximating them via distributional similarities (Alfonseca et al., 2009), or by exploring the acquisition of untyped, similarity-based relations from query logs (Baeza-Yates and Tiberi, 2007). In both cases, the computed relations hold among full-length queries. Untyped relations can also be identified among query terms for the purpose of query reformulation (Wang and Zhai, 2008). More generally, the choice of query substitutions may reveal various relations among full queries or query terms (Jones et al., 2006), but requires individual queries to be connected to one another via query sessions or via search-result click-through data.

6 Conclusion

Anonymized search queries submitted by Web users represent requests for knowledge. Collectively, they can also be seen as informal, lexicalized knowledge assertions. By asking about a property of some class, fact-seeking queries implicitly assert the relevance of the property for the class.

Since Web search queries refer to properties that Web users are collectively interested in, factual knowledge extracted from queries tends to be more relevant than facts extracted from arbitrary documents using previous methods. Current work explores the extraction of facts from implicit rather than explicit fact-seeking questions, that is, from queries that do not start with a question prefix; and the combination of queries as a source of more accurate facts, and documents as a source of more numerous facts.

References

E. Alfonseca, K. Hall, and S. Hartmann. 2009. Large-scale computation of distributional similarities for queries. In *Proceedings of the 2009 Conference of the North American Association for Computational Linguistics (NAACL-HLT-09), Short Papers*, pages 29–32, Boulder, Colorado.

E. Alfonseca, M. Paşca, and E. Robledo-Arnuncio. 2010. Acquisition of instance attributes via labeled and related instances. In *Proceedings of the 33rd International Conference on Research and Development in Information Retrieval (SIGIR-10)*, pages 58–65, Geneva, Switzerland.

R. Baeza-Yates and A. Tiberi. 2007. Extracting semantic relations from query logs. In *Proceedings of the 13th ACM Conference on Knowledge Discovery and Data Mining (KDD-07)*, pages 76–85, San Jose, California.

M. Banko, Michael J Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2670–2676, Hyderabad, India.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 International Conference on Management of Data (SIGMOD-08)*, pages 1247–1250, Vancouver, Canada.

L. Brown, T. Cai, and A. DasGupta. 2001. Interval estimation for a binomial proportion. *Statistical Science*, 16(2):101–117.

M. Cafarella, A. Halevy, D. Wang, E. Wu, and Y. Zhang. 2008. WebTables: Exploring the power of tables on the Web. In *Proceedings of the 34th Conference on Very Large Data Bases (VLDB-08)*, pages 538–549, Auckland, New Zealand.

A. Carlson, J. Betteridge, R. Wang, E. Hruschka, and T. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the 3rd ACM Conference on Web Search and Data Mining (WSDM-10)*, pages 101–110, New York.

C. Castillo and B. Davison. 2011. Adversarial web search. *Journal of Foundations and Trends in Information Retrieval*, 4(5):377–486.

J. Dalton, L. Dietz, and J. Allan. 2014. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th International Conference on Research and Development in Information Retrieval (SIGIR-14)*, pages 365–374, Gold Coast, Queensland, Australia.

D. Downey, M. Broadhead, and O. Etzioni. 2007. Locating complex named entities in Web text. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2733–2739, Hyderabad, India.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the Web: an experimental study. *Artificial Intelligence*, 165(1):91–134.

O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*, pages 3–10, Barcelona, Spain.

A. Fader, S. Soderland, and O. Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*, pages 1535–1545, Edinburgh, Scotland.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database and Some of its Applications*. MIT Press.

T. Flati, D. Vannella, T. Pasini, and R. Navigli. 2014. Two is bigger (and better) than one: the Wikipedia Bitaxonomy project. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14)*, pages 945–955, Baltimore, Maryland.

M. Gamon, T. Yano, X. Song, J. Apacible, and P. Pantel. 2013. Identifying salient entities in web pages. In *Proceedings of the 22nd International Conference on Information and Knowledge Management (CIKM-13)*, pages 2375–2380, Burlingame, California.

T. Hasegawa, S. Sekine, and R. Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 415–422, Barcelona, Spain.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 539–545, Nantes, France.

J. Hoffart, F. Suchanek, K. Berberich, and G. Weikum. 2013. YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence Journal. Special Issue on Artificial Intelligence, Wikipedia and Semi-Structured Resources*, 194:28–61.

A. Jain and M. Pennacchiotti. 2010. Open entity extraction from Web search query logs. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-10)*, pages 510–518, Beijing, China.

R. Jones, B. Rey, O. Madani, and W. Greiner. 2006. Generating query substitutions. In *Proceedings of the 15h World Wide Web Conference (WWW-06)*, pages 387–396, Edinburgh, Scotland.

Z. Kozareva and E. Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*, pages 1110–1118, Cambridge, Massachusetts.

Z. Kozareva, E. Riloff, and E. Hovy. 2008. Semantic class learning from the Web with hyponym pattern linkage graphs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 1048–1056, Columbus, Ohio.

N. Lao, T. Mitchell, and W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*, pages 529–539, Edinburgh, Scotland.

D. Lin and X. Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP-09)*, pages 1030–1038, Singapore.

O. Lopez de Lacalle and M. Lapata. 2013. Unsupervised relation extraction with general domain knowledge. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*, pages 415–425, Seattle, Washington.

Mausam, M. Schmitz, S. Soderland, R. Bart, and O. Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-12)*, pages 523–534, Jeju Island, Korea.

F. Mesquita, J. Schmidek, and D. Barbosa. 2013. Effectiveness and efficiency of open relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*, pages 447–457, Seattle, Washington.

V. Nastase and M. Strube. 2008. Decoding Wikipedia categories for knowledge acquisition. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI-08)*, pages 1219–1224, Chicago, Illinois.

M. Paşca. 2007. Organizing and searching the World Wide Web of facts - step two: Harnessing the wisdom of the crowds. In *Proceedings of the 16th World Wide Web Conference (WWW-07)*, pages 101–110, Banff, Canada.

M. Paşca. 2014. Acquisition of noncontiguous class attributes from Web search queries. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL-14)*, pages 386–394, Gothenburg, Sweden.

P. Pantel and A. Fuxman. 2011. Jigs and lures: Associating web queries with structured entities. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-11)*, pages 83–92, Portland, Oregon.

P. Pantel, E. Crestan, A. Borkovsky, A. Popescu, and V. Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 938–947, Singapore.

P. Pantel, T. Lin, and M. Gamon. 2012. Mining entity types from query logs via user intent modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL-12)*, pages 563–571, Jeju Island, Korea.

P. Pasupat and P. Liang. 2014. Zero-shot entity extraction from Web pages. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14)*, pages 391–401, Baltimore, Maryland.

M. Pennacchiotti and P. Pantel. 2009. Entity extraction via ensemble semantics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 238–247, Singapore.

S. Petrov, P. Chang, M. Ringgaard, and H. Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*, pages 705–713, Cambridge, Massachusetts.

S. Raju, P. Pingali, and V. Varma. 2008. An unsupervised approach to product attribute extraction. In *Proceedings of the 31st International Conference on Research and Development in Information Retrieval (SIGIR-08)*, pages 35–42, Singapore.

M. Remy. 2002. Wikipedia: The free encyclopedia. *Online Information Review*, 26(6):434.

S. Sekine and H. Suzuki. 2007. Acquiring ontological knowledge from query logs. In *Proceedings of the 16th World Wide Web Conference (WWW-07), Posters*, pages 1223–1224, Banff, Canada.

K. Tokunaga, J. Kazama, and K. Torisawa. 2005. Automatic discovery of attribute words from Web documents. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP-05)*, pages 106–118, Jeju Island, Korea.

B. Van Durme and M. Paşca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI-08)*, pages 1243–1248, Chicago, Illinois.

B. Van Durme, T. Qian, and L. Schubert. 2008. Class-driven attribute extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*, pages 921–928, Manchester, United Kingdom.

B. Van Durme, P. Michalak, and L. Schubert. 2009. Deriving generalized knowledge from corpora using Wordnet abstraction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*, pages 808–816, Athens, Greece.

R. Wang and W. Cohen. 2009. Automatic set instance extraction using the Web. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP-09)*, pages 441–449, Singapore.

X. Wang and C. Zhai. 2008. Mining term association patterns from search logs for effective query reformulation. In *Proceedings of the 17th International Conference on Information and Knowledge Management (CIKM-08)*, pages 479–488, Napa Valley, California.

Z. Wang, Z. Li, J. Li, J. Tang, and J. Pan. 2013. Transfer learning based cross-lingual knowledge extraction for Wikipedia. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-13)*, pages 641–650, Sofia, Bulgaria.

J. Weston, A. Bordes, O. Yakhnenko, and N. Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*, pages 1366–1371, Seattle, Washington.

T. Wong, W. Lam, and T. Wong. 2008. An unsupervised framework for extracting and normalizing product attributes from multiple Web sites. In *Proceedings of the 31st International Conference on Research and Development in Information Retrieval (SIGIR-08)*, pages 35–42, Singapore.

F. Wu and D. Weld. 2010. Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 118–127, Uppsala, Sweden.

W. Wu, , H. Li, H. Wang, and K. Zhu. 2012. Probase: a probabilistic taxonomy for text understanding. In *Proceedings of the 2012 International Conference on Management of Data (SIGMOD-12)*, pages 481–492, Scottsdale, Arizona.

X. Yao and B. Van Durme. 2014. Information extraction over structured data: Question Answering with Freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14)*, pages 956–966, Baltimore, Maryland.

N. Yoshinaga and K. Torisawa. 2007. Open-domain attribute-value acquisition from semi-structured texts. In *Proceedings of the 6th International Semantic Web Conference (ISWC-07), Workshop on Text to Knowledge: The Lexicon/Ontology Interface (OntoLex-2007)*, pages 55–66, Busan, South Korea.

L. Zang, C. Cao, Y. Cao, Y. Wu, and C. Cao. 2013. A survey of commonsense knowledge acquisition. *Journal of Computer Science and Technology*, 28(4):689–719.

Question Answering over Linked Data Using First-order Logic*

Shizhu He, Kang Liu, Yuanzhe Zhang, Liheng Xu and Jun Zhao
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
{shizhu.he, kliu, yzzhang, lhxu, jzhao}@nlpr.ia.ac.cn

Abstract

Question Answering over Linked Data (QALD) aims to evaluate a question answering system over structured data, the key objective of which is to translate questions posed using natural language into structured queries. This technique can help common users to directly access open-structured knowledge on the Web and, accordingly, has attracted much attention. To this end, we propose a novel method using first-order logic. We formulate the knowledge for resolving the ambiguities in the main three steps of QALD (phrase detection, phrase-to-semantic-item mapping and semantic item grouping) as first-order logic clauses in a Markov Logic Network. All clauses can then produce interacted effects in a unified framework and can jointly resolve all ambiguities. Moreover, our method adopts a pattern-learning strategy for semantic item grouping. In this way, our method can cover more text expressions and answer more questions than previous methods using manually designed patterns. The experimental results using open benchmarks demonstrate the effectiveness of the proposed method.

1 Introduction

With the rapid development of the Web of Data, many RDF datasets have been published as Linked Data (Bizer et al., 2009), such as DBpedia (Auer et al., 2007), Freebase (Bollacker et al., 2008) and YAGO (Suchanek et al., 2007). The growing amount of Linked Data contains a wealth of knowledge, including entities, classes and relations. Moreover, these linked data usually have complex structures and are highly heterogeneous. As a result, there are gaps for users regarding access. Although a few experts can write queries using structured languages (such as SPARQL) based on their needs, this skill cannot be easily utilized by common users (Christina and Freitas, 2014). Thus, providing user-friendly, simple interfaces to access these linked data becomes increasingly more urgent.

Because of this, question answering over linked data (QALD) (Walter et al., 2012) has recently received much interest, and most studies on this topic have focused on translating natural language questions into structured queries (Freitas and Curry, 2014; Yahya et al., 2012; Unger et al., 2012; Shekarpour et al., 2013; Yahya et al., 2013; Bao et al., 2014; Zou et al., 2014). For example, with respect to the question

"Which software has been developed by organizations founded in California, USA?",

the aim is to automatically convert this utterance into an SPARQL query that contains the following *subject-property-object (SPO)* triple format: ⟨**?url rdf:type dbo:Software, ?url dbo:developer ?x1, ?x1 rdf:type dbo:Company, ?x1 dbo:foundationPlace dbr:California**⟩[1].

To fulfill this objective, existing systems (Lopez et al., 2006; Unger et al., 2012; Yahya et al., 2012; Zou et al., 2014) usually adopt a pipeline framework that contains four major steps: 1) decomposing the question and detecting phrases, 2) mapping the detected phrases into semantic items of Linked Data, 3) grouping the mapped semantic items into semantic triples, and 4) generating the correct SPARQL query.

However, completing these four steps and constructing such a structured query is not easy. The first three steps mentioned above are subject to the

*Shizhu He and Kang Liu have equal contribution to this work.

[1]The prefixes in semantic items indicate the source of their vocabularies.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1092–1103,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

problem of ambiguity, which is the major challenge in QALD. Using the question mentioned above as an example, we can choose *California* or *California, USA* when detecting phrases, the phrase *California* can be mapped to the entity **California_State** or **California_Film**, and the class **Software** (mapped from the phrase *software*) can be matched with the first argument of the relation **producer** or **developer** (these two relations can be mapped from the phrase *developed*). Previous methods (Lopez et al., 2006; Lehmann et al., 2012; Freitas and Curry, 2014) have usually performed disambiguation at each step only, and the subsequent step was performed based on the disambiguation results in the previous step(s). However, we argue that the three steps mentioned above have mutual effects. In the previous example, the phrase *founded in (verb)* can be mapped to the entities (**Founding_of_Rome** and **Founder_(company)**), classes (**Company** and **Department**) or relations (**foundedBy** and **foundationPlace**). If we know that the phrase *California* can refer to the entity **California_State**, and which can be the second argument of the relation **foundationPlace**, together with a *verb* phrase being more likely to be mapped to *Relation*, we should map the phrase *founded in* to **foundationPlace** in this question. **Thus, we aim to determine if joint disambiguation is better than individual disambiguation.** (**Question One**)

In addition, previous systems usually employed manually designed patterns to extract predicate-argument structures that are used to guide the disambiguation process in the three steps mentioned above (Yahya et al., 2012; Unger et al., 2012; Zou et al., 2014). For example, (Yahya et al., 2012) used only three dependency patterns to group the mapped semantic items into semantic triples. Nevertheless, these three manually designed patterns miss many cases because of the diversity of the question expressions. We gathered statistics on 144 questions and found that the macro-average F1 and micro-average F1 of the three patterns[2] used in (Yahya et al., 2012) are only 62.8 and 66.2%, respectively. Furthermore, these specially designed patterns may not be valid with variations in domains or languages. **Therefore, another important question arises: can we automatically learn rules or patterns to achieve the same objective?** (**Question Two**)

Focusing on the two problems mentioned above, this paper proposes a novel algorithm based on a learning framework, Markov Logic Networks (MLNs) (Richardson and Domingos, 2006), to learn a joint model for constructing structured queries from natural language utterances. MLN is a statistical relational learning framework that combines first-order logic and Markov networks. The appealing property of MLN is that it is readily interpretable by humans and that it is a natural framework for performing joint learning. We formulate the knowledge for resolving the ambiguities in the main three steps of QALD (phrase detection, phrase-to-semantic-item mapping and semantic item grouping) as first-order logic clauses in an MLN. In the framework of MLN, all clauses will produce interacted effects that jointly resolve all problems into a unified process. In this way, the result in each step can be globally optimized. Moreover, in contrast to previous methods, we adopt a learning strategy to automatically learn the patterns for semantic item grouping. We design several meta patterns as opposed to the specific patterns. In addition, these meta patterns are formulated as the first-order logic formulas in the MLN. The specific patterns can be generated by these meta patterns based on the training data. The model will learn the weights of each clause to determine the most effective patterns for semantic triple construction. In this way, with little effort, our approach can cover more semantic expressions and answer more questions than previous methods, which depend on manually designed patterns.

We evaluate the proposed method using several benchmarks (QALD-1, QALD-3, QALD-4). The experimental results demonstrate the advantage of the joint disambiguation process mentioned above. They also prove that our approach, employing MLN to automatically learn the patterns of semantic triple grouping, is effective. Our system can answer more questions and obtain better performance than the traditional methods based on manually designed heuristic rules.

2 Background

2.1 Linked Data Sources

Linked Data consist of many relational data, which are usually inter-linked as subject-property-object (*SPO*) triple statements (such as using the **owl:sameAs** relation). In this paper, we mainly use

[2]They are 1) verbs and their arguments, 2) adjectives and their arguments and 3) propositionally modified tokens and objects of prepositions.

Subject(Arg1)	Relation(Property)	Object(Arg2)
ProgrammingLanguage	subClassOf	Software
Java_(programming_language)	type	Software
Java_(programming_language)	developer	Oracle_Corporation
Oracle_Corporation	foundationPlace	California_(State)
foundationPlace	domain	Organisation
California_(State)	label	"California"
California_(1977_film)	label	"California"
Oracle_Corporation	numEmployees	118119(xsd:integer)

Figure 1: Sample knowledge base facts.

DBpedia[3] and some classes from Yago[4]. These knowledge bases (*KBs*) are composed of many ontological and instance statements, and all statements are expressed by *SPO* triple facts. Figure 1 shows some triple fact samples from DBpedia.

Each fact is composed of three **semantic items**. A semantic item can be an entity (**California_(State)**, **Oracle_Corporation**, etc.), a class (**Software**, **Organisation**, etc.) or a relation (called a property or predicate in some occasions). Some entities are literals including strings, numbers and dates (**118119(xsd:integer)**, etc.). Relations contain standard Semantic Web relations (**subClassOf**, **type**, **domain** and **label**) and ontological relations (**developer**, **foundationPlace** and **numEmployees**).

2.2 Task Statement

Given a knowledge base (*KB*), our objective is to translate a natural language question q_{NL} into a formal language query q_{FL} that targets the semantic vocabularies given by the *KB*, and the query q_{FL} should capture the user information needs expressed by q_{NL}.

Following (Yahya et al., 2012), we focus on the factoid questions, and the answers to such questions are an entity or a set of entities. We ignore the questions that need the aggregation[5] (max/min, etc.) and negation operations. That is, we generate queries that consist of a plentiful number of triple patterns, which are multiple conjunctions of *SPO* search conditions.

3 Framework

Figure 2 shows the entire framework of our system for translating a question into a formal SPARQL query. The first three steps address the input question through 1) Phrase Detection (detecting possible phrases), 2) Phrase Mapping (mapping all phrase candidates to the corresponding semantic items), and 3) Feature Extraction (extracting the linguistic features and semantic item features from the question and the Linked Data, respectively). As a result, a space of candidates is constructed, including possible phrases, mapped semantic items and the possible argument match relations among them. Next, the fourth step (Inference) formulates the joint disambiguation as a generalized inference task. We employ rich features and constraints (including hard and soft constraints) to infer a joint decision through an MLN. Finally, with the inference results, we can construct a semantic item query graph and generate an executable SPARQL query. In the following subsection, we demonstrate each step in detail.

1) Phrase detection. In this step, we detect phrases (sequences of tokens) that probably indicate semantic items in the *KB*. We do not use a named entity recognizer (NER) because of its low coverage. We perform testing on two commonly used question corpora, QALD-3 and free917[6], using the Stanford NER tool[7]. The results demonstrate that only 51.5 and 23.8% of the NEs are correctly recognized, respectively. To avoid missing useful phrases, we retain all n-grams as phrase candidates, and then use some rules to filter them. The rules include the following: the span length must be less than 4 (accepting that all contiguous tokens are capitalizations), the POS tag of the start token must be *jj*, *nn*, *rb* and *vb*, all contiguous capitalization tokens must not be split, etc. For instance, *software*, *developed by*, *organizations*, *founded in* and *California* are detected in the example of the first section.

2) Phrase mapping. After the phrases are detected, each phrase can be mapped to the corresponding semantic item in *KB* (entity, class and relation). For example, *software* is mapped to **dbo:Software**, **dbo:developer**, etc., and *California* is mapped to **dbr:California**, **dbr:California_(wine)**, etc. For different types of semantic items, we use different techniques. For mapping phrases to entities, considering that the entities in DBpedia and Wikipedia are consistent, we employ anchor, redirection and disambiguation information from Wikipedia. For mapping phrases to classes, considering that classes have lexical variation, especially synonyms, e.g., **dbo:Film** can be mapped

[3]http://dbpedia.org/

[4]http://www.mpi-inf.mpg.de/yago-naga/yago/

[5]We can address the *count* query questions, which will be explained in Section 3.

[6]http://www.cis.temple.edu/∼yates/open-sem-parsing/index.html

[7]http://nlp.stanford.edu/software/CRF-NER.shtml

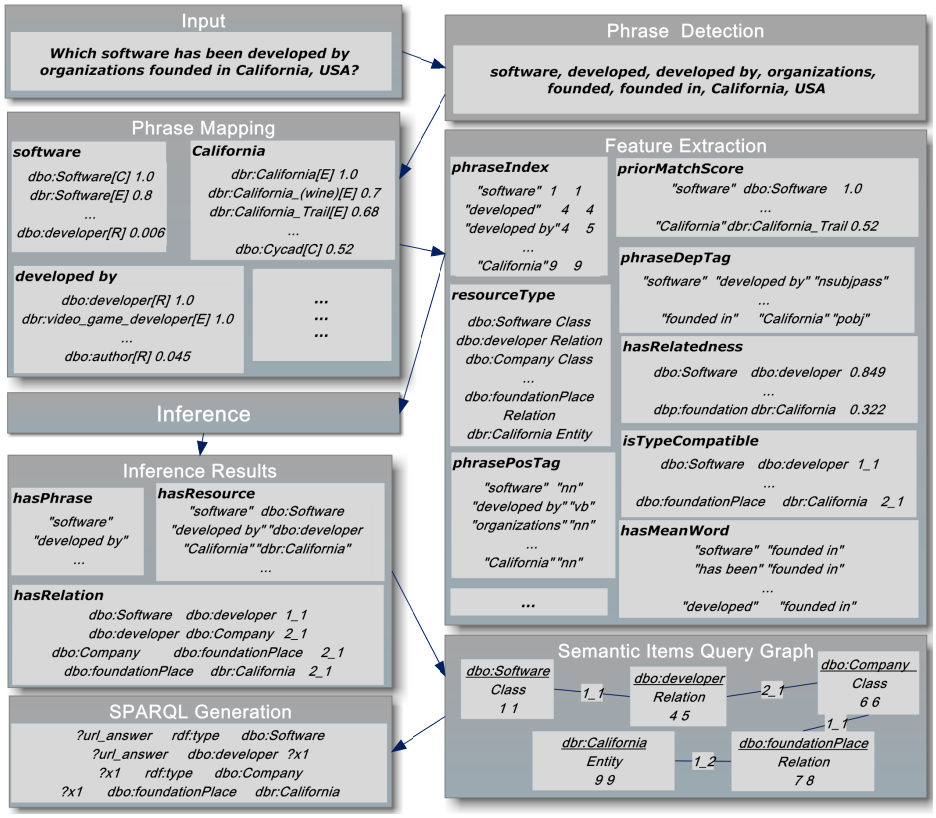


Figure 2: Framework of our system.

from *film*, *movie* and *show*, we compute the similarity between the phrase and the class in the *KB* with the *word2vec* tool[8]. The *word2vec* tool computes fixed-length vector representations of words with a recurrent-neural-network based language model (Mikolov et al., 2010). The similarity scoring methods are introduced in Section 4.2. Then, the top-N most similar classes for each phrase are returned. For mapping phrases to relations, we employ the resources from PATTY (Nakashole et al., 2012) and ReVerb (Fader et al., 2011). Specifically, we first compute the associations between the ontological relations in DBpedia and the relation patterns in PATTY and ReVerb through instance alignments as in (Berant et al., 2013). Next, if a detected phrase is matched to some relation pattern, the corresponding ontological relations in DBpedia will be returned as a candidate. This step only generates candidates for every possible mapping, and the decision of the best selection will be performed in the next step.

3) Feature extraction and joint inference. There exist ambiguities in phrase detection and in mapping phrases to semantic items. This step focuses on addressing these ambiguities and determining the argument match relations among the mapped semantic items. This is the core component of our system, and it performs disambiguation in a unified manner. First, feature extraction is performed to prepare a rich number of features from the input question and from the *KB*. Next, the disambiguation is performed in a joint fashion with a Markov Logic Network. Detailed information will be presented in Section 4.

4) Semantic item query graph construction. Based on the inference results, we construct a query graph. The vertices contain the following: the detected phrase, the token span indexes of the phrases, the mapped semantic items and their types. The edge indicates the argument match relation between two semantic items. For example, we use ***1_2*** to indicate that the first argument of an item matches the second argument of another item[9]. The right bottom in Figure 2 shows an example of this.

5) Query generation. The SPARQL queries require the grouped triples of semantic items. Thus, in this step, we convert a query graph into multiple joined semantic triples. Three interconnected semantic items, whereby it must

[8]https://code.google.com/p/word2vec/

[9]The other marks will be introduced in Section 4.2.

be ensured that the middle item is a ***relation***, are converted into a semantic triple (multiple joined facts containing variables). For example, the query graph $[$**dbo:Book[Class]** $\underset{\longleftrightarrow}{1\_2}$ **dbo:author[Relation]** $\underset{\longleftrightarrow}{1\_1}$ **dbr:Danielle_Steel[Entity]**$]$ is converted into $\langle$**?x rdf:type dbo:Book, dbr:Danielle dbo:author ?x**$\rangle$, and $[$**dbo:populationTotal[Relation]** $\underset{\longleftrightarrow}{1\_2}$ **dbo:capital[Relation]** $\underset{\longleftrightarrow}{1\_1}$ **dbr:Australia[Entity]**$]$[10] is converted into $\langle$**?x1 dbo:populationTotal ?answer, ?x1 dbo:capital dbr:Australia**$\rangle$. If the query graph only contains one vertex that indicates a class ***ClassURI***, we generate $\langle$**?x rdf:type ClassURI**$\rangle$. If the query graph only contains two connected vertexes, we append a variable to bind the missing match argument of the semantic item.

The final SPARQL query is constructed by joining the semantic item triples based on the corresponding SPARQL template. We divide the questions into three types: **Yes/No**, **Normal** and **Number**. Yes/No questions use the ***ASK WHERE*** template. Normal questions use the ***SELECT ?url WHERE*** template. Number questions first use the normal question template, and if they cannot obtain a correct answer (a valid numeric value), we use the ***SELECT COUNT(?url) WHERE*** template to generate a query again. For instance, we construct the SPARQL query ***SELECT(?url) WHERE***{ ***?url rdf:type dbo:Software. ?url dbo:developer ?x1. ?x1 rdf:type dbo:Company. ?x1 dbo:foundationPlace dbr:California.***} for this example.

4 Joint Disambiguation with MLN

In this section, we present our method for question answering over linked data using a Markov Logic Network (MLN). In the following subsections, we first briefly describe the MLN. Then, we present the predicates and the first-order logic formulas used in the model.

4.1 Markov Logic Networks

Markov logic networks combine Markov networks with first-order logic in a probabilistic framework (Richardson and Domingos, 2006). An MLN $\mathcal{M}$ consists of several weighted formulas $\{(\phi_i, w_i)\}_i$, where ϕ_i is a first order formula and w_i is the penalty (the formula's weight). In contrast to the first-order logic, whereby a formula represents a hard constraint, these logic formulas are relaxed and can be violated with penalties in the MLN. Each formula ϕ_i consists of a set of first-order predicates, logical connectors and variables. These weighted formulas define a probability distribution over a possible world. Let $\mathbf{y}$ denote a possible world. Then $p(\mathbf{y})$ is defined as follows:

$$p(\mathbf{y}) = \frac{1}{Z} exp\left(\sum_{(\phi_i, w_i)\in\mathcal{M}} w_i \sum_{\mathbf{c}\in C^{n_{\phi_i}}} f_{\mathbf{c}}^{\phi_i}(\mathbf{y})\right),$$

where each c is a binding of the free variables in ϕ_i to constants; $f_{\mathbf{c}}^{\phi_i}$ is a binary feature function that returns 1 if the ground formula that we obtain through replacing the free variables in ϕ_i with the constants in $\mathbf{c}$ under the given possible world $\mathbf{y}$ is true and is 0 otherwise; and $C^{n_{\phi_i}}$ is the set of all possible bindings for the free variables in ϕ_i. Z is a normalized constant. The Markov network corresponds to this distribution, where nodes represent ground atoms and factors represent ground formulas.

4.2 Predicates

In the MLN, we design several predicates to resolve the ambiguities in phrase detection, mapping phrases to semantic items and semantic item grouping. Specifically, we design a hidden predicate *hasPhrase(i)* to indicate that the *i*-th candidate phrase has been chosen. The predicate *hasResource(i,j)* indicates that the *i*-th phrase is mapped to the *j*-th semantic item. The predicate *hasRelation(j,k,rr)* indicates that the *j*-th semantic item and the *k*-th semantic item should be grouped together with the argument-match-type *rr*. Note that we define four argument match types between two semantic items: *1_1*, *1_2*, *2_1* and *2_2*. Here, the argument match type *t_s* denotes that the *t*-th argument of the first semantic item corresponds to the *s*-th argument of the second semantic item[11]. The detailed illustration is shown in Table 1.

Type	Example	Question
1_1	*dbo:height 1_1 dbr:Michael_Jordan*	*How tall is Michael Jordan?*
1_2	*dbo:River 1_2 dbo:crosses*	*Which river does the Brooklyn Bridge cross?*
2_1	*dbo:creator 2_1 dbr:Walt_Disney*	*Which television shows were created by Walt Disney?*
2_2	*dbo:birthPlace 2_2 dbo:capital*	*Which actors were born in the capital of American?*

Table 1: Examples of the argument match types.

[10] This corresponds to the question "*How many people live in the capital of Australia?*"

[11] The *2*-nd argument is corresponding to the object argument of the relation, and the *1*-st argument is corresponding with the subject argument of the relation and the entity (including the class) itself.

Describing the attributes of phrases and relation between two phrases	
phraseIndex(p, i, j)	The start and end position of phrase *p* in question.
phrasePosTag(p, pt)	The POS tag of the head word in phrase *p*.
phraseDepTag(p, q, dt)	The dependency path tags between phrase *p* and *q*.
phraseDepOne (p, q)	If there is only one tag in the dependency path, the predicate is true.
hasMeanWord (p, q)	If there is any one meaning word in the dependency path of two phrases, the predicate is true.
Describing the attributes of semantic item and the mappings between phrases and semantic items	
resourceType(r, rt)	The type of semantic item r. Types of semantic items include *Entity*, *Class* and *Relation*
priorMatchScore(p, r, s)	The prior score of phrase *p* mapping to semantic item *r*.
Describing the attributes of relation between two semantic items in a knowledge base	
hasRelatedness(p, q, s)	The semantic coherence of semantic items.
isTypeCompatible(p, q, rr)	If the semantic items *p* are type-compatible with the semantic items *q*, the predicate is true.
hasQueryResult(s, p, o, rr1, rr2)	If the triple pattern consisting of semantic items *s*, *p*, *o* and argument-match-types *rr1* and *rr2* have query results, the predicate is true.

Table 2: Descriptions of observed predicates.

Moreover, we define a set of observed predicates to describe the properties of phrases, semantic items, relations between phrases and relations between semantic items. The observed predicates and descriptions are shown in Table 2.

Previous methods usually designed some heuristic patterns to group semantic items, which usually employed a human-designed syntactic path between two phrases to determine their relations. In contrast, we collect all the tokens in the dependency path between two phrases as possible patterns. The predicates *phraseDepTag* and *hasMeanWord* are designed to indicate the possible patterns. Note that if these tokens only contain POS tags *dt|in|wdt|to|cc|ex|pos|wp* or *stop words*, the value of the predicate *hasMeanWord* is false; otherwise, it is true. In this way, our system is expected to cover more question expressions. Moreover, the SPARQL endpoint is used to verify the type compatibility of two semantic items and if one triple pattern can obtain query results.

The predicate *hasRelatedness* needs to compute the coherence score between two semantic items. Following (Yahya et al., 2012), we use the Jaccard coefficient (Jaccard, 1908) based on the inlinks between two semantic items.

The predicate *priorMatchScore* assigns a prior score when mapping a phrase to a semantic item. We use different methods to compute this score according to different semantic item types. For entities, we use a normalized score based on the frequencies of a phrase referring to an entity. For classes and relations, we use different methods. We first define the following three similarity metrics: a) s_1: The Levenshtein distance score (Navarro, 2001) between the labels of the semantic item and the phrase; b) s_2: The word embedding (Mikolov et al., 2010) score, which measures the similarity between two phrases and is the maximum cosine value of the words' word embeddings between two phrases; and c) s_3: the instance overlap score, which is computed using the Jaccard coefficient of the instance overlap. All scores are normalized to produce a comparable scores in the interval of (0, 1). The final prior scores for mapping phrases to classes and relations are $\gamma s_1 + (1-\gamma)s_2$ and $\alpha s_1 + \beta s_2 + (1-\alpha-\beta)s_3$, respectively. The parameters are set to empirical values[12].

4.3 Formulas

According to these predicates, we design several first-order logic formulas for joint disambiguation. As mentioned in the first section, these formulas represent the meta patterns. The concrete patterns can be generated through these meta patterns with training data. Specifically, we use two types of formulas for the joint decisions: *Boolean* and *Weighted* formulas. *Boolean* formulas are hard constraints, which must be satisfied by all of the ground atoms in the final inference results. *Weighted* formulas are soft constraints, which can be violated with some penalties.

4.3.1 Boolean Formulas (Hard Constraints)

Table 3 lists the Boolean formulas used in this work. The "_" notation in the formulas indicates an arbitrary constant. The "$|f|$" notation expresses the number of true grounded atoms in the formula f. These formulas express the following constraints:

hf1: If a phrase is chosen, then it must have a mapped semantic item;

hf2: If a semantic item is chosen, then its mapped phrase must be chosen;

hf3: A phrase can be mapped to at most one semantic item;

hf4: If the phrase is not chosen, then its mapped

[12]Set γ to 0.6 for Class and set α and β to 0.3 and 0.3 for Relation, respectively.

hf1	$hasPhrase(p) \Rightarrow hasResource(p, \_)$
hf2	$hasResource(p, \_) \Rightarrow hasPhrase(p)$
hf3	$\lvert hasResource(p, \_)\rvert \leq 1$
hf4	$!hasPhrase(p) \Rightarrow !hasResource(p, r)$
hf5	$hasResource(\_, r) \Rightarrow hasRelation(r, \_, \_) \vee hasRelation(\_, r, \_)$
hf6	$\lvert hasRelation(r1, r2, \_)\rvert \leq 1$
hf7	$hasRelation(r1, r2, \_) \Rightarrow hasResource(\_, r1) \wedge hasResource(\_, r2)$
hf8	$phraseIndex(p1, s1, e1) \wedge phraseIndex(p2, s2, e2) \wedge overlap(s1, e1, s2, e2) \wedge hasPhrase(p1) \Rightarrow !hasPhrase(p2)$
hf9	$resourceType(r, \text{“}Entity\text{”}) \Rightarrow !hasRelation(r, \_, \text{“}2\_1\text{”}) \wedge !hasRelation(r, \_, \text{“}2\_2\text{”})$
hf10	$resourceType(r, \text{“}Entity\text{”}) \Rightarrow !hasRelation(\_, r, \text{“}2\_1\text{”}) \wedge !hasRelation(r, \_, \text{“}2\_2\text{”})$
hf11	$resourceType(r, \text{“}Class\text{”}) \Rightarrow !hasRelation(r, \_, \text{“}2\_1\text{”}) \wedge !hasRelation(r, \_, \text{“}2\_2\text{”})$
hf12	$resourceType(r, \text{“}Class\text{”}) \Rightarrow !hasRelation(\_, r, \text{“}2\_1\text{”}) \wedge !hasRelation(r, \_, \text{“}2\_2\text{”})$
hf13	$!isTypeCompatible(r1, r2, rr) \Rightarrow !hasRelation(r1, r2, rr)$

Table 3: Descriptions of Boolean formulas.

sf1	$priorMatchScore(p, r, s) \Rightarrow hasPhrase(p)$
sf2	$priorMatchScore(p, r, s) \Rightarrow hasResource(p)$
sf3	$phrasePosTag(p, pt+) \wedge resourceType(r, rt+) \Rightarrow hasResource(p, r)$
sf4	$phraseDepTag(p1, p2, dp+) \wedge hasResource(p1, r1) \wedge hasResource(p2, r2) \Rightarrow hasRelation(r1, r2, rr+)$
sf5	$phraseDepTag(p1, p2, dp+) \wedge hasResource(p1, r1) \wedge hasResource(p2, r2) \wedge !hasMeanWord(p1, p2) \Rightarrow hasRelation(r1, r2, rr+)$
sf6	$phraseDepTag(p1, p2, dp+) \wedge hasResource(p1, r1) \wedge hasResource(p2, r2) \wedge phraseDepOne(p1, p2) \Rightarrow hasRelation(r1, r2, rr+)$
sf7	$hasRelatedness(r1, r2, s) \wedge hasResource(\_, r1) \wedge hasResource(\_, r2) \Rightarrow hasRelation(r1, r2, \_)$
sf8	$hasQueryResult(r1, r2, r3, rr1, rr2) \Rightarrow hasRelation(r1, r2, rr1) \wedge hasRelation(r2, r3, rr2)$

Table 4: Descriptions of weighted formulas.

semantic item should not be chosen;

hf5: If a semantic item is chosen, then it should have at least one argument match relation with other semantic items;

hf6: Two semantic items have at most one argument match relation;

hf7: If an argument match relation for two semantic items is chosen, then they must be chosen;

hf8: Each of two chosen phrases must not overlap;

hf9, ***hf10***, ***hf11***, ***hf12***: The semantic item with type *Entity* and *Class* should not have a second argument that matches with others;

hf13: The chosen argument match relation for two sematic items must be type compatible.

4.3.2 Weighted Formulas (Soft Constraints)

Table 4 lists the weighted formulas used in this work. The "+" notation in the formulas indicates that each constant of the logic variable should be weighted separately. Those formulas express the following properties in joint decisions:

sf1, ***sf2***: The larger the score of the phrase mapping to a semantic item, the more likely the corresponding phrase and semantic item should been chosen;

sf3: There are some associations between the POS tags of phase and the types of mapped semantic items;

sf4, ***sf5***, ***sf6***: There are some associations between the dependency tags in the dependency pattern path of two phases and the types of argument match relations of two mapped semantic items;

sh7: The larger the relatedness of two semantic items, the more likely they have an argument match relation;

sf8: If the triple pattern has query results, these semantic items should have corresponding argument match relations.

5 Experiments

5.1 Dataset & Evaluation Metrics

We use the following three collections of questions from the QALD[13] task for question answering over linked data: QALD-1, QALD-3 and QALD-4. The generated SPARQL queries are evaluated on Linked Data from DBpedia and YAGO using a Virtuoso engine[14]. A typical example question from the QALD benchmark is *"Which books written by Kerouac were published by Viking Press?"*. As mentioned in Section 2.2, our system is not designed to answer questions that contain numbers, date comparisons and aggregation operations such as *group by* or *order by*. Therefore, we remove these types of questions and retain 110 questions from the QALD-4 training set for generating the specific formulas and for training their weights in MLN. We test our system using 37, 75 and 26 questions from the training set of QALD-1[15], and the testing set of QALD-3 and QALD-4 respectively. We use #T, #Q and #A to indicate the total

[13] www.sc.cit-ec.uni-bielefeld.de/qald/

[14] https://github.com/openlink/virtuoso-opensource

[15] We use the training set because we try to make a fair comparison with (Yahya et al., 2012).

number of questions in the testing set, the number of questions we could address and the number of questions answered correct, respectively. We select Precision ($P = \frac{\#A}{\#Q}$), Recall ($R = \frac{\#A}{\#T}$), and F1-score ($F1 = \frac{2 \cdot P \cdot R}{P+R}$) as the evaluation metrics. To assess the effectiveness of the disambiguation process in the MLN, we computed the overall quality measures by precision and recall with the manually obtained results.

5.2 Experimental Configurations

The Stanford dependency parser (De Marneffe et al., 2006) is used for extracting features from the dependency parse trees. We use the toolkit *thebeast*[16] to learn the weights of the formulas and to perform the MAP inference. The inference algorithm uses a cutting plane approach. In addition, for the parameter learning, we set all initial weights to zero and use an online learning algorithm with MIRA update rules to update the weights of the formulas. The number of iterations for the training and testing are set to 10 and 200, respectively.

5.3 Results and Discussion

5.3.1 The Effect of Joint Learning

To demonstrate the advantages of our joint learning, we design a pipeline system for comparison, which independently performs phrase detection, phrase mapping, and semantic item grouping by removing the unrelated formulas in MLN. For example, the formulas[17] related to the predicates *hasResource* and *hasRelation* are removed when detecting phrases in questions.

Table 5 shows the results, where **Joint** denotes the proposed method with joint inference and **Pipeline** denotes the compared method performing each step independently. We perform a comparison with the question answering results of QALD (QA), and comparisons at each of the following steps: PD (phrase detection), PM (phrase mapping) and MG (mapped semantic items grouping). From the results, we observe that our method answers over half of the questions. Moreover, our joint model based on MLN can obtain better performance in question answering compared to the pipeline system. We also observe that Joint exhibits better performance than Pipeline in most steps, except for MG in QALD-3. We believe this is because the three tasks (phrase detection, phrase mapping, and semantic item grouping) are connected with each other. Each step can provide useful information for the other two tasks. Therefore, performing joint inference can effectively improve the performance. Finally, we observe that the former task usually produces better results than the subsequent tasks (phrase detection exhibits a better performance than phrase mapping, and phrase mapping exhibits a better performance than semantic item grouping). The main reason is that the latter subtask is more complex than the former task. The decisions of the latter subtask strongly rely on the former results even though they have interacted effects.

5.3.2 The Effect of Pattern Learning

Table 6 shows a comparison of our system with *DEANNA* (Yahya et al., 2012), which is based on a joint disambiguation model but which employs hand-written patterns in its system. Because *DEANNA* only reports its results of the QALD-1 dataset, we do not show the results for QALD-3 and QALD-4 for equity. From the results, we can see that our system solved more questions and exhibited a better performance than did *DEANNA*. One of the greatest strengths of our system is that the learning system can address more questions than hand-written pattern rules.

System	#T	#Q	#A	P	R	F1
DEANNA (Yahya et al., 2012)	50	27	13	0.48	0.26	0.33
Ours	50	37	20	0.54	0.4	0.46

Table 6: Comparisons with *DEANNA* using the QALD-1 test questions.

Compared to the ILP (Integer Linear Programming) used in (Yahya et al., 2012) for joint disambiguation, we argue that there are two major differences to our method. 1) Our method is a data-driven approach that can learn effective patterns or rules for the task. Therefore, it exhibits more robustness and adaptability for various *KBs*. 2) We design several meta rules in MLN as opposed to specific ones. The specific rules can be generated by these meta rules based on the training data. By contrast, the traditional approach using ILP needs to set specific rules in advance, which requires more intensive labor than our approach.

To further illustrate the effectiveness of our pattern-learning strategy, we show the weights of the learned patterns corresponding to formula *sf3* in the MLN, as shown in Table 7. From the table,

[16] http://code.google.com/p/thebeast

[17] including entire formulas, excluding *hf8* and *sf1*

Benchmark	PD			PM			MG			QA					
	P	R	F1	P	R	F1	P	R	F1	#T	#Q	#A	P	R	F1
QALD-1(Joint)	0.93	0.981	**0.955**	0.895	0.944	**0.919**	0.703	0.813	**0.754**	50	37	20	0.54	0.4	**0.46**
QALD-1(Pipeine)	0.921	0.972	0.946	0.868	0.917	0.892	0.585	0.859	0.696	50	34	17	0.5	0.34	0.41
QALD-3(Joint)	0.941	0.941	**0.941**	0.878	0.918	**0.898**	0.636	0.798	0.708	99	75	45	0.6	0.46	**0.52**
QALD-3(Pipeline)	0.912	0.912	0.912	0.829	0.867	0.848	0.677	0.789	**0.729**	99	75	42	0.56	0.42	0.48
QALD-4(Joint)	0.947	0.978	**0.963**	0.937	0.967	**0.952**	0.776	0.865	**0.817**	50	26	15	0.58	0.3	**0.4**
QALD-4(Pipeline)	0.937	0.967	0.952	0.905	0.935	0.920	0.683	0.827	0.748	50	24	13	0.54	0.26	0.35

Table 5: The performance of joint learning on three benchmark datasets.

we can see that *nn*[18] is more likely mapped to *Entity*[19] than to *Class* and *Relation*, and *vb* is most likely mapped to *Relation*. This proves that our model can learn effective and reasonable patterns for QALD.

POS tag of Phrase	type of mapped Item	Weight
nn	*Entity*	2.11
nn	*Class*	0.243
nn	*Relation*	0.335
vb	*Relation*	0.517
wp	*Class*	0.143
wr	*Class*	0.025

Table 7: Sample weights of formulas, corresponding with formula *sf3*.

5.3.3 Comparison to the state of the art

To illustrate the effectiveness of the proposed method, we perform comparisons to the state-of-the-art methods. Table 8 shows the results using QALD-3 and QALD-4. These systems are the participants in the QALD evaluation campaigns. From the results, we can see that our system outperforms most systems at a competitive performance. They further prove the effectiveness of the proposed method.

Test set	System	#T	#Q	#A	P	R	F1
QALD-3	CASIA (He et al., 2013)	99	52	29	0.56	0.3	0.38
	Scalewelis (Joris and Ferré, 2013)	99	70	32	0.46	0.32	0.38
	RTV (Cristina et al., 2013)	99	55	30	0.55	0.3	0.39
	Intui2 (Corina, 2013)	99	99	28	0.28	28	0.28
	SWIP (Pradel et al., 2013)	99	21	15	0.71	0.15	0.25
	Ours	99	75	45	0.6	0.46	0.52
QALD-4[20]	gAnswer	50	25	16	0.64	0.32	0.43
	Intui3	50	33	10	0.30	0.2	0.24
	ISOFT	50	50	10	0.2	0.2	0.2
	RO FII	50	50	6	0.12	0.12	0.12
	Ours	50	26	15	0.58	0.3	0.4

Table 8: Comparisons with state-of-the-art systems using the QALD benchmark.

[18]The POS tag of the head word in the phrase

[19]The type of semantic item

[20]Because the QALD-4 conference does not start until after submission, we have no citation for the state-of-the-art systems in QALD-4. The results can be found at http://greententacle.techfak.uni-bielefeld.de/ cunger/qald.

5.3.4 The Effect of Different Formulas

To determine which formulas are more useful for QALD, we evaluate the performance of the proposed method with different predicate sets. We subtract one weighted formula from the original sets at a time, except retaining the first two formulas *sf1* and *sf2* for basic inference. Because of space limitations, only the results using QALD-3 testing set are shown in Table 9.

From the results, we can observe that removing some formulas can boost the performance on some single tasks, but employing all formulas can produce the best performance. This illustrates that solely resolving the steps in QALD (phrase detection, phrase mapping, semantic items grouping) can obtain local results, and that making joint inference is necessary and useful.

6 Related Work

Our proposed method is related to two lines of work: *Question Answering over Knowledge bases* and *Markov Logic Networks*.

Question answering over knowledge bases has attracted a substantial amount of interest over a long period of time. The initial attempts included BaseBall (Green Jr et al., 1961) and Lunar (Woods, 1977). However, these systems were mostly limited to closed domains due to a lack of knowledge resources. With the rapid development of structured data, such as DBpedia, Freebase and Yago, the need for providing user-friendly interface to these data has become increasingly urgent. Keyword (Elbassuoni and Blanco, 2011) and semantic (Pound et al., 2010) searches are limited to their ability to specify the relations among the different keywords.

The open topic progress has also been pushed by the QALD evaluation campaigns (Walter et al., 2012). Lopez et al. (2011) gave a comprehensive survey in this research area. The authors developed the *PowerAqua* system (Lopez et al., 2006) to

Formulas	PD			PM			MG			Avg		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
All Formulas	**0.941**	**0.941**	**0.941**	0.878	**0.918**	**0.898**	0.636	0.798	0.708	0.839	**0.901**	**0.869**
-sf3	0.931	0.927	0.929	0.877	0.913	0.895	0.637	**0.816**	0.715	0.834	0.897	0.864
-sf4	0.926	0.917	0.922	0.852	0.883	0.867	0.63	0.763	0.69	0.824	0.87	0.846
-sf5	0.931	0.927	0.929	0.873	0.908	0.89	0.633	**0.816**	0.713	0.831	0.895	0.862
-sf6	0.922	0.922	0.922	0.844	0.883	0.863	**0.702**	0.746	**0.723**	**0.842**	0.868	0.855
-sf7	0.931	0.917	0.924	**0.881**	0.908	0.894	0.621	0.763	0.685	0.833	0.88	0.856
-sf8	0.927	0.927	0.927	0.868	0.908	0.888	0.639	0.807	0.713	0.83	0.893	0.861

Table 9: Performance comparisons of different weighted formulas evaluated using the QALD-3 question set.

answer questions on large, heterogeneous datasets. For questions containing quantifiers, comparatives or superlatives, Unger et al. (2012) translated *NL* to *FL* using several SPARQL templates and using a set of heuristic rules mapping phrases to semantic items. The system most similar to ours is *DEANNA* (Yahya et al., 2012). However, *DEANNA* extracts predicate-argument structures from the questions using three hand-written patterns. Our system jointly learns these mappings and extractions completely from scratch.

Recently, the Semantic Parsing (SP) community targeted this problem from limited domains (Tang and Mooney, 2001; Liang et al., 2013) to open domains (Cai and Yates, 2013; Berant et al., 2013). The methods in semantic parsing answer questions by first converting natural language utterances into meaningful representations (e.g., the lambda calculus) and subsequently executing the formal logical forms over *KBs*. Compared to deriving the complete logical representation, our method aims to parse a question into a limited logic form with the semantic item query, which we believe is more appropriate for answering factoid questions.

Markov Logic Networks have been widely used in NLP tasks. Huang (2012) applied MLN to compress sentences by formulating the task as a word/phrase deletion problem. Fahrni and Strube (2012) jointly disambiguated and clustered concepts using MLN. MLN has also been used in coreference resolution (Song et al., 2012). For the task of identifying subjective text segments and of extracting their corresponding explanations from product reviews, Zhang et al. (2013) modeled these segments with MLN. To discover logical knowledge for deep question answering, Liu (2012) used MLN to resolve the inconsistencies of multiple knowledge bases.

Meza-Ruiz and Riedel (2009) employed MLN for Semantic Role Labeling (SRL). They jointly performed the following tasks for a sentence: predicate identification, frame disambiguation, argument identification and argument classification. The semantic analysis of SRL solely rested on the lexical level, but our analysis focuses on the knowledge-base level and aims to obtain an executable query and to support natural language inference.

7 Conclusions and Future Work

For the task of QALD, we present a joint learning framework for phrase detection, phrase mapping and semantic item grouping. The novelty of our method lies in the fact that we perform joint inference and pattern learning for all subtasks in QALD using first-order logic. Our experimental results demonstrate the effectiveness of the proposed method.

In the future, we plan to address the following limitations that still exist in the current system: a) numerous hand-labeled data are required for training the MLN, and we could use a latent form of semantic item query graphs (Liang et al., 2013); b) more robust solutions can be developed to find the implicit relations in questions; c) our system can be scaled up to large-scale open-domain knowledge bases (Fader et al., 2013; Yao and Van Durme, 2014); and d) the learning system has the advantage of being easily adapted to new settings, and we plan to extend it to other domains and languages (Liang and Potts, 2014).

Acknowledgments

The authors are grateful to the anonymous reviewers for their constructive comments. This work was sponsored by the National Basic Research Program of China (No. 2014CB340503) and the National Natural Science Foundation of China (No. 61202329, 61272332), CCF-Tencent Open Fund. This work was also supported in part by Noahs Ark Lab of Huawei Tech. Ltm.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *ACL*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Christian Bizer, Tom Heath, and Tim Berners-Lee. 2009. Linked data-the story so far. *International journal on semantic web and information systems*, 5(3):1–22.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL*.

Unger Christina and Andr Freitas. 2014. Question answering over linked data: Challenges, approaches, trends. In *ESWC*.

Dima Corina. 2013. Intui2: A prototype system for question answering over linked data. In *Work. Multilingual Question Answering over Linked Data (QALD-3)*.

Giannone Cristina, Bellomaria Valentina, and Basili Roberto. 2013. A hmm-based approach to question answering against linked data. In *Work. Multilingual Question Answering over Linked Data (QALD-3)*.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

Shady Elbassuoni and Roi Blanco. 2011. Keyword search over rdf graphs. In *CIKM*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL*.

Angela Fahrni and Michael Strube. 2012. Jointly disambiguating and clustering concepts and entities with markov logic. In *COLING*.

Andre Freitas and Edward Curry. 2014. Natural language queries over heterogeneous linked data graphs: A distributional-compositional semantics approach. In *IUI*.

Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. 1961. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224. ACM.

Shizhu He, Shulin Liu, Yubo Chen, Guangyou Zhou, Kang Liu, and Jun Zhao. 2013. Casia@qald-3: A question answering system over linked data. In *Work. Multilingual Question Answering over Linked Data (QALD-3)*.

Minlie Huang, Xing Shi, Feng Jin, and Xiaoyan Zhu. 2012. Using first-order logic to compress sentences. In *AAAI*.

Paul. Jaccard. 1908. Nouvelles recherches sur la distribution florale. *Bulletin de la Sociète Vaudense des Sciences Naturelles*, 44:223–270.

Guyonvarc'H Joris and Sébastien Ferré. 2013. Scalewelis: a scalable query-based faceted search system on top of sparql endpoints. In *Work. Multilingual Question Answering over Linked Data (QALD-3)*.

Jens Lehmann, Tim Furche, Giovanni Grasso, Axel-Cyrille Ngonga Ngomo, Christian Schallhart, Andrew Sellers, Christina Unger, Lorenz Bühmann, Daniel Gerber, Konrad Höffner, et al. 2012. Deqa: deep web extraction for question answering. In *ISWC*.

Percy Liang and Christopher Potts. 2014. Bringing machine learning and compositional semantics together. *Annual Reviews of Linguistics (to appear)*.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Zhao Liu, Xipeng Qiu, Ling Cao, and Xuanjing Huang. 2012. Discovering logical knowledge for deep question answering. In *CIKM*.

Vanessa Lopez, Enrico Motta, and Victoria Uren. 2006. Poweraqua: Fishing the semantic web. In *The Semantic Web: research and applications*, pages 393–410. Springer.

Vanessa Lopez, Victoria Uren, Marta Sabou, and Enrico Motta. 2011. Is question answering fit for the semantic web?: a survey. *Semantic Web*, 2(2):125–155.

Ivan Meza-Ruiz and Sebastian Riedel. 2009. Jointly identifying predicates, arguments and senses using markov logic. In *NAACL*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: a taxonomy of relational patterns with semantic types. In *EMNLP*.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88.

Jeffrey Pound, Ihab F Ilyas, and Grant Weddell. 2010. Expressive and flexible access to web-extracted data: a keyword-based structured query language. In *SIGMOD*.

C Pradel, G Peyet, O Haemmerlé, and N Hernandez. 2013. Swip at qald-3: results, criticisms and lesson learned (working notes). In *Work. Multilingual Question Answering over Linked Data (QALD-3)*.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer. 2013. Question answering on interlinked data. In *WWW*.

Yang Song, Jing Jiang, Wayne Xin Zhao, Sujian Li, and Houfeng Wang. 2012. Joint learning for coreference resolution with markov logic. In *EMNLP*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*, pages 466–477.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over rdf data. In *WWW*.

Sebastian Walter, Christina Unger, Philipp Cimiano, and Daniel Bär. 2012. Evaluation of a layered approach to question answering over linked data. In *The Semantic Web–ISWC 2012*, pages 362–374. Springer.

William A Woods. 1977. Lunar rocks in natural english: Explorations in natural language question answering. In *Linguistic structures processing*, pages 521–569.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural language questions for the web of data. In *EMNLP*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *CIKM*.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *ACL*.

Qi Zhang, Jin Qian, Huan Chen, Jihua Kang, and Xuanjing Huang. 2013. Discourse level explanatory relation extraction from product reviews using first-order logic. In *ACL*.

Lei Zou, Ruizhe Huang, Haixun WangZou, Jeffrey Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over rdf — a graph data driven approach. In *SIGMOD*.

Knowledge Graph and Corpus Driven Segmentation and Answer Inference for Telegraphic Entity-seeking Queries

Mandar Joshi *
IBM Research
mandarj90@in.ibm.com

Uma Sawant
IIT Bombay, Yahoo Labs
uma@cse.iitb.ac.in

Soumen Chakrabarti
IIT Bombay
soumen@cse.iitb.ac.in

Abstract

Much recent work focuses on formal interpretation of natural question utterances, with the goal of executing the resulting structured queries on knowledge graphs (KGs) such as Freebase. Here we address two limitations of this approach when applied to open-domain, entity-oriented Web queries. First, Web queries are rarely well-formed questions. They are "telegraphic", with missing verbs, prepositions, clauses, case and phrase clues. Second, the KG is always incomplete, unable to directly answer many queries. We propose a novel technique to segment a telegraphic query and assign a coarse-grained purpose to each segment: a base entity e_1, a relation type r, a target entity type t_2, and contextual words s. The query seeks entity $e_2 \in t_2$ where $r(e_1, e_2)$ holds, further evidenced by schema-agnostic words s. Query segmentation is integrated with the KG and an unstructured corpus where mentions of entities have been linked to the KG. We do not trust the best or any specific query segmentation. Instead, evidence in favor of candidate e_2s are aggregated across several segmentations. Extensive experiments on the ClueWeb corpus and parts of Freebase as our KG, using over a thousand telegraphic queries adapted from TREC, INEX, and WebQuestions, show the efficacy of our approach. For one benchmark, MAP improves from 0.2–0.29 (competitive baselines) to 0.42 (our system). NDCG@10 improves from 0.29–0.36 to 0.54.

*Work done as Masters student at IIT Bombay

1 Introduction

A majority of Web queries mention an entity or type (Lin et al., 2012), as users increasingly explore the Web of objects using Web search. To better support entity-oriented queries, commercial Web search engines are rapidly building up large catalogs of types, entities and relations, popularly called a "knowledge graph" (KG) (Gallagher, 2012). Despite these advances, robust, Web-scale, open-domain, entity-oriented search faces many challenges. Here, we focus on two.

1.1 "Telegraphic" queries

First, the surface utterances of entity-oriented Web queries are dramatically different from TREC- or Watson-style factoid question answering (QA), where questions are grammatically well-formed. Web queries are usually "telegraphic": they are short, rarely use function words, punctuations or clausal structure, and use relatively flexible word orders. E.g., the natural utterance "on the bank of which river is the Hermitage Museum located" may be translated to the telegraphic Web query hermitage museum river bank. Even on well-formed question utterances, 50% of interpretation failures are contributed by parsing or structural matching failures (Kwiatkowski et al., 2013). Telegraphic utterances will generally be even more challenging.

Consequently, whereas TREC-QA/NLP-style research has focused on parsing and precise interpretation of a well-formed query sentence to a strongly structured (typically graph-oriented) query language (Kasneci et al., 2008; Pound et al., 2012; Yahya et al., 2012; Berant et al., 2013; Kwiatkowski et al., 2013), the Web search and information retrieval (IR) community has focused on telegraphic queries (Guo et al., 2009; Sarkas et al., 2010; Li et al., 2011; Pantel et al., 2012; Lin et al., 2012; Sawant and Chakrabarti, 2013). In terms of target schema richness, these efforts may

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1104–1114,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

appear more modest. The act of query 'interpretation' is mainly a segmentation of query tokens by *purpose*. In the example above, one may report segments "Hermitage Museum" (a located artifact or named entity), and "river bank" (the target type). This is reminiscent of record segmentation in information extraction (IE). Over well-formed utterances, IE baselines are quite competitive (Yao and Van Durme, 2014). But here, we are interested exclusively in telegraphic queries.

1.2 Incomplete knowledge graph

The second problem is that the KG is always work in progress (Pereira, 2013), and connections found within nodes of the KG, between the KG and the query, or the KG and unstructured text, are often incomplete or erroneous. E.g., Wikipedia is considered tiny, and Freebase rather small, compared to what is needed to answer all but the "head" queries. Google's Freebase annotations (Gabrilovich et al., 2013) on ClueWeb (ClueWeb09, 2009) number fewer than 15 per page to ensure precision. Fewer than 2% are to entities in Freebase but not in Wikipedia.

It may also be difficult to harness the KG for answering certain queries. E.g., answering the query fastest odi century batsman, the intent of which is to find the batsman holding the record for the fastest century in One Day International (ODI) cricket, may be too difficult for most KG-only systems, but may be answered quite effectively by a system that also utilizes evidence from unstructured text.

There is a clear need for a "pay-as-you-go" architecture that involves both the corpus and KG. A query easily served by a curated KG should give accurate results, but it is desirable to have a *graceful interpolation* supported by the corpus: e.g., if the relation $r(e_1, e_2)$ is not directly evidenced in the KG, but strongly hinted in the corpus, we still want to use this for ranking.

1.3 Our contributions

Here, we make progress beyond the above frontier of prior work in the following significant ways. We present a new architecture for structural interpretation of a telegraphic query into these segments (some may be empty):

- Mention/s $\widehat{e_1}$ of an entity e_1,
- Mention $\widehat{r}$ of a relation type r,
- Mention $\widehat{t_2}$ of a target type t_2, and
- Other contextual matching words s (sometimes called *selectors*),

with the simultaneous intent of finding and ranking entities $e_2 \in t_2$, such that $r(e_1, e_2)$ is likely to hold, evidenced near the matching words in unstructured text.

Given the short, telegraphic query utterances, we limit our scope to at most one relation mention, unlike the complex mapping of clauses in well-formed questions to twig and join style queries (e.g., "find an actor whose spouse was an Italian bookwriter"). On the other hand, we need to deal with the unhelpful input, as well as consolidate the KG with the corpus for ranking candidate e_2s. Despite the modest specification, our query template is quite expressive, covering a wide range of entity-oriented queries (Yih et al., 2014).

We present a novel discriminative graphical model to capture the entity ranking inference task, with query segmentation as a by-product. Extensive experiments with over a thousand entity-seeking telegraphic queries using the ClueWeb09 corpus and a subset of Freebase show that we can accurately predict the segmentation and intent of telegraphic relational queries, and simultaneously rank candidate responses with high accuracy. We also present evidence that the KG and corpus have synergistic salutary effects on accuracy.

§2 explores related work in more detail. §3 gives some examples fitting our query template, explains why interpreting some of them is nontrivial, and sets up notation. §4 presents our core technical contributions. §5 presents experiments. Data can be accessed at `http://bit.ly/Spva49` and `http://bit.ly/WSpxvr`.

2 Related work

The NLP/QA community has traditionally assumed that question utterances are grammatically well-formed, from which precise clause structure, ground constants, variables, and connective relations can be inferred via semantic parsing (Kasneci et al., 2008; Pound et al., 2012; Yahya et al., 2012; Berant et al., 2013; Kwiatkowski et al., 2013) and translated to lambda expressions (Liang, 2013) or SPARQL style queries (Kasneci et al., 2008), with elaborate schema knowledge. Such approaches are often correlated with the assumption that all usable knowledge has been curated into a KG. The query is first translated to a structured form and then "executed" on the KG. A

Telegraphic query	$\widehat{e_1}$	$\widehat{r}$	$\widehat{t_2}$	s
first african american nobel prize winner	nobel prize	winner	african american	first
	nobel prize	-	winner	first african american
	-	-	winner	first african american nobel prize
dave navarro first band	dave navarro	band	-	first
	dave navarro	band	band	first
merril lynch headquarters	merril lynch	headquarters	-	-
	merril lynch	-	headquarters	-
spanish poet died in civil war	spanish	died in	poet	civil war
	civil war	died	-	spanish poet
	spanish	in	poet	died civil war
first american in space	-	-	-	first american in space
	-	-	american	first, in space

Figure 1: Example queries and some potential segmentations.

large corpus may be used to build relation expression models (Yao and Van Durme, 2014), but not as supporting evidence for target entities.

In contrast, the Web and IR community generally assumes a free-form query that is often telegraphic (Guo et al., 2009; Sarkas et al., 2010; Li et al., 2011). Queries being far more noisy, the goal of structure discovery is more modest, and often takes the form of a segmentation of the query regarded as a token sequence, assigning a broad *purpose* (Pantel et al., 2012; Lin et al., 2012) to each segment, mapping them probabilistically to a relatively loose schema, and ranking responses in conjunction with segmentations (Sawant and Chakrabarti, 2013). To maintain quality in the face of noisy input, these approaches often additionally exploit clicks (Li et al., 2011) or a corpus that has been annotated with entity mentions (Cheng and Chang, 2010; Li et al., 2010). The corpus provides contextual snippets for queries where the KG fails, preventing the systems from falling off the "structure cliff" (Pereira, 2013).

Our work advances the capabilities of the latter class of approaches, bringing them closer to the depth of the former, while handling telegraphic queries and retaining the advantage of corpus evidence over and above the KG. Very recently, (Yao et al., 2014) have concluded that for current benchmarks, deep parsing and shallow information extraction give comparable interpretation accuracy. The very recent work of (Yih et al., 2014) is similar in spirit to ours, but they do not unify segmentation and answer inference, along with corpus evidence, like we do.

3 Notation and examples

We use e_1, r, t_2, e_2 to represent abstract nodes and edges (MIDs in case of Freebase) from the KG, and $\widehat{e_1}, \widehat{r}, \widehat{t_2}$ to represent their textual mentions or hints, if any, in the query. s is a set of uninterpreted textual tokens in the query that are used to match and collect corpus contexts that lend evidence to candidate entities.

Figure 1 shows some telegraphic queries with possible segmentation into the above parts. Consider another example: dave navarro first band. 'Band' is a hint for type `/music/musical_group`, so it comprises $\widehat{t_2}$. Dave Navarro is an entity, with mention words 'dave navarro' comprising $\widehat{e_1}$. $\widehat{r}$ is made up of 'band', and represents the relation `/music/group_member/membership`. Finally, the word first cannot be mapped to any simple KG artifact, so are relegated to s (which makes the corpus a critical part of answer inference). We use s and $\widehat{s}$ interchangeably.

Generally, there will be enough noise and uncertainty that the search system should try out several of the most promising segmentations as shown in Figure 1. The accuracy of any specific segmentation is expected to be low in such adversarial settings. Therefore, support for an answer entity is aggregated over several segmentations. The expectation is that by considering multiple interpretations, the system will choose the entity with best supporting evidence from corpus and knowledge base.

4 Our Approach

Telegraphic queries are usually short, so we enumerate query token spans (with some restrictions, similar to beam search) to propose segmentations (§4.1). Candidate response entities are lined up for each interpretation, and then scored in a global model along with query segmentations (§4.2). §4.3 describes how model parameters are trained.

1: **input:** query token sequence q
2: initialize segmentations $\mathcal{I} = \varnothing$
3: $\mathcal{E}_1$ = (entity, mention) pairs from linker
4: **for all** $(e_1, \widehat{e_1}) \in \mathcal{E}_1$ **do**
5: assign label E_1 to mention tokens $\widehat{e_1}$
6: **for all** contiguous span $v \subset q \setminus \widehat{e_1}$ **do**
7: label each word $w \in v$ as T_2R
8: label other words $w \in q \setminus \widehat{e_1} \setminus v$ as S
9: add segments (E_1, T_2R, S) to $\mathcal{I}$
10: **end for**
11: **end for**
12: **return** candidate segmentations $\mathcal{I}$

Figure 2: Generating candidate query segmentations.

4.1 Generating candidate query segmentations

Each query token can have four labels, E_1, T_2, R, S, corresponding to the mentions of the base entity, target type, connecting relation, and context words. We found that segments hinting at T_2 and R frequently overlapped (e.g., 'author' in the query zhivago author). In our implementation, we simplified to three labels, E_1, T_2R, S, where tokens labeled T_2R are involved with both t_2 and r, the proposed structured target type and connecting relation. Another reasonable assumption was that the base entity mention and type/relation mentions are contiguous token spans, whereas context words can be scattered in multiple segments.

Figure 2 shows how candidate segmentations are generated. For step 3, we use TagMe (Ferragina and Scaiella, 2010), an entity linker backed by an entity gazette derived from our KG.

4.2 Graphical model

Based on the previous discussion, we assume that an entity-seeking query q is a sequence of tokens $q_1, q_2, \ldots$, and this can be partitioned into different kinds of subsequences, corresponding to e_1, r, t_2 and s, and denoted by a structured (vector) labeling $z = z_1, z_2, \ldots$. Given sequences q and z, we can separate out (possibly empty) token segments $\widehat{e_1}(q,z)$, $\widehat{t_2}(q,z)$, $\widehat{r}(q,z)$, and $\widehat{s}(q,z)$.

A query segmentation z becomes plausible in conjunction with proposals for e_1, r, t_2 and e_2 from the KG. The probability $\Pr(z, e_1, r, t_2, e_2|q)$ is modeled as proportional to the product of several potentials (Koller and Friedman, 2009) in a graphical model. In subsequent subsections, we will present the design of specific potentials.

- $\Psi_R(q,z,r)$ denotes the compatibility between the relation hint segment $\widehat{r}(q,z)$ and a proposed relation type r in the KG (§4.2.1).
- $\Psi_{T_2}(q,z,t_2)$ denotes the compatibility between the type hint segment $\widehat{t_2}(q,z)$ and a proposed target entity type t_2 in the KG (§4.2.2).
- $\Psi_{E_1,R,E_2,S}(q,z,e_1,r,e_2)$ is a novel corpus-based evidence potential that measures how strongly e_1 and e_2 appear in corpus snippets in the proximity of words in $\widehat{s}(q,z)$, and apparently related by relation type r (§4.2.3).
- $\Psi_{E_1}(q,z,e_1)$ denotes the compatibility between the query segment $\widehat{e_1}(q,z)$ and entity e_1 that it purportedly mentions (§4.2.4).
- $\Psi_S(q,z)$ denotes selector compatibility. Selectors are a fallback label, so this is pinned arbitrarily to 1; other potentials are balanced against this base value.
- $\Psi_{E_1,R,E_2}(e_1,r,e_2)$ is A if the relation $r(e_1,e_2)$ exists in the KG, and is $B > 0$ otherwise, for tuned/learnt constants $A > B > 0$. Note that this is a soft constraint ($B > 0$); if the KG is incomplete, the corpus may be able to supplement the required information.
- $\Psi_{E_2,T_2}(e_2,t_2)$ is 1 if e_2 belongs to t_2 and zero otherwise. In other words, candidate e_2s must be proposed to be instances of the proposed t_2 — this is a hard constraint, but can be softened if desired, like Ψ_{E_1,R,E_2}.

Figure 3 shows the relevant variable states as circled nodes, and the potentials as square factor nodes. To rank candidate entities e_2, we pin the node E_2 to each entity in turn. With E_2 pinned, we perform a MAP inference over all other hidden variables and note the score of e_2 as the product of the above potentials maximized over choices of all other variables: score$(e_2) =$

$$\begin{aligned}\max_{z,t_2,r,e_1} & \Psi_{T_2}(q,z,t_2)\Psi_R(q,z,r) \\ & \Psi_{E_1}(q,z,e_1)\Psi_S(q,z) \\ & \Psi_{E_2,T_2}(e_2,t_2)\Psi_{E_1,R,E_2}(e_1,r,e_2) \\ & \Psi_{E_1,R,E_2,S}(q,z,e_1,r,e_2). \qquad (1)\end{aligned}$$

We rank candidate e_2s by decreasing score, which is estimated by max-product message-passing (Koller and Friedman, 2009).

As noted earlier, any of the relation/type, or query entity partitions may be empty. To handle

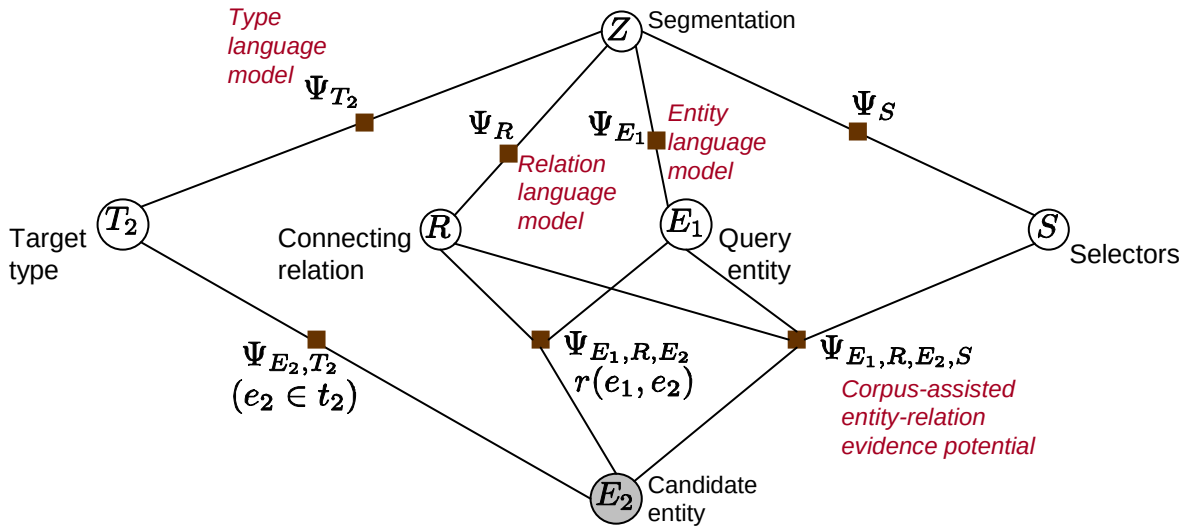


Figure 3: Graphical model for query segmentation and entity scoring. Factors/potentials are shown as squares. A candidate e_2 is observed and scored using equation (1). Query q is also observed but not shown to reduce clutter; most potentials depend on it.

this case, we allow each of the entity, relation or target type nodes in the graphical to take the value $\perp$ or 'null'. To support this, the value of the factor between the query segmentation node Z and $\Psi_{E1}(q, z, e_1)$, $\Psi_{T2}(q, z, t_2)$, and $\Psi_R(q, z, r)$) are set to suitable low values.

Next, we will describe the detailed design of some of the key potentials introduced above.

4.2.1 Relation language model for Ψ_R

Potential $\Psi_R(q, z, r)$ captures the compatibility between $\widehat{r}(q, z)$ and the proposed relation r. E.g., if the query is steve jobs death reason, and $\widehat{r}$ is (correctly chosen as) death reason, then the correct candidate r is `/people/deceased_person/cause_of_death`. An incorrect r is `/people/deceased_person/place_of_death`. An incorrect z may lead to $\widehat{r}(q, z)$ being jobs death.

Using corpus: Considerable variation may exist in how r is represented textually in a query. The relation language model needs to build a bridge between the formal r and the textual $\widehat{r}$, so that (un)likely r's have (small) large potential. Many approaches (Berant et al., 2013; Berant and Liang, 2014; Kwiatkowski et al., 2013; Yih et al., 2014) to this problem have been intensely studied recently. Given our need to process billions of Web pages efficiently, we chose a pattern-based approach (Nakashole et al., 2012): with each r, discover the most strongly associated phrase patterns from a reference corpus, then mark these patterns into much larger payload corpus.

We started with the 2000 (out of approximately 14000) most frequent relation types in Freebase, and the ClueWeb09 corpus annotated with Freebase entities (Gabrilovich et al., 2013). For each triple instance of each relation type, we located all corpus sentences that mentioned both participating entities. We made the crude assumption that if $r(e_1, e_2)$ holds and e_1, e_2 co-occur in a sentence then this sentence is evidence of the relationship. Each such sentence is parsed to obtain a dependency graph using the Malt Parser (Hall et al., 2014). Words in the path connecting the entities are joined together and added to a candidate phrase dictionary, provided the path is at most three hops. (Inspection suggested that longer dependency paths mostly arise out of noisy sentences or botched parses.) 30% of the sentences were thus retained. Finally, we defined

$$\Psi_R(q, z, r) = \frac{n(r, \widehat{r}(q, z))}{\sum_{p'} n(r, p')}, \qquad (2)$$

where p' ranges over all phrases that are known to hint at r, and $n(r, p)$ denotes the number of sentences where the phrase p occurred in the dependency path between the entities participating in relation r.

Assuming entity co-occurrence implies evidence is admittedly simplistic. However, the primary function of the relation model is to retrieve top-k relations that are compatible with the type/s

of e_1 and the given relation hint. Moreover, the remaining noise is further mitigated by the collective scoring in the graphical model. While we may miss relations if they are expressed in the query through obscure hints, allowing the relation to be $\perp$ acts as a safety net.

Using Freebase relation names: As mentioned earlier, queries may express relations differently as compared to the corpus. A relation model based solely on corpus annotations may not be able to bridge that gap effectively, particularly so, because of sparsity of corpus annotations or the rarity of Freebase triples in ClueWeb. E.g., for the Freebase relation `/people/person/profession`, we found very few annotated sentences. One way to address this problem is to utilize relation type names in Freebase to map hints to relation types. Thus, in addition to the corpus-derived relation model, we also built a language model that used Freebase relation type names as lemmas. E.g., the word 'profession' would contribute to the relation type `/people/person/profession`.

Our relation models are admittedly simple. This is mainly because telegraphic queries may express relations very differently from natural language text. As it is difficult to ensure precision of query interpretation stage, our models are geared towards recall. The system generates a large number of interpretations and relies on signals from the corpus and KG to bring forth correct interpretations.

4.2.2 Type language model for Ψ_{T_2}

Similar to the relation language model, we need a type language model to measure compatibility between t_2 and $\widehat{t_2}(q, z)$. Estimating the target entity type, without over-generalizing or over-specifying it, has always been important for QA. E.g., when $\widehat{t_2}$ is 'city', a good type language model should prefer t_2 as `/location/citytown` over `/location/location` while avoiding `/location/es_autonomous_city`.

A catalog like Freebase suggests a straightforward method to collect a type language model. Each type is described by one or more phrases through the link `/common/topic/alias`. We can collect these into a micro-'document' and use a standard Dirichlet-smoothed language model from IR (Zhai, 2008). In Freebase, an entity node (e.g., Einstein, `/m/0jcx`) may be linked to a type node (e.g. `/base/scientist/physicist`) using an edge with label `/type/object/type`.

But relation types provide additional clues to types of the endpoint entities. Freebase relation types have the form `/x/y/z`, where `x` is the domain of the relation, and `y` and `z` are string representations of the type of the entities participating in the relation. E.g., the (directed) relation type `/location/country/capital` connects from from `/location/country` to `/location/citytown`. Therefore, "capital" can be added to the set of descriptive phrases of entity type `/location/citytown`.

It is important to note that while we use Freebase link nomenclature for relation and type language models, our models are not incompatible with other catalogs. Indeed, most catalogs have established ways of deriving language models that describe their various structures. For example, most YAGO types are derived from WordNet synsets with associated phrasal descriptions (lemmas). YAGO relations also have readable names such as actedIn, isMarriedTo, etc. which can be used to estimate language models. DBPedia relations are mostly derived from (meaningfully) named attributes taken from infoboxes, hence they can be used directly. Furthermore, others (Wu and Weld, 2007) have shown how to associate language models with such relations.

4.2.3 Snippet scoring

The factor $\Psi_{E_1,R,E_2,S}(q, z, e_1, r, e_2)$ should be large if many snippets contain a mention of e_1 and e_2, relation r, and many high-signal words from s. Recall that we begin with a corpus annotated with entity mentions. Our corpus is not directly annotated with relation mentions. Therefore, we get from relations to documents via high-confidence phrases. Snippets are retrieved using a combined entity + word index, and scored for a given e_1, r, e_2, and selectors $\widehat{s}(q, z)$.

Given that relation phrases may be noisy and that their occurrence in the snippet may not necessarily mean that the given relation is being expressed, we need a scoring function that is cognizant of the roles of relation phrases and entities occurring in the snippets. In a basic version, $e_1, p, e_2, \widehat{s}$ are used to probe a combined entity+word index to collect high scoring snippets, with the score being adapted from BM25. The second, refined scoring function used a RankSVM-

style (Joachims, 2002) optimization.

$$\min_{\lambda,\xi} \|\lambda\|^2 + C\sum_{e^+,e^-} \xi_{e^+,e^-} \quad \text{s.t.}$$
$$\forall e^+, e^- : \quad \lambda \cdot f(q, D_{e^+}, e^+) + \xi_{e^+,e^-} \geq \lambda \cdot f(q, D_{e^-}, e^-) + 1. \tag{3}$$

where e^+ and e^- are positive and negative entities for the query q and $f(q, D_e, e)$ represents the feature map for the set of snippets D_e belonging to entity e. The assumption here is that all snippets containing e^+ are "positive" snippets for the query. f consolidates various signals like the number of snippets where e occurs near query entity e_1 and a relation phrase, or the number of snippets with high proportion of query IDF, hinting that e is a positive entity for the given query. A partial list of features used for snippet scoring is given in Figure 4.

Number of snippets with distance(e_2, $\widehat{e_1}$) $< k_1$ (k_1 = 5, 10)
Number of snippets with distance(e_2, relation phrase) $< k_2$ (k_2 = 3, 6)
Number of snippets with relation $r = \perp$
Number of snippets with relation phrases as prepositions
Number of snippets covering fraction of query IDF $> k_3$ (k_3 = 0.2, 0.4, 0.6, 0.8)

Figure 4: Sample features used for learning weights λ to score snippets.

4.2.4 Query entity model

Potential $\Psi_{E_1}(q, z, e_1)$ captures the compatibility between $\widehat{e_1}(q, z)$ (i.e., the words that mention e_1) and the claimed entity e_1 mentioned in the query. We used the TagMe entity linker (Ferragina and Scaiella, 2010) for annotating entities in queries. TagMe annotates the query with Wikipedia entities, which we map to Freebase, and use the annotation confidence scores as the potential $\Psi_{E_1}(q, z, e_1)$.

4.3 Discriminative parameter training with latent variables

We first set the potentials in (1) as explained in §4.2 (henceforth called 'Unoptimized'), and got encouraging accuracy. Then we rewrote each potential as

$$\Psi_\bullet(\cdots) = \exp\big(w_\bullet \cdot \phi_\bullet(\cdots)\big) \tag{4}$$
$$\text{or} \quad \log \textstyle\prod_\bullet \Psi_\bullet(\cdots) = \sum_\bullet w_\bullet \cdot \phi_\bullet(\cdots),$$

with $w_\bullet$ being a weight vector for a specific potential $\bullet$, and $\phi_\bullet$ being a corresponding feature vector. During inference, we seek to maximize

$$\max_{q,z,e_1,t_2,r} w \cdot \phi(q, z, e_1, t_2, r, e_2), \tag{5}$$

for a fixed w, to find the score of each candidate entity e_2. Here all $w_\bullet$ and $\phi_\bullet$ have been collected into unified weight and feature vectors w, ϕ. During training of w, we are given pairs of correct and incorrect answer entities e_2^+, e_2^-, and we wish to satisfy constraints of the form

$$\max_{q,z,e_1,t_2,r} w \cdot \phi(q, z, e_1, t_2, r, e_2^+) + \xi \geq 1 + \max_{q,z,e_1,t_2,r} w \cdot \phi(q, z, e_1, t_2, r, e_2^-), \tag{6}$$

because collecting e_2^+, e_2^- pairs is less work than supervising with values of z, e_1, t_2, r, e_2 for each query. Similar distant supervision problems were posed via bundle method by (Bergeron et al., 2008), and (Yu and Joachims, 2009), who used CCCP (Yuille and Rangarajan, 2006). These are equivalent in our setting. We use the CCCP style, and augment the objective with an additional entropy term as in (Sawant and Chakrabarti, 2013). We call this LVDT (latent variable discriminative training) in §5.

5 Experiments

5.1 Testbed

Corpus and knowledge graph: We used the ClueWeb09B (ClueWeb09, 2009) corpus containing 50 million Web documents. This corpus was annotated by Google with Freebase entities (Gabrilovich et al., 2013). The average page contains 15 entity annotations from Freebase. We used the Freebase KG and its links to Wikipedia.

Queries: We report on two sets of entity-seeking queries. A sample of about 800 well-formed queries from WebQuestions (Berant et al., 2013) were converted to telegraphic utterances (such as would be typed into commercial search engines) by volunteers familiar with Web search. We call this WQT (WebQuestions, telegraphic). Queries are accompanied by ground truth entities. The second data set, TREC-INEX, from (Sawant and Chakrabarti, 2013) has about 700 queries sampled from TREC and INEX, available at `http://bit.ly/WSpxvr`. These come with well-formed and telegraphic utterances, as well as ground truth entities.

There are some notable differences between these query sets. For WQT, queries were generated by using Google's query suggestions interface. Volunteers were asked to find answers using single Freebase pages. Therefore, by construction, queries retained can be answered using the Freebase KG alone, with a simple $r(e_1, ?)$ form. In contrast, TREC-INEX queries provide a balanced mix of t_2 and r hints in the queries, and direct answers from triples is relatively less available.

5.2 Implementation details

On an average, the pseudocode in Figure 2 generated 13 segmentations per query, with longer queries generating more segmentations than shorter ones.

We used an MG4J (Boldi and Vigna, 2005) based query processor, written in Java, over entity and word indices on ClueWeb09B. The index supplies snippets with a specified maximum width, containing a mention of some entity and satisfying a WAND (Broder et al., 2003) predicate over words in $\widehat{s}$. In case of phrases in the query, the WAND threshold was computed by adding the IDF of constituent words. The index returned about 330,000 snippets on average for WAND threshold of 0.6.

We retained the top 200 candidate entities from the corpus; increasing this horizon did not give benefits. We also considered as candidates for e_2 those entities that are adjacent to e_1 in the KG via top-scoring r candidates. In order to generate supporting snippets for an interpretation containing entity annotation e, we need to match e with Google's corpus annotations. However, relying solely on corpus annotations fails to retrieve many potential evidence snippets, because entity annotations are sparse. Therefore we probed the token index with the textual mention of e_1 in the query; this improved recall.

We also investigated the feasibility of our proposals for interactive search. There are three major processes involved in answering a query - generating potential interpretations, collecting/scoring snippets, and inference (MAP for Unoptimized and $w\phi(\cdot)$ for LVDT). For the WQT dataset, average time per query for each stage was approximately - 0.2, 16.6 and 1.3 seconds respectively. Our (Java) code did not optimize the bottleneck at all; only 10 hosts and no clever load balancing were used. We believe commercial search engines can cut this down to less than a second.

5.3 Research questions

In the rest of this section we will address these questions:

- For telegraphic queries, is our entity-relation-type-selector segmentation better than the type-selector segmentation of (Sawant and Chakrabarti, 2013)?
- When semantic parsers (Berant et al., 2013; Kwiatkowski et al., 2013) are subjected to telegraphic queries, how do they perform compared to our proposal?
- Are the KG and corpus really complementary as regards their support of accurate ranking of candidate entities?
- Is the prediction of r and t_2 from our approach better than a greedy assignment based on local language models?

We also discuss anecdotes of successes and failures of various systems.

5.4 Benefits of relation in addition to type

Figure 5 shows entity-ranking MAP, MRR, and NDCG@10 (n@10) for two data sets and various systems. "No interpretation" is an IR baseline without any KG. Type+selector is our implementation of (Sawant and Chakrabarti, 2013). Unoptimized and LVDT both beat "no interpretation" and "type+selector" by wide margins. (Boldface implies best performing formulation.) There are two notable differences between S&C and our work. First, S&C do not use the knowledge graph (KG) and rely on a noisy corpus. This means S&C fails to answer queries whose answers are found only in KG. This can be seen from WQT results; they perform only slightly better than the baseline. Second, even for queries that can be answered through the corpus alone, S&C miss out on two important signals that the query may provide - namely the query entity and the relation. Our framework not only provides a way to use a curated and high precision knowledge graph but also attempts to provide more reachability to corpus by the use of relational phrases.

In case of TREC-INEX, LVDT improves upon the unoptimized graphical model, where for WQT, it does not. Preliminary inspection suggests this is because WQT has noisy and incomplete ground truth, and LVDT trains to the noise; a non-convex

Dataset	Formulation	map	mrr	n@10
TREC -INEX	No interpretation	.205	.215	.292
	Type+selector	.292	.306	.356
	Unoptimized	.409	.419	.502
	LVDT	**.419**	**.436**	**.541**
WQT	No interpretation	.080	.095	.131
	Type+selector	.116	.152	.201
	Unoptimized	**.377**	**.401**	**.474**
	LVDT	.295	.323	.406

Figure 5: 'Entity-relation-type-selector' segmentation yields better accuracy than 'type-selector' segmentation.

objective makes matters worse. The bias in our unoptimized model circumvents training noise.

5.5 Comparison with semantic parsers

For TREC-INEX, both unoptimized and LVDT beat SEMPRE (Berant et al., 2013) convincingly, whether it is trained with Free917 or WebQuestions (Figure 6).

SEMPRE's relatively poor performance, in this case, is explained by its complete reliance on the knowledge graph. As discussed previously, the TREC-INEX dataset contains a sizable proportion of queries that may be difficult to answer using a KG alone. When SEMPRE is compared with our systems with a telegraphic sample of WebQuestions (WQT), results are mixed. Our Unoptimized model still compares favorably to SEMPRE, but with slimmer gains. As before, LVDT falls behind.

Dataset	Formulation	map	mrr	n@10
TREC -INEX	SEMPRE(Free917)	.154	.159	.186
	SEMPRE(WQ)	.197	.208	.247
	Unoptimized	.409	.419	.502
	LVDT	**.419**	**.436**	**.541**
WQT	SEMPRE(Free917)	.229	.255	.285
	SEMPRE(WQ)	.374	**.406**	.449
	Unoptimized	**.377**	.401	**.474**
	Jacana	.239	.256	.329
	LVDT	.295	.323	.406

Figure 6: Comparison with semantic parsers.

Our smaller gains over SEMPRE in case of WebQuestions is explained by how WebQuestions was assembled (Berant et al., 2013). Although Google's query suggestions gave an eclectic pool, only those queries survived that could be answered using a single Freebase page, which effectively reduced the role of a corpus. In fact, a large fraction of WQT queries cannot be answered well using the corpus alone, because FACC1 annotations are too sparse and rarely cover common nouns and phrases such as 'democracy' or 'drug overdose' which are needed for some WQT queries.

For WQT, our system also compares favorably with Jacana (Yao and Van Durme, 2014). Given that they subject their input to natural langauge parsing, their relatively poor performance is not unsurprsing.

5.6 Complementary benefits of KG & corpus

Figure 7 shows the synergy between the corpus and the KG. In all cases and for all metrics, using the corpus and KG together gives superior performance to using any of them alone. However, it is instructive that in case of TREC-INEX, corpus-only is better than KG-only, whereas this is reversed for WQT, which also supports the above argument.

Data	Formulation	map	mrr	n@10
TREC-INEX	Unoptimized (KG)	.201	.209	.241
	Unoptimized (Corpus)	.381	.388	.471
	Unoptimized (Both)	**.409**	**.419**	**.502**
	LVDT (KG only)	.255	.264	.293
	LVDT (Corpus)	.267	.272	.315
	LVDT (Both)	**.419**	**.436**	**.541**
WQT	Unoptimized (KG)	.329	.343	.394
	Unoptimized (Corpus)	.188	.228	.291
	Unoptimized (Both)	**.377**	**.401**	**.474**
	LVDT (KG only)	.257	.281	.345
	LVDT (Corpus only)	.170	.210	.280
	LVDT (Both)	**.295**	**.323**	**.406**

Figure 7: Synergy between KB and corpus.

5.7 Collective vs. greedy segmentation

To judge the quality of interpretations, we asked paid volunteers to annotate queries with an appropriate relation and type, and compared them with the interpretations associated with top-ranked entities. Results in Figure 8 indicate that in spite of noisy relation and type language models, our formulations produce high quality interpretations through collective inference.

Figure 9 demonstrates the benefit of collective inference over greedy segmentation followed by

Formulation	Type	Relation	Type/Rel
Unoptimized (top 1)	23	49	60
Unoptimized (top 5)	29	57	68
LVDT (top 1)	25	52	61
LVDT (top 5)	33	61	69

Figure 8: Fraction of queries (%) with correct interpretations of t_2, r, and t_2 or r, on TREC-INEX.

evaluation. Collective inference boosts absolute MAP by as much as 0.2.

Dataset	Formulation	map	mrr	n@10
TREC -INEX	Unoptimized (greedy)	.343	.347	.432
	Unoptimized	.409	.419	.502
	LVDT (greedy)	205	.214	.259
	LVDT	**.419**	.436	.541
WQT	Unoptimized (greedy)	.246	.271	.335
	Unoptimized	.377	.401	.474
	LVDT (greedy)	.212	.246	.317
	LVDT	.295	.323	.406

Figure 9: Collective vs. greedy segmentation

5.8 Discussion

Closer scrutiny revealed that collective inference often overcame errors in earlier stages to produce a correct ranking over answer entities. E.g., for the query automobile company makes spider the entity disambiguation stage fails to identify the car Alfa Romeo Spider (`/m/08ys39`). However, the interpretation stage recovers from the error and segments the query with Automobile (`/m/0k4j`) as the query entity e_1, `/organization/organization` and `/business/industry/companies` as target type t_2 and relation r respectively (from the relation/type hint 'company'), and *spider* as selector to arrive at the correct answer Alfa Romeo (`/m/09c50`). The corpus features also play a crucial role for queries which may not be accurately represented with an appropriate logical formula. For the query meg ryan bookstore movie, the textual patterns for the relation `ActedIn` in conjunction with the selector word 'bookstore' correctly identifies the answer entity You've Got Mail (`/m/014zwb`).

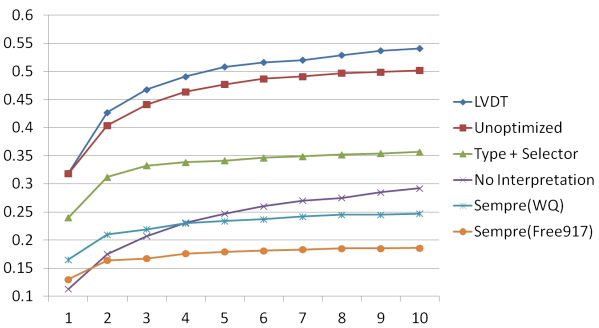


Figure 10: Comparison of various approaches for NDCG at rank 1 to 10, TREC-INEX dataset

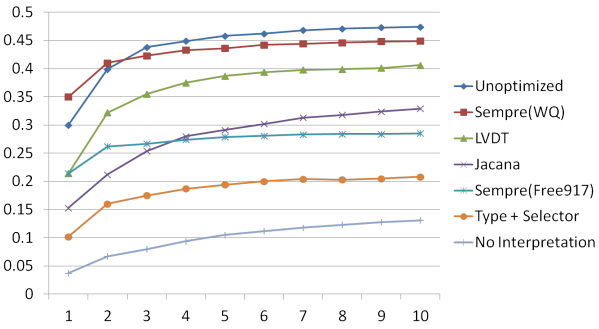


Figure 11: Comparison of various approaches for NDCG at rank 1 to 10, WQT dataset

We also analyzed samples of queries where our system did not perform particularly well. We observed that one of the recurring themes of these queries was that their answer entities had very little corpus support, and the type/relation hint mapped to too many or no candidate type/relations. For example, in the query south africa political system, the relevant type/relation hint 'political system' could not be mapped to `/government/form_of_government` and `/location/country/form_of_government` respectively.

6 Conclusion and future work

We presented a technique to partition telegraphic entity-seeking queries into functional segments and to rank answer entities accordingly. While our results are favorable compared to strong prior art, further improvements may result from relaxing our model to recognize multiple e_1s and rs. It may also help to deploy more sophisticated paraphrasing models (Berant and Liang, 2014) or word embeddings (Yih et al., 2014) for relation hints. It would also be interesting to supplement entity-linked corpora and curated KGs with extracted triples (Fader et al., 2014). Another possibility is to apply the ideas presented here to well-formed questions.

References

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *ACL Conference*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Charles Bergeron, Jed Zaretzki, Curt Breneman, and Kristin P. Bennett. 2008. Multiple instance ranking. In *ICML*, pages 48–55. ACM.

Paolo Boldi and Sebastiano Vigna. 2005. MG4J at TREC 2005. In Ellen M. Voorhees and Lori P. Buckland, editors, *TREC*, number SP 500-266 in Special Publications. NIST.

Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. 2003. Efficient query evaluation using a two-level retrieval process. In *CIKM*, pages 426–434. ACM.

Tao Cheng and Kevin Chen-Chuan Chang. 2010. Beyond pages: supporting efficient, scalable entity search with dual-inversion index. In *EDBT*. ACM.

ClueWeb09. 2009. `http://www.lemurproject.org/clueweb09.php/`.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *SIGKDD Conference*.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). *CoRR/arXiv*, abs/1006.3498. `http://arxiv.org/abs/1006.3498`.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora. `http://lemurproject.org/clueweb12/`, June. Version 1 (Release date 2013-06-26, Format version 1, Correction level 0).

Sean Gallagher. 2012. How Google and Microsoft taught search to 'understand' the Web. ArsTechnica article. `http://goo.gl/NWs0zT`.

Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named entity recognition in query. In *SIGIR Conference*, pages 267–274. ACM.

Johan Hall, Jens Nilsson, and Joakim Nivre. 2014. Maltparser. `http://www.maltparser.org/`.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *SIGKDD Conference*, pages 133–142. ACM.

Gjergji Kasneci, Fabian M. Suchanek, Georgiana Ifrim, Maya Ramanath, and Gerhard Weikum. 2008. NAGA: Searching and ranking knowledge. In *ICDE*. IEEE.

Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP Conference*, pages 1545–1556.

Xiaonan Li, Chengkai Li, and Cong Yu. 2010. EntityEngine: Answering entity-relationship queries using shallow semantics. In *CIKM*, October. (demo).

Yanen Li, Bo-Jun Paul Hsu, ChengXiang Zhai, and Kuansan Wang. 2011. Unsupervised query segmentation using clickthrough for information retrieval. In *SIGIR Conference*, pages 285–294. ACM.

Percy Liang. 2013. Lambda dependency-based compositional semantics. Technical Report arXiv:1309.4408, Stanford University. `http://arxiv.org/abs/1309.4408`.

Thomas Lin, Patrick Pantel, Michael Gamon, Anitha Kannan, and Ariel Fuxman. 2012. Active objects: Actions for entity-centric search. In *WWW Conference*, pages 589–598. ACM.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: A taxonomy of relational patterns with semantic types. In *EMNLP Conference*, EMNLP-CoNLL '12, pages 1135–1145. ACL.

Patrick Pantel, Thomas Lin, and Michael Gamon. 2012. Mining entity types from query logs via user intent modeling. In *ACL Conference*, pages 563–571, Jeju Island, Korea, July.

Fernando Pereira. 2013. Meaning in the wild. Invited talk at EMNLP Conference. `http://hum.csse.unimelb.edu.au/emnlp2013/invited-talks.html`.

Jeffrey Pound, Alexander K. Hudek, Ihab F. Ilyas, and Grant Weddell. 2012. Interpreting keyword queries over Web knowledge bases. In *CIKM*.

Nikos Sarkas, Stelios Paparizos, and Panayiotis Tsaparas. 2010. Structured annotations of Web queries. In *SIGMOD Conference*.

Uma Sawant and Soumen Chakrabarti. 2013. Learning joint query interpretation and response ranking. In *WWW Conference*, Brazil.

Fei Wu and Daniel S Weld. 2007. Automatically semantifying Wikipedia. In *CIKM*, pages 41–50.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural language questions for the Web of data. In *EMNLP Conference*, pages 379–390, Jeju Island, Korea, July.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with Freebase. In *ACL Conference*. ACL.

Xuchen Yao, Jonathan Berant, and Benjamin Van Durme. 2014. Freebase QA: Information extraction or semantic parsing? In *ACL 2014 Workshop on Semantic Parsing (SP14)*.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *ACL Conference*. ACL.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *ICML*, pages 1169–1176. ACM.

A. L. Yuille and Anand Rangarajan. 2006. The concave-convex procedure. *Neural Computation*, 15(4):915–936.

ChengXiang Zhai. 2008. Statistical language models for information retrieval: A critical review. *Foundations and Trends in Information Retrieval*, 2(3):137–213, March.

A Regularized Competition Model for Question Difficulty Estimation in Community Question Answering Services

Quan Wang[†] **Jing Liu**[‡] **Bin Wang**[†] **Li Guo**[†]
[†]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, P. R. China
{wangquan,wangbin,guoli}@iie.ac.cn
[‡]Harbin Institute of Technology, Harbin, P. R. China
jliu@ir.hit.edu.cn

Abstract

Estimating questions' difficulty levels is an important task in community question answering (CQA) services. Previous studies propose to solve this problem based on the question-user comparisons extracted from the question answering threads. However, they suffer from data sparseness problem as each question only gets a limited number of comparisons. Moreover, they cannot handle newly posted questions which get no comparisons. In this paper, we propose a novel question difficulty estimation approach called Regularized Competition Model (RCM), which naturally combines question-user comparisons and questions' textual descriptions into a unified framework. By incorporating textual information, RCM can effectively deal with data sparseness problem. We further employ a K-Nearest Neighbor approach to estimate difficulty levels of newly posted questions, again by leveraging textual similarities. Experiments on two publicly available data sets show that for both well-resolved and newly-posted questions, RCM performs the estimation task significantly better than existing methods, demonstrating the advantage of incorporating textual information. More interestingly, we observe that RCM might provide an automatic way to quantitatively measure the knowledge levels of words.

1 Introduction

Recent years have seen rapid growth in community question answering (CQA) services. They have been widely used in various scenarios, including general information seeking on the web[1], knowledge exchange in professional communities[2], and question answering in massive open online courses (MOOCs)[3], to name a few.

An important research problem in CQA is how to automatically estimate the difficulty levels of questions, i.e., *question difficulty estimation* (QDE). QDE can benefit many applications. Examples include 1) Question routing. Routing questions to appropriate answerers can help obtain quick and high-quality answers (Li and King, 2010; Zhou et al., 2009). Ackerman and McDonald (1996) have demonstrated that routing questions by matching question difficulty level with answerer expertise level will make better use of answerers' time and expertise. This is even more important for enterprise question answering and MOOCs question answering, where human resources are expensive. 2) Incentive mechanism design. Nam et al. (2009) have found that winning point awards offered by reputation systems is a driving factor for user participation in CQA services. Assigning higher point awards to more difficult questions will significantly improve user participation and satisfaction. 3) Linguistics analysis. Researchers in computational linguistics are always interested in investigating the correlation between language and knowledge, to see how the language reflects one's knowledge (Church, 2011). As we will show in Section 5.4, QDE provides an automatic way to quantitatively measure the knowledge levels of words.

Liu et al. (2013) have done the pioneer work on QDE, by leveraging question-user comparisons extracted from the question answering threads. Specifically, they assumed that the difficulty level of a question is higher than the expertise level of the asker (i.e. the user who asked the question), but lower than that of the best answerer (i.e. the user who provided the best answer). A TrueSkill al-

[1]http://answers.yahoo.com/

[2]http://stackoverflow.com/

[3]http://coursera.org/

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1115–1126,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

gorithm (Herbrich et al., 2006) was further adopted to estimate question difficulty levels as well as user expertise levels from the pairwise comparisons among them. To our knowledge, it is the only existing work on QDE. Yang et al. (2008) have proposed a similar idea, but their work focuses on a different task, i.e., estimating difficulty levels of tasks in crowdsourcing contest services.

There are two major drawbacks of previous methods: 1) *data sparseness problem* and 2) *cold-start problem*. By the former, we mean that under the framework of previous work, each question is compared only twice with the users (once with the asker and the other with the best answerer), which might not provide enough information and contaminate the estimation accuracy. By the latter, we mean that previous work only deals with well-resolved questions which have received the best answers, but cannot handle newly posted questions with no answers received. In many real-world applications such as question routing and incentive mechanism design, however, it is usually required that the difficulty level of a question is known instantly after it is posted.

To address the drawbacks, we propose further exploiting questions' textual descriptions (e.g., title, body, and tags) to perform QDE. Preliminary observations have shown that a question's difficulty level can be indicated by its textual description (Liu et al., 2013). We take advantage of the observations, and assume that if two questions are close in their textual descriptions, they will also be close in their difficulty levels, i.e., the *smoothness assumption*. We employ manifold regularization (Belkin et al., 2006) to characterize the assumption. Manifold regularization is a well-known technique to preserve local invariance in manifold learning algorithms, i.e., nearby points are likely to have similar embeddings (Belkin and Niyogi, 2001). Then, we propose a novel Regularized Competition Model (RCM), which formalizes QDE as minimizing a loss on question-user comparisons with manifold regularization on questions' textual descriptions. As the smoothness assumption offers extra information for inferring question difficulty levels, incorporating it will effectively deal with data sparsity. Finally, we adopt a K-Nearest Neighbor approach (Cover and Hart, 1967) to perform cold-start estimation, again by leveraging the smoothness assumption.

Experiments on two publicly available data sets collected from Stack Overflow show that 1) RCM performs significantly better than existing methods in the QDE task for both well-resolved and cold-start questions. 2) The performance of RCM is insensitive to the particular choice of the term weighting schema (determines how a question's textual description is represented) and the similarity measure (determines how the textual similarity between two questions is measured). The results demonstrate the advantage of incorporating textual information for QDE. Qualitative analysis further reveals that RCM might provide an automatic way to quantitatively measure the knowledge levels of words.

The main contributions of this paper include: 1) We take fully advantage of questions' textual descriptions to address data sparseness problem and cold-start problem which previous QDE methods suffer from. To our knowledge, it is the first time that textual information is introduced in QDE. 2) We propose a novel QDE method that naturally combines question-user comparisons and questions' textual descriptions into a unified framework. The proposed method performs QDE significantly better than existing methods. 3) We demonstrate the practicability of estimating difficulty levels of cold-start questions purely based on their textual descriptions, making various applications feasible in practice. As far as we know, it is the first work that considers cold-start estimation. 4) We explore how a word's knowledge level can be automatically measured by RCM.

The rest of the paper is structured as follows. Section 2 describes the problem formulation and the motivation of RCM. Section 3 presents the details of RCM. Section 4 discusses cold-start estimation. Section 5 reports experiments and results. Section 6 reviews related work. Section 7 concludes the paper and discusses future work.

2 Preliminaries

2.1 Problem Formulation

A CQA service provides a platform where people can ask questions and seek answers from others. Given a CQA portal, consider a specific category where questions on the same topic are asked and answered, e.g., the "C++ programming" category of Stack Overflow. When an asker u_a posts a question q in the category, there will be several answerers to answer the question. Among all the received answers, a best one will be chosen

by the asker or voted by the community. The answerer who provides the best answer is called the best answerer u_b. The other answerers are denoted by $O = \{u_{o_1}, u_{o_2}, \cdots, u_{o_M}\}$. A question answering thread (QA thread) is represented as a quadruplet (q, u_a, u_b, O). Collecting all such QA threads in the category, we get M users and N questions, denoted by $\mathcal{U} = \{u_1, u_2, \cdots, u_M\}$ and $\mathcal{Q} = \{q_1, q_2, \cdots, q_N\}$ respectively. Each user u_m is associated with an expertise score θ_m, representing his/her expertise level. A larger θ_m indicates a higher expertise level of the user. Each question q_n is associated with a difficulty score β_n, representing its difficulty level. A larger β_n indicates a higher difficulty level of the question. Difficulty scores (as well as expertise scores) are assumed to be comparable with each other in the specified category. Besides, each question q_n has a textual description, and is represented as a V-dimensional term vector $\boldsymbol{d}_n$, where V is the vocabulary size.

The *question difficulty estimation* (QDE) task aims to automatically learn the question difficulty scores (β_n's) by utilizing the QA threads $\mathcal{T} = \{(q, u_a, u_b, O) : q \in \mathcal{Q}\}$ as well as the question descriptions $\mathcal{D} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_N\}$ in the specified category. Note that in Section 2 and Section 3, we consider estimating difficulty scores of resolved questions, i.e., questions with the best answers selected or voted. Estimating difficulty scores of unresolved questions, e.g., newly posted ones, will be discussed in Section 4.

2.2 Competition-based Methods

Liu et al. (2013) have proposed a competition-based method for QDE. The key idea is to 1) extract pairwise competitions from the QA threads and 2) estimate question difficulty scores based on extracted competitions.

To extract pairwise competitions, it is assumed that question difficulty scores and user expertise scores are expressed on the same scale. Given a QA thread (q, u_a, u_b, O), it is further assumed that:

Assumption 1 (pairwise comparison assumption) *The difficulty score of question q is higher than the expertise score of the asker u_a, but lower than that of the best answerer u_b. Moreover, the expertise score of the best answerer u_b is higher than that of the asker u_a, as well as any answerer in O.*[4]

[4]The difficulty score of question q is not assumed to be lower than the expertise score of any answerer in O, since such a user may just happen to see the question and respond to it, rather than knowing the answer well.

Given the assumption, there are $(|O| + 3)$ pairwise competitions extracted from the QA thread, including 1) one competition between the question q and the asker u_a, 2) one competition between the question q and the best answerer u_b, 3) one competition between the best answerer u_b and the asker u_a, and 4) $|O|$ competitions between the best answerer u_b and each of the answerers in O. The question q is the winner of the first competition, and the best answerer u_b is the winner of the remaining $(|O| + 2)$ competitions. These pairwise competitions are denoted by

$$C_q = \{u_a \prec q, q \prec u_b, u_a \prec u_b, u_{o_1} \prec u_b, \cdots, u_{o_M} \prec u_b\},$$

where $i \prec j$ means that competitor j beats competitor i in a competition. Let

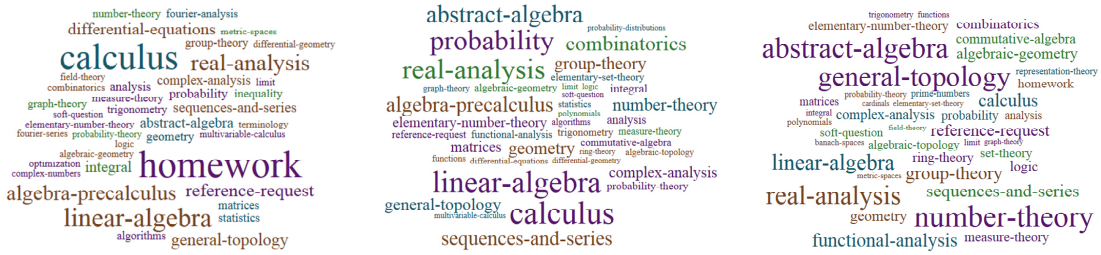
$$C = \bigcup_{q \in \mathcal{Q}} C_q \tag{1}$$

be the set containing all the pairwise competitions extracted from $\mathcal{T}$.

Given the competition set C, Liu et al. (2013) further adopted a TrueSkill algorithm (Herbrich et al., 2006) to learn the competitors' skill levels (i.e. the question difficulty scores and the user expertise scores). TrueSkill assumes that the practical skill level of each competitor follows a normal distribution $N\left(\mu, \sigma^2\right)$, where μ is the average skill level and σ is the estimation uncertainty. Then it updates the estimations in an online mode: for a newly observed competition with its win-loss result, 1) increase the average skill level of the winner, 2) decrease the average skill level of the loser, and 3) shrink the uncertainties of both competitors as more data has been observed. Yang et al. (2008) have proposed a similar competition-based method to estimate tasks' difficulty levels in crowdsourcing contest services, by leveraging PageRank (Page et al., 1999) algorithm.

2.3 Motivating Discussions

The methods introduced above estimate competitors' skill levels based solely on the pairwise competitions among them. The more competitions a competitor participates in, the more accurate the estimation will be. However, according to the pairwise comparison assumption (Assumption 1), each question participates in only two competitions, one with the asker and the other with the best answerer. Hence, there might be no enough information to accurately infer its difficulty score. We call this the *data sparseness problem*.



(a) Low difficulty. (b) Medium difficulty. (c) High difficulty.

Figure 1: Tag clouds of SO/Math questions with different difficulty levels.

Taking advantage of additional metadata has been demonstrated to be an effective way of dealing with data sparsity in various applications such as collaborative filtering (Claypool et al., 1999; Schein et al., 2002) and personalized search (Dou et al., 2007; Sugiyama et al., 2004). The rationale behind is to bridge the gap among users/items by leveraging their similarities based on the metadata. As for QDE, preliminary observations have shown that a question's difficulty level can be indicated by its textual description (Liu et al., 2013). As an example, consider the QA threads in the "mathematics" category of Stack Overflow. Divide the questions into three groups: 1) low difficulty, 2) medium difficulty, and 3) high difficulty, according to their difficulty scores estimated by TrueSkill. Figure 1 visualizes the frequency distribution of tags in each group, where the size of each tag is in proportion to its frequency in the group. The results indicate that the tags associated with the questions do have the ability to reflect the questions' difficulty levels, e.g., low difficulty questions usually have tags such as "homework" and "calculus", while high difficulty ones usually have tags such as "general topology" and "number theory". We further calculate the Pearson correlation coefficient (Rodgers and Nicewander, 1988) between 1) the gap between the averaged difficulty scores in each two groups and 2) the Euclidean distance between the aggregated textual descriptions in each two groups . The result is $r = 0.6424$, implying that the difficulty gap is positively correlated with the textual distance. In other words, the more similar two questions' textual descriptions are, the more close their difficulty levels are. Therefore, we take the textual information to bridge the difficulty gap among questions, by assuming that

Assumption 2 (smoothness assumption) *If two questions q_i and q_j are close in their textual descriptions $\boldsymbol{d}_i$ and $\boldsymbol{d}_j$, they will also be close in their difficulty scores β_i and β_j.*

The smoothness assumption brings us additional information about question difficulty scores by inferring textual similarities. It serves as a supplement to the pairwise competitions, and might help address the data sparseness problem which previous methods suffer from.

3 Modeling Text Similarities for QDE

This section presents a novel Regularized Competition Model (RCM) for QDE, which combines the pairwise competitions and the textual descriptions into a unified framework. RCM can alleviate the data sparseness problem and perform more accurate estimation.

3.1 Regularized Competition Model

We start with several notations. As question difficulty scores can be directly compared with user expertise scores, we take questions as pseudo users. Let $\bar{\boldsymbol{\theta}} \in \mathbb{R}^{M+N}$ denote the skill levels (i.e. the expertise scores and the difficulty scores) of all the (pseudo) users:

$$\bar{\theta}_i = \begin{cases} \theta_i, & 1 \le i \le M, \\ \beta_{i-M}, & M < i \le M+N, \end{cases}$$

where $\bar{\theta}_i$ is the i-th entry of $\bar{\boldsymbol{\theta}}$. The first M entries are the user expertise scores, denoted by $\bar{\boldsymbol{\theta}}_u \in \mathbb{R}^M$. The last N entries are the question difficulty scores, denoted by $\bar{\boldsymbol{\theta}}_q \in \mathbb{R}^N$. Let $\bar{\theta}_i^{(u)}$ and $\bar{\theta}_i^{(q)}$ denote the i-th entries of $\bar{\boldsymbol{\theta}}_u$ and $\bar{\boldsymbol{\theta}}_q$ respectively.

Exploiting Pairwise Competitions. We define a loss on each pairwise competition $i \prec j$:

$$\ell\left(\bar{\theta}_i, \bar{\theta}_j\right) = \max\left(0, \delta - \left(\bar{\theta}_j - \bar{\theta}_i\right)\right)^p, \qquad (2)$$

where p is either 1 or 2. The loss is defined on the skill gap between the two competitors, i.e., $\bar{\theta}_j - \bar{\theta}_i$,

measuring the inconsistency between the expected outcome and the actual outcome. If the gap is larger than a predefined threshold δ, competitor j would probably beat competitor i in the competition, which coincides with the actual outcome. Then the loss will be zero. Otherwise, there is a higher chance that competitor j loses the competition, which goes against the actual outcome. Then the loss will be greater than zero. The smaller the gap is, the higher the chance of inconsistency becomes, and the greater the loss will be. Note that the threshold δ can take any positive value since we do not pose a norm constraint on $\bar{\boldsymbol{\theta}}$.[5] Without loss of generality we take $\delta = 1$ throughout this paper. As we will show in Section 3.2, the loss defined in Eq. (2) has some similarity with the SVM loss (Chapelle, 2007). We name it hinge loss when $p = 1$, and quadratic loss when $p = 2$.

Given the competition set C, estimating skill levels of (pseudo) users then amounts to solving the following optimization problem:

$$\min_{\bar{\boldsymbol{\theta}}} \sum_{(i \prec j) \in C} \ell\left(\bar{\theta}_i, \bar{\theta}_j\right) + \frac{\lambda_1}{2} \bar{\boldsymbol{\theta}}^T \bar{\boldsymbol{\theta}}, \tag{3}$$

where the first term is the empirical loss measuring the total inconsistency; the second term is a regularizer to prevent overfitting; and $\lambda_1 \geq 0$ is a trade-off coefficient. It is also a competition-based QDE method, called Competition Model (CM).

Exploiting Question Descriptions. Manifold regularization is a well-known technique used in manifold learning algorithms to preserve local invariance, i.e., nearby points are likely to have similar embeddings (Belkin and Niyogi, 2001). In QDE, the smoothness assumption expresses similar "invariance", i.e., nearby questions (in terms of textual similarities) are likely to have similar difficulty scores. Hence, we characterize the assumption with the following manifold regularizer:

$$\begin{aligned} \mathcal{R} &= \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\bar{\theta}_i^{(q)} - \bar{\theta}_j^{(q)}\right)^2 w_{ij} \\ &= \bar{\boldsymbol{\theta}}_q^T \mathbf{D} \bar{\boldsymbol{\theta}}_q - \bar{\boldsymbol{\theta}}_q^T \mathbf{W} \bar{\boldsymbol{\theta}}_q = \bar{\boldsymbol{\theta}}_q^T \mathbf{L} \bar{\boldsymbol{\theta}}_q, \end{aligned} \tag{4}$$

where w_{ij} is the textual similarity between question i and question j; $\mathbf{W} \in \mathbb{R}^{N \times N}$ is the similarity matrix with the (i, j)-th entry being w_{ij}; $\mathbf{D} \in \mathbb{R}^{N \times N}$ is a diagonal matrix with the i-th entry on the diagonal being $d_{ii} = \sum_{j=1}^{N} w_{ij}$; and $\mathbf{L} = \mathbf{D} - \mathbf{W} \in \mathbb{R}^{N \times N}$ is the graph Laplacian (Chung, 1997). Minimizing $\mathcal{R}$ results in the smoothness assumption: for any questions i and j, if their textual similarity w_{ij} is high, the difficulty gap $\left(\bar{\theta}_i^{(q)} - \bar{\theta}_j^{(q)}\right)^2$ will be small.

A Hybrid Method. Combining Eq. (3) and Eq. (4), we obtain RCM, which amounts to the following optimization problem:

$$\min_{\bar{\boldsymbol{\theta}}} \sum_{(i \prec j) \in C} \ell\left(\bar{\theta}_i, \bar{\theta}_j\right) + \frac{\lambda_1}{2} \bar{\boldsymbol{\theta}}^T \bar{\boldsymbol{\theta}} + \frac{\lambda_2}{2} \bar{\boldsymbol{\theta}}_q^T \mathbf{L} \bar{\boldsymbol{\theta}}_q. \tag{5}$$

Here $\lambda_2 \geq 0$ is also a trade-off coefficient. The advantages of RCM include 1) It naturally formalizes QDE as minimizing a manifold regularized loss function, which seamlessly integrates both the pairwise competitions and the textual descriptions. 2) By incorporating textual information, it can address the data sparseness problem which previous methods suffer from, and perform significantly better in the QDE task.

3.2 Learning Algorithm

Redefine the k-th pairwise competition (assumed to be carried out between competitors i and j) as $(\boldsymbol{x}_k, y_k)$. $\boldsymbol{x}_k \in \mathbb{R}^{M+N}$ indicates the competitors:

$$x_i^{(k)} = 1, x_j^{(k)} = -1, \text{and } x_l^{(k)} = 0 \text{ for any } l \neq i, j,$$

where $x_l^{(k)}$ is the l-th entry of $\boldsymbol{x}_k$. $y_k \in \{1, -1\}$ is the outcome: if competitor i beats competitor j, $y_k = 1$; otherwise, $y_k = -1$. The objective in Eq. (5) can then be rewritten as

$$\mathcal{L}\left(\bar{\boldsymbol{\theta}}\right) = \sum_{k=1}^{|C|} \max\left(0, 1 - y_k \left(\bar{\boldsymbol{\theta}}^T \boldsymbol{x}_k\right)\right)^p + \frac{1}{2} \bar{\boldsymbol{\theta}}^T \mathbf{Z} \bar{\boldsymbol{\theta}},$$

where $\mathbf{z} = \begin{pmatrix} \lambda_1 \mathbf{I}_M & \mathbf{0} \\ \mathbf{0} & \lambda_1 \mathbf{I}_N + \lambda_2 \mathbf{L} \end{pmatrix}$ is a block matrix; $\mathbf{I}_M \in \mathbb{R}^{M \times M}$ and $\mathbf{I}_N \in \mathbb{R}^{N \times N}$ are identity matrices; $p = 1$ corresponds to the hinge loss, and $p = 2$ the quadratic loss. It is clear that the loss defined in Eq. (2) has the same format as the SVM loss.

The objective $\mathcal{L}$ is differentiable for the quadratic loss but non-differentiable for the hinge loss. We employ a subgradient method (Boyd et al., 2003) to solve the optimization problem. The algorithm starts at a point $\bar{\boldsymbol{\theta}}_0$ and, as many iterations as needed, moves from $\bar{\boldsymbol{\theta}}_t$ to $\bar{\boldsymbol{\theta}}_{t+1}$ in the direction of the negative subgradient:

$$\bar{\boldsymbol{\theta}}_{t+1} = \bar{\boldsymbol{\theta}}_t - \gamma_t \nabla \mathcal{L}\left(\bar{\boldsymbol{\theta}}_t\right),$$

[5] Given any $\bar{\theta}_i$, $\bar{\theta}_j$, and δ, there always exists a linear transformation which keeps the sign of $\left(\delta - \left(\bar{\theta}_j - \bar{\theta}_i\right)\right)$ unchanged.

Algorithm 1 Regularized Competition Model

Require: competition set C and description set $\mathcal{D}$
1: $\bar{\boldsymbol{\theta}}_0 \leftarrow \mathbf{1}$
2: **for** $t = 0 : T-1$ **do**
3: $\quad \mathcal{K}_t \leftarrow \left\{k : 1 - y_k\left(\bar{\boldsymbol{\theta}}_t^T \boldsymbol{x}_k\right) > 0\right\}$
4: $\quad \nabla\mathcal{L}\left(\bar{\boldsymbol{\theta}}_t\right) \leftarrow$ calculated by Eq. (6)
5: $\quad \bar{\boldsymbol{\theta}}_{t+1} \leftarrow \bar{\boldsymbol{\theta}}_t - \gamma_t \nabla\mathcal{L}\left(\bar{\boldsymbol{\theta}}_t\right)$
6: $\quad \boldsymbol{\Theta}_{t+1} \leftarrow \left\{\bar{\boldsymbol{\theta}}_0, \bar{\boldsymbol{\theta}}_1, \cdots, \bar{\boldsymbol{\theta}}_{t+1}\right\}$
7: $\quad \bar{\boldsymbol{\theta}}_{t+1} \leftarrow \arg\min_{\bar{\boldsymbol{\theta}} \in \boldsymbol{\Theta}_{t+1}} \mathcal{L}\left(\bar{\boldsymbol{\theta}}\right)$
8: **end for**
9: **return** $\bar{\boldsymbol{\theta}}_T$

where $\gamma_t > 0$ is the learning rate. The subgradient is calculated as

$$\nabla\mathcal{L}\left(\bar{\boldsymbol{\theta}}_t\right) = \begin{cases} \mathbf{Z}\bar{\boldsymbol{\theta}}_t - \sum\limits_{k\in\mathcal{K}_t} y_k \boldsymbol{x}_k, & p=1, \\ \mathbf{Z}\bar{\boldsymbol{\theta}}_t + 2\sum\limits_{k\in\mathcal{K}_t} \boldsymbol{x}_k \boldsymbol{x}_k^T \bar{\boldsymbol{\theta}}_t - 2\sum\limits_{k\in\mathcal{K}_t} y_k \boldsymbol{x}_k, & p=2, \end{cases} \quad (6)$$

where $\mathcal{K}_t = \left\{k : 1 - y_k\left(\bar{\boldsymbol{\theta}}_t^T \boldsymbol{x}_k\right) > 0\right\}$. As it is not always a descent method, we keep track of the best point found so far (Boyd et al., 2003):

$$\bar{\boldsymbol{\theta}}_{t+1} = \arg\min_{\bar{\boldsymbol{\theta}} \in \boldsymbol{\Theta}_{t+1}} \mathcal{L}\left(\bar{\boldsymbol{\theta}}\right),$$

where $\boldsymbol{\Theta}_{t+1} = \left\{\bar{\boldsymbol{\theta}}_0, \bar{\boldsymbol{\theta}}_1, \cdots, \bar{\boldsymbol{\theta}}_{t+1}\right\}$. The whole procedure is summarized in Algorithm 1.

Convergence. For constant learning rate (i.e., $\gamma_t = \gamma$), Algorithm 1 is guaranteed to converge to within some range of the optimal value, i.e.,

$$\lim_{t\to\infty} \mathcal{L}\left(\bar{\boldsymbol{\theta}}_t\right) - \mathcal{L}^* < \epsilon,$$

where $\mathcal{L}^*$ denotes the minimum of $\mathcal{L}(\cdot)$, and ϵ is a constant defined by the learning rate γ. For more details, please refer to (Boyd et al., 2003). During our experiments, we set the iteration number as $T = 1000$ and the learning rate as $\gamma_t = 0.001$, and convergence was observed.

Complexity. For both the hinge loss and the quadratic loss, the time complexity (per iteration) and the space complexity of RCM are both $O\left(|C| + \eta N^2\right)$. Here, $|C|$ is the total number of competitions, M and N are the numbers of users and questions respectively, and η is the ratio of non-zero entries in the graph Laplacian $\mathbf{L}$.[6] In the analysis, we have assumed that $M \ll \eta N^2$ and $N \ll \eta N^2$.

[6]Owing to the sparse nature of questions' textual descriptions, the graph Laplacian $\mathbf{L}$ is usually sparse, with about 70% entries being zero according to our experiments.

4 Cold-Start Estimation

Previous sections discussed estimating difficulty scores of resolved questions, from which pairwise competitions could be extracted. However, for newly posted questions without any answers received, no competitions could be extracted and none of the above methods work. We call it the *cold-start problem*.

We heuristically apply a K-Nearest Neighbor (KNN) approach (Cover and Hart, 1967) to cold-start estimation, again by leveraging the smoothness assumption. The key idea is to propagate difficulty scores from well-resolved questions to cold-start ones according to their textual similarities. Specifically, suppose that there exists a set of well-resolved questions whose difficulty scores have already been estimated by a QDE method. Given a cold-start question q^*, we first pick K well-resolved questions that are closest to q^* in textual descriptions, referred to as the nearest neighbors. The difficulty score of question q^* is then predicted as the averaged difficulty scores of its nearest neighbors. The KNN method bridges the gap between cold-start and well-resolved questions by inferring their textual similarities, and might effectively deal with the cold-start problem.

5 Experiments

We have conducted experiments to test the effectiveness of RCM in estimating difficulty scores of both well-resolved and cold-start questions. Moreover, we have explored how a word's difficulty level can be quantitatively measured by RCM.

5.1 Experimental Settings

Data Sets. We obtained a publicly available data set of Stack Overflow between July 31, 2008 and August 1, 2012[7], containing QA threads in various categories. We considered the categories of "C++ programming" and "mathematics", and randomly sampled about 10,000 QA threads from each category, denoted by SO/CPP and SO/Math respectively. For each question, we took the title and body fields as its textual description. For both data sets, stop words in a standard list[8] and words whose total frequencies are less than 10 were removed. Table 1 gives the statistics of the data sets.

[7]http://blog.stackoverflow.com/category/cc-wiki-dump/

[8]http://jmlr.org/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

	# users	# questions	# competitions	# words
SO/CPP	14,884	10,164	50,043	2,208
SO/Math	6,564	10,528	40,396	2,009

Table 1: Statistics of the data sets.

For evaluation, we randomly sampled 600 question pairs from each data set, and asked annotators to compare the difficulty levels of the questions in each pair. We had two graduate students majoring in computer science annotate the SO/CPP questions, and two majoring in mathematics annotate the SO/Math questions. For each question, only the title, body, and tags were exposed to the annotators. Given a question pair (q_1, q_2), the annotators were asked to give one of the three labels: $q_1 \succ q_2$, $q_2 \succ q_1$, or $q_1 = q_2$, which respectively means that question q_1 has a higher, lower, or equal difficulty level compared with question q_2. We used Cohen's kappa coefficient (Cohen, 1960) to measure the inter-annotator agreement. The result is $\kappa = 0.7533$ on SO/CPP and $\kappa = 0.8017$ on SO/Math, indicating that the inter-annotator agreement is quite substantial on both data sets. After removing the question pairs with inconsistent labels, we got 521 annotated SO/CPP question pairs and 539 annotated SO/Math question pairs.

We further randomly split the annotated question pairs into development/test/cold-start sets, with the ratio of 2:2:1. The first two sets were used to evaluate the methods in estimating difficulty scores of resolved questions. Specifically, the development set was used for parameter tuning and the test set was used for evaluation. The last set was used to evaluate the methods in cold-start estimation, and the questions in this set were excluded from the learning process of RCM as well as any baseline method.

Baseline Methods. We considered three baseline methods: PageRank (PR), TrueSkill (TS), and CM, which are based solely on the pairwise competitions.

- PR first constructs a competitor graph, by creating an edge from competitor i to competitor j if j beats i in a competition. A PageRank algorithm (Page et al., 1999) is then utilized to estimate the relative importance of the nodes, i.e., question difficulty scores and user expertise scores. The damping factor was set from 0.1 to 0.9 in steps of 0.1.
- TS has been applied to QDE by Liu et al. (2013). We set the model parameters in the same way as they suggested.
- CM performs QDE by solving Eq. (3). We set λ_1 in $\{0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$.

We compared RCM with the above baseline methods. In RCM, both parameters λ_1 and λ_2 were set in $\{0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$.

Evaluation Metric. We employed accuracy (ACC) as the evaluation metric:

$$\text{ACC} = \frac{\text{\# correctly judged question pairs}}{\text{\# all question pairs}}.$$

A question pair is regarded as correctly judged if the relative difficulty ranking given by an estimation method is consistent with that given by the annotators. The higher the accuracy is, the better a method performs.

5.2 Estimation for Resolved Questions

The first experiment tested the methods in estimating difficulty scores of resolved questions.

Estimation Accuracies. We first compared the estimation accuracies of PR, TS, CM, and RCM on the test sets of SO/CPP and SO/Math, obtained with the best parameter settings determined by the development sets. Table 2 gives the results, where "H" denotes the hinge loss and "Q" the quadratic loss. In RCM, to calculate the graph Laplacian $\mathbf{L}$, we adopted Boolean term weighting schema and took Jaccard coefficient as the similarity measure. From the results, we can see that 1) RCM performs significantly better than the baseline methods on both data sets (t-test, p-value < 0.05), demonstrating the advantage of exploiting questions' textual descriptions for QDE. 2) The improvements of RCM over the baseline methods on SO/Math are greater than those on SO/CPP, indicating that the textual descriptions of the SO/Math questions are more powerful in reflecting their difficulty levels. The reason is that the SO/Math questions are much more heterogeneous, belonging to various subfields of mathematics. The difficulty gaps among different subfields are sometimes obvious (e.g., a question in topology in general has a higher difficulty level than a question in linear algebra), making the textual descriptions more powerful in distinguishing the difficulty levels.

Graph Laplacian Variants. We further investigated the performances of different term weighting schemas and similarity measures in the graph

	PR	TS	CM		RCM	
			H	Q	H	Q
SO/CPP	0.5876	0.6134	0.6340	0.6753	**0.7371**	0.7268
SO/Math	0.6067	0.6109	0.6527	0.6820	**0.7699**	**0.7699**

Table 2: ACC of different methods for well-resolved questions.

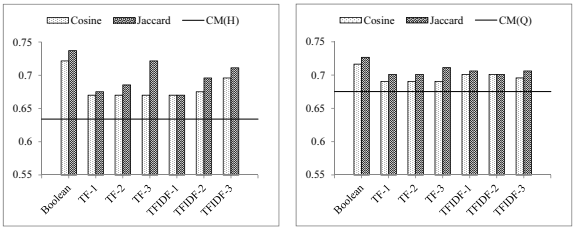
Notation	Definition
Boolean	$v(w,q) = \begin{cases} 1, & \text{if word } w \text{ occurs in question } q \\ 0, & \text{otherwise} \end{cases}$
TF-1	$v(w,q) = f(w,q)$, the number of occurrences
TF-2	$v(w,q) = \log(f(w,q)+1)$
TF-3	$v(w,q) = 0.5 + \frac{0.5 \times f(w,q)}{\max\{f(w,q) : w \in q\}}$
TFIDF-1	$v(w,q) = \text{TF-1} \times \log \frac{\lvert Q \rvert}{\lvert\{q \in Q : w \in q\}\rvert}$
TFIDF-2	$v(w,q) = \text{TF-2} \times \log \frac{\lvert Q \rvert}{\lvert\{q \in Q : w \in q\}\rvert}$
TFIDF-3	$v(w,q) = \text{TF-3} \times \log \frac{\lvert Q \rvert}{\lvert\{q \in Q : w \in q\}\rvert}$
Cosine	$\text{Sim}(\boldsymbol{d}_1, \boldsymbol{d}_2) = \frac{\boldsymbol{d}_1^T \boldsymbol{d}_2}{\lVert\boldsymbol{d}_1\rVert \times \lVert\boldsymbol{d}_2\rVert} \in [0,1]$
Jaccard	$\text{Sim}(\boldsymbol{d}_1, \boldsymbol{d}_2) = \frac{\boldsymbol{d}_1^T \boldsymbol{d}_2}{\lVert\boldsymbol{d}_1\rVert^2 + \lVert\boldsymbol{d}_2\rVert^2 - \lVert\boldsymbol{d}_1\rVert \times \lVert\boldsymbol{d}_2\rVert} \in [0,1]$

Table 3: Different term weighting schemas and similarity measures.

Laplacian. The term weighting schema determines how a question's textual description is represented. We explored a Boolean schema, three TF schemas, and three TFIDF schemas (Salton and Buckley, 1988). The similarity measure determines how the textual similarity between two questions is calculated. We explored the Cosine similarity and the Jaccard coefficient (Huang, 2008). Detailed descriptions are given in Table 3.

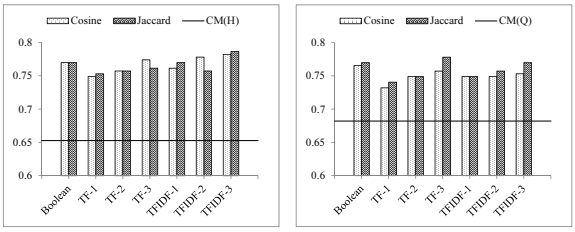
Figure 2 and Figure 3 show the estimation accuracies of the RCM variants on the test sets of SO/CPP and SO/Math respectively, again obtained with the best parameter settings determined by the development sets. The performance of CM is also given (the straight lines in the figures).[9] From the results, we can see that 1) All the RCM variants can improve over CM on both data sets, and most of the improvements are significant (t-test, p-value < 0.05). This further demonstrates that the effectiveness of incorporating textual descriptions is not affected by the particular choice of the term weighting schema or similarity measure. 2) Boolean term weighting schema performs the best, considering different similarity measures, loss types, and data sets collectively. 3) Jaccard

[9]CM performs better than PR and TS on both data sets.



(a) Hinge loss. (b) Quadratic loss.

Figure 2: ACC of RCM variants for well-resolved questions on SO/CPP.



(a) Hinge loss. (b) Quadratic loss.

Figure 3: ACC of RCM variants for well-resolved questions on SO/Math.

coefficient performs as well as Cosine similarity on SO/Math, but almost consistently better on SO/CPP. Throughout the experiments, we adopted Boolean term weighting schema and Jaccard coefficient to calculate the graph Laplacian.

5.3 Estimation for Cold-Start Questions

The second experiment tested the methods in estimating difficulty scores of cold-start questions. We employed Boolean term weighting schema to represent a cold-start question, and utilized Jaccard Coefficient to select its nearest neighbors.

Figure 4 and Figure 5 list the cold-start estimation accuracies of different methods on SO/CPP and SO/Math respectively, with different K values (the number of nearest neighbors). As the accuracy oscillates drastically with a K value smaller than 11 on SO/CPP and smaller than 6 on SO/Math, we report the results with $K \in [11, 20]$ on SO/CPP and $K \in [6, 15]$ on SO/Math. The averaged (over different K values) cold-start estimation accuracies are further given in Table 4. All the results are reported on the cold-start sets, with the optimal parameter settings adopted in Section 5.2. From the results, we can see that 1) Cold-start estimation is possible, and can achieve a considerably high accuracy by choosing a proper method (e.g. RCM), making applications such as better question routing and better incentive mechanism

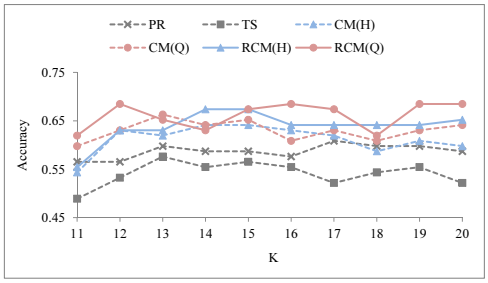


Figure 4: ACC of different methods for cold-start questions on SO/CPP.

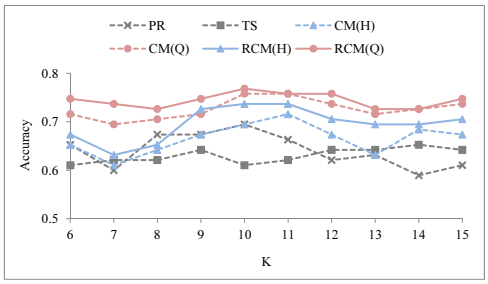


Figure 5: ACC of different methods for cold-start questions on SO/Math.

design feasible in practice. 2) As the value of K varies, RCM (the red/blue solid line) performs almost consistently better than CM with the same loss type (the red/blue dotted line), as well as PR and TS (the gray dotted lines), showing the advantages of RCM in the cold-start estimation. 3) The cold-start estimation accuracies on SO/Math are higher than those on SO/CPP, again demonstrating that the textual descriptions of the SO/Math questions are more powerful in reflecting their difficulty levels. This is consistent with the phenomenon observed in Section 5.2.

5.4 Difficulty Levels of Words

The third experiment explored how a word's difficulty level can be measured by RCM automatically and quantitatively.

On both SO/CPP and SO/Math, we evenly split the range of question difficulty scores (estimated by RCM) into 10 buckets, and assigned questions to the buckets according to their difficulty scores. A larger bucket ID indicates a higher difficulty level. Then, given a word w, we calculated its frequency in each bucket as follows:

$$f_i(w) = \frac{\text{\# questions in bucket } i \text{ where } w \text{ occurs}}{\text{\# all questions in bucket } i}.$$

To make the frequency meaningful, buckets with less than 50 questions were discarded. We picked

	PR	TS	CM		RCM	
			H	Q	H	Q
SO/CPP	0.5870	0.5413	0.6120	0.6304	0.6380	**0.6609**
SO/Math	0.6411	0.6305	0.6653	0.7263	0.6958	**0.7442**

Table 4: Averaged ACC of different methods for cold-start questions.

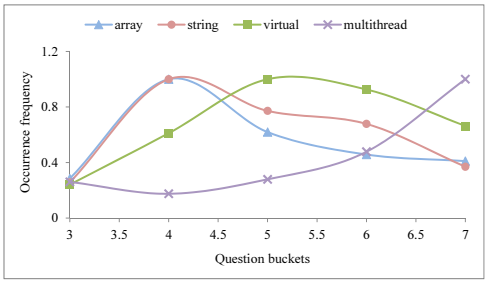


Figure 6: Frequencies of different words in the buckets on SO/CPP.

four words from each data set as examples. Their normalized frequencies in different buckets are shown in Figure 6 and Figure 7. On SO/CPP, we can observe that "array" and "string" occur most frequently in questions with lower difficulty levels, "virtual" higher, and "multithread" the highest. It coincides with the intuition: "array" and "string" are usually related to some basic concepts in programming language, while "virtual" and "multithread" usually discuss more advanced topics. Similar phenomena can be observed on SO/Math. The results indicate that RCM might provide an automatic way to measure the difficulty levels of words.

6 Related Work

QDE is relevant to the problem of estimating task difficulty levels and user expertise levels in crowdsourcing services (Yang et al., 2008; Whitehill et al., 2009). Studies on this problem fall into two categories: 1) binary response based and 2) partially ordered response based. In the first category, binary responses (i.e. whether the solution provided by a user is correct or not) are observed, and techniques based on item response theory are further employed (Whitehill et al., 2009; Welinder et al., 2010; Zhou et al., 2012). In the second category, partially ordered responses (i.e. which of the two given solutions is better) are observed, and pairwise comparison based methods are further adopted (Yang et al., 2008; Liu et al., 2013). QDE belongs to the latter.

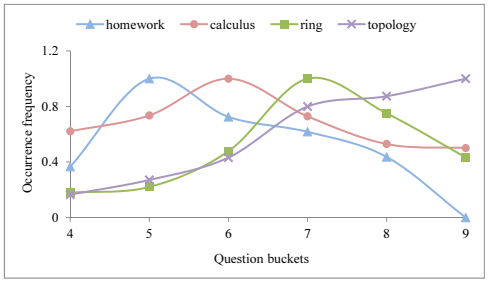


Figure 7: Frequencies of different words in the buckets on SO/Math.

The most relevant work to ours is a pairwise comparison based approach proposed by Liu et al. (2013) to estimate question difficulty levels in C-QA services. They have also demonstrated that a similar approach can be utilized to estimate user expertise levels (Liu et al., 2011). Yang et al. (2008) and Chen et al. (2013) have also proposed pairwise comparison based methods, for task difficulty estimation and rank aggregation in crowdsourcing settings. Our work differs from previous pairwise comparison based methods in that it further utilizes textual information, formalized as a manifold regularizer.

Manifold regularization is a geometrically motivated framework for machine learning, enforcing the learning model to be smooth w.r.t. the geometrical structure of data (Belkin et al., 2006). Within the framework, dimensionality reduction (Belkin and Niyogi, 2001; Cai et al., 2008) and semi-supervised learning (Zhou et al., 2004; Zhu and Lafferty, 2005) algorithms have been constructed. In dimensionality reduction, manifold regularization is utilized to guarantee that nearby points will have similar low-dimensional representations (Cai et al., 2008), while in semi-supervised learning it is utilized to ensure that nearby points will have similar labels (Zhou et al., 2004). In our work, we assume that nearby questions (in terms of textual similarities) will have similar difficulty levels.

Predicting reading difficulty levels of text is also a relevant problem (Collins-Thompson and Callan, 2004; Schwarm and Ostendorf, 2005). It is a key to automatically finding materials at appreciate reading levels for students, and also helps in personalized web search (Collins-Thompson et al., 2011). In the task of predicting reading difficulty levels, documents targeting different grade levels are taken as ground truth, which can be easily obtained from the web. However, there is no naturally annotated data for our QDE task on the web. Other related problems include query difficulty estimation for search engines (Carmel et al., 2006; Yom-Tov et al., 2005) and question difficulty estimation for automatic question answering systems (Lange et al., 2004). In these tasks, query/question difficulty is system-oriented and irrelevant with human knowledge, which is a different setting from ours.

7 Conclusion and Future Work

In this paper, we have proposed a novel method for estimating question difficulty levels in CQA services, called Regularized Competition Model (RCM). It takes fully advantage of questions' textual descriptions besides question-user comparisons, and thus can effectively deal with data sparsity and perform more accurate estimation. A K-Nearest Neighbor approach is further adopted to estimate difficulty levels of cold-start questions. Experiments on two publicly available data sets show that RCM performs significantly better than existing methods in the estimation task, for both well-resolved and cold-start questions, demonstrating the advantage of incorporating textual information. It is also observed that RCM might automatically measure the knowledge levels of words.

As future work, we plan to 1) Enhance the efficiency and scalability of RCM. The complexity analysis in Section 3.2 indicates that storing and processing the graph Laplacian is a bottleneck of RCM. We would like to investigate how to deal with the bottleneck, e.g., via parallel or distributed computing. 2) Apply RCM to non-technical domains. For non-technical domains such as the "news and events" category of Yahoo! Answers, there might be no strongly distinct notions of "experts" and "non-experts", and it might be more difficult to distinguish between "hard questions" and "easy questions". It is worthy investigating whether RCM still works on such domains.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This work is supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (grant No. XDA06030200), the National Key Technology R&D Program (grant No. 2012BAH46B03), and the National Natural Science Foundation of China (grant No. 61272427).

References

Mark S. Ackerman and David W. McDonald. 1996. Answer garden 2: merging organizational memory with collaborative help. In *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work*, pages 97–105.

Mikhail Belkin and Partha Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, pages 585–591.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434.

Stephen Boyd, Lin Xiao, and Almir Mutapcic. 2003. Subgradient methods. *Lecture Notes of EE392o, Stanford University*.

Deng Cai, Xiaofei He, Xiaoyun Wu, and Jiawei Han. 2008. Non-negative matrix factorization on manifold. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 63–72.

David Carmel, Elad Yom-Tov, Adam Darlow, and Dan Pelleg. 2006. What makes a query difficult? In *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 390–397.

Olivier Chapelle. 2007. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178.

Xi Chen, Paul N. Bennett, Kevyn Collins-Thompson, and Eric Horvitz. 2013. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, pages 193–202.

Fan RK. Chung. 1997. *Spectral Graph Theory*, volume 92.

Kenneth Church. 2011. How many multiword expressions do people know. In *Proceedings of the ACL-HLT Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, pages 137–144.

Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. 1999. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the ACM SIGIR workshop on Recommender Systems*.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Kevyn Collins-Thompson and James P. Callan. 2004. A language modeling approach to predicting reading difficulty. In *Proceedings of the 2004 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 193–200.

Kevyn Collins-Thompson, Paul N Bennett, Ryen W White, Sebastian de la Chica, and David Sontag. 2011. Personalizing web search results by reading level. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 403–412.

Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Zhicheng Dou, Ruihua Song, and Ji Rong Wen. 2007. A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of the 16th International Conference on World Wide Web*, pages 581–590.

Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. Trueskill: a bayesian skill rating system. In *Advances in Neural Information Processing Systems*, pages 569–576.

Anna Huang. 2008. Similarity measures for text document clustering. In *Proceedings of the 6th New Zealand Computer Science Research Student Conference*, pages 49–56.

Rense Lange, Juan Moran, Warren R. Greiff, and Lisa Ferro. 2004. A probabilistic rasch analysis of question answering evaluations. In *Proceedings of the 2004 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 65–72.

Baichuan Li and Irwin King. 2010. Routing questions to appropriate answerers in community question answering services. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1585–1588.

Jing Liu, Young-In Song, and Chin-Yew Lin. 2011. Competition-based user expertise score estimation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 425–434.

Jing Liu, Quan Wang, Chin-Yew Lin, and Hsiao-Wuen Hon. 2013. Question difficulty estimation in community question answering services. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 85–90.

Kevin Kyung Nam, Mark S. Ackerman, and Lada A. Adamic. 2009. Questions in, knowledge in?: a study of naver's question answering community. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 779–788.

Larry Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: bringing order to the web. *Technical Report, Stanford University*.

Joseph Lee Rodgers and W. Alan Nicewander. 1988. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523.

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–260.

Sarah E. Schwarm and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 523–530.

Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. 2004. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th International Conference on World Wide Web*, pages 675–684.

Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. 2010. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems*, pages 2424–2432.

Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier R Movellan. 2009. Whose vote should count more: optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems*, pages 2035–2043.

Jiang Yang, Lada Adamic, and Mark Ackerman. 2008. Competing to share expertise: the taskcn knowledge sharing community. In *Proceedings of the 2nd International AAAI Conference on Weblogs and Social Media*.

Elad Yom-Tov, Shai Fine, David Carmel, and Adam Darlow. 2005. Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *Proceedings of the 28th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 512–519.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pages 321–328.

Yanhong Zhou, Gao Cong, Bin Cui, Christian S. Jensen, and Junjie Yao. 2009. Routing questions to the right users in online communities. In *Proceedings of the 25th IEEE International Conference on Data Engineering*, pages 700–711.

Dengyong Zhou, John C Platt, Sumit Basu, and Yi Mao. 2012. Learning from the wisdom of crowds by minimax entropy. In *Advances in Neural Information Processing Systems*, pages 2204–2212.

Xiaojin Zhu and John Lafferty. 2005. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 1052–1059.

Vote Prediction on Comments in Social Polls

Isaac Persing and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{persingq,vince}@hlt.utdallas.edu

Abstract

A poll consists of a question and a set of predefined answers from which voters can select. We present the new problem of vote prediction on comments, which involves determining which of these answers a voter selected given a comment she wrote after voting. To address this task, we exploit not only the information extracted from the comments but also extra-textual information such as user demographic information and inter-comment constraints. In an evaluation involving nearly one million comments collected from the popular SodaHead social polling website, we show that a vote prediction system that exploits only textual information can be improved significantly when extended with extra-textual information.

1 Introduction

We introduce in this paper a new opinion mining task, *vote prediction* on comments in social polls. Recall that a poll consists of a question accompanied by a set of predefined answers. A user who votes on the question will choose one of these answers and will be prompted to enter a comment giving an explanation of why she chose the answer. Given a poll and a user comment written in response to it, the task of *vote prediction* seeks to determine which predefined answer was chosen by the author of the comment.

A solution to the vote prediction problem would contribute significantly to our understanding of the underlying attitudes of individual social polling website users. This understanding could be exploited for tasks such as improving user experience or directed advertising; if we can predict how a user will vote on a question, we can make more accurate guesses about what kind of content/ads related to the question the user would like to see. Unfortunately, a major difficulty of vote prediction arises from the casual nature of discussion in social media. A comment often contains insufficient information for inferring the user's vote, or in some cases may even be entirely absent.

In light of this difficulty, we exploit two additional types of information in the prediction process. First, we employ demographic features derived from user profiles. Demographic features may be broadly useful for other opinion mining tasks such as stance classification (Somasundaran and Wiebe, 2010), as many social media websites like CreateDebate[1] allow users to create profiles with similar demographic information. Previous work has attempted to predict such latent features (e.g., Rao and Yarowsky (2010), Burger et al. (2011)) rather than employing them for opinion mining tasks.

Second, we exploit inter-comment constraints to help us perform joint inference over votes on different questions. Note that previous work on debate stance recognition has also employed constraints to improve the inference process. Specifically, in stance prediction, it is typical to employ so-called author constraints (e.g., Thomas et al. (2006), Bansal et al. (2008), Walker et al. (2012a), Hasan and Ng (2013)), which specify that two documents written by the same author for the same topic should have the same stance. However, in vote prediction, author constraints are not useful because a user is not permitted to cast more than one vote per question, unlike in stance prediction, where users may engage in a debate and therefore post more than once per debate topic. Consequently, we propose two new types of constraints for exploiting inter topic user voting patterns. One constraint involves pairs of authors and the other involves pairs of questions. These constraints are also potentially useful for other opin-

[1] http://www.createdebate.com/

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1127–1138,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

ion mining tasks involving social media, as social media sites typically allow users to comment on multiple topics. Note that enforcing constraints involving two questions is by no means trivial, as the possible class values associated with the two comments may not necessarily be the same.

Another contribution of our work lies in our adaptation of the label propagation algorithm (Zhu and Ghahramani, 2002) to enforce constraints for vote prediction. Recall that existing stance classification approaches enforce constraints using minimum cut (Thomas et al., 2006), integer linear programming (Lu et al., 2012), and loopy belief propagation (Burfoot et al., 2011). Our decision to employ label propagation stems in part from the inability of loopy belief propagation and integer linear programming to efficiently process the nearly one million comments we have, and in part from the inability of the traditional two-way minimum cut algorithm to handle multiclass classification. It is worth noting, however, that other variations of the label propagation algorithm have been proposed for unrelated NLP tasks such as automatically harvesting temporal facts from the web (e.g., Wang et al. (2011) and Wang et al. (2012)).

While we are the first to address the vote prediction task, other researchers have previously used social media to predict the outcomes of various events, primarily by analyzing Twitter data. For example, Tumasjan et al. (2010) and Gayo-Avello et al. (2011) performed the related task of predicting the outcomes of elections. Rather than predicting election outcomes, O'Connor et al. (2010) focused on finding correlations between measures derived from tweets and the outcomes of political events like elections and polls. Finally, Asur and Huberman (2010) predicted movies' box office success. These tasks contrast with our task of vote prediction in that they are concerned with aggregate measures such as the fraction of the vote each candidate or party will win in an election or how much money a movie will make at the box office, whereas vote prediction is concerned with predicting how individual people will vote on a much wider variety of news/political topics.

2 Corpus

SodaHead[2] is a social polling website where users vote on and ask questions about a wide variety of topics ranging from the serious (e.g., "Should the U.S. raise the minimum wage?") to the silly (e.g "What is your favorite kind of pie?"). Whenever a user votes on one of these questions, choosing one of a set of predefined answers, she is prompted to enter a comment giving an explanation of why she chose the answer she did. Our corpus[3] consists of all the comments[4] users posted under all featured questions in the News & Politics category of the SodaHead website between March 12, 2008 and August 21, 2013.

This dataset consists of a total of 997,379 comments over 4,803 different questions, so an average of 208 comments are written in response to each question. The length of an average comment is 49 words. As Table 1 illustrates, these questions may have more than two possible answers, with an average question having 2.4 possible answers.

Each SodaHead user has her own profile that contains demographic information about her. As we can see from Table 2, many users choose to provide only some information about themselves, leaving many of the demographic fields blank. 108,462 users posted at least one comment in our corpus, with an average user commenting on 9.2 of our questions.

3 Baseline Systems

To perform our experiments, we first split our comments into three sets, a test set for evaluating performance, a training set for training classifiers, and a development set for tuning parameters. In order to ensure that the comparisons of our experiments are valid, we construct our test set using the same 20% of comments in the dataset regardless of experiment. Since our goal is to plot a learning curve illustrating how our various vote prediction systems perform given different amounts of training and development data, we vary the size of our training and development sets across experiments so that in the smallest experiment, together they comprise 25% of the remaining (non-test) comments, and in the largest experiment, they com-

[2] http://www.sodahead.com

[3] http://www.hlt.utdallas.edu/%7epersingq/SocialPolls/ is the distribution site for our corpus. We preserve user anonymity by replacing the original id of each user with a random number in our corpus.

[4] A "comment" is the text a user posted when submitting her vote on a question. It does not include posts not associated with a vote (such as responses to other posts) or votes where the user chose not to enter a comment. Thus, there is a one-to-one relationship between comments in votes in our dataset. The vote associated with a comment is always known.

Question	Vote	Comment
Who Won Round Two of the Presidential Debate?	Barack Obama	Binders full of women. That is all.
	Mitt Romney	Obama is inept and a liar. We can't survive 4 more years of his crazy crap.
What's the Best Way to Read a Magazine?	in print	Upside down like Luna Lovegood.
	online	Print costs money. It also doesn't have a Search function.
	on a tablet device	since sooooo many people have tablet devices why read it as print or online?
	on a smartphone	Clicked in print!!! Aargh

Table 1: Sample questions and comments. All of the pre-defined answers for these questions are represented by one comment.

User ID	3479864	3189372
Age	25-34	
Smoker		No
Drinker		No
Income		
Sexual Orientation		Straight
Relationship Status		Single
Political Views	Conservative	Moderate
Ethnicity		
Looking For		
Career Industry		
Children		Undecided
Education		High School
Gender	Female	Male
Religious Views	Other	Christian
Employment Status		
Weight Type		

Table 2: Sample user profiles.

prise 100% of the remaining comments. For each experiment, we maintain a ratio of three training comments to one development comment.

Recall that each comment in our dataset is written in response to a particular question. For each test comment, our goal is to predict the user's answer to the question given the text of her comment. One of the major inherent difficulties of our task is that it consists not of one, but of 4,803 separate multiclass classification problems (one for each question). As a result, our approach to the problem necessarily has to be somewhat generic, as it would be too time-consuming to develop an appropriate feature set for each question.

3.1 Baseline 1

Our first baseline's (B_1) approach employs 4,803 multiclass classifiers (one for each question). Each classifier is trained on one question's training set, representing each comment using only a bias feature. Each of our classifiers is trained using MALLET's (McCallum, 2002) implementation of maximum entropy (ME) classification. This is equivalent to merely counting the number of training set comments that voted for each possible answer, selecting the most frequent answer, then applying this label to all the comments in the test set. This majority baseline serves primarily to tell us how well our more sophisticated baseline performs.

3.2 Baseline 2

Our second baseline (B_2) is constructed in exactly the same way as B_1 except that each classifier is trained using both a bias feature and a standard set of feature types described below.

3.2.1 Features

Since the questions in our dataset come from the News & Politics category of the SodaHead website, many of the questions' topics are political. For that reason, it makes sense to use features which have been shown to work well on other political classification problems. We therefore base our feature set on that used by Walker et al. (2012b) for political debate classification. Our features are described below.

N-grams. Unigrams have been shown to perform well in ideological debates (Somasundaran and Wiebe, 2010), so we therefore present our classifiers with lemmatized unigram, bigram, and trigram features. We normalize the n-gram feature vector to unit length to avoid giving undue influence to longer comments.

Cue Words. Based on other work (Fox Tree and Schrock, 1999; Fox Tree and Schrock, 2002; Groen et al., 2010; Walker et al., 2012b), we also present our classifiers with features representing the first lemmatized unigram, bigram, and trigram appearing in each comment. These may be useful in our task when, for example, a user's comment begins with or entirely consists of a restatement of the answer she chose. So if the possible answers for a given question are "Yes" and "No", a user might write in her comment "Yes. Because ...", and this would make the "CueWord:Yes" feature useful for classifying this comment.

Emotion Frequency. For each word in a comment, we used the NRC Emotion Word Lexicon

(Mohammad and Yang, 2011) to discover if the word conveys any emotion. Then, for each emotion or sentiment covered by the lexicon (anger, anticipation, disgust, fear, joy, sadness, surprise, trust, positive, or negative) e_i, we construct a feature e_i:$\frac{C(e_i)}{total}$ describing how much of the comment consists of words conveying emotion e_i, where $C(e_i)$ is the count of words in the comment bearing emotion e_i and $total$ is the number of words in the comment. To understand why this feature may be useful, consider the question "Does Sarah Palin deserve VP?" We suspect that users who post comments laden with words associated with positive emotions like joy are more likely to vote "Yes" because the positive emotions imply they are happy about a Sarah Palin vice presidency. Similarly, users who post comments laden with negative emotions like anger might be more likely to vote "No".

Dependencies. We use the Stanford Parser (de Marneffe et al., 2006) to extract a set of dependencies from each comment. For an example of how dependencies might help in our task, consider the second comment in Table 1. From this comment, we can extract the dependency triple dependency:(nsubj,inept,obama), which indicates that the user who wrote it does not like Obama and is therefore more likely to have voted for Romney in the question. Dependency feature vectors are normalized to unit length.

Emotion Dependencies. To form an emotion dependency feature, we take a regular dependency feature and replace each of its words where possible with the emotion it evokes as determined by the NRC Emotion Word Lexicon. Thus from the dependency:(nsubj,inept,obama) example above, we would generate three features: emotiondependency:(nsubj,anger,obama), emotiondependency:(nsubj,disgust,obama), and emotiondependency:(nsubj,negative,obama). These features help generalize dependencies, and this is useful because predictive features like emotiondependency:(nsubj,negative,obama) appear frequently in the comments for this question, but dependency:(nsubj,inept,obama) does not. Emotion Dependency feature vectors are normalized to unit length.

Post Information. Features under this category just calculate some basic statistics about a comment. These features may be useful because, for example, the question "Most Scandalous Politicians of 2008– Who deserves the title?" has six possible answers, each except the last naming a particular well-known politician. The last choice is "The most scandalous politician of 2008 is ..." and the user is expected to name a politician in her comment. It would make sense for users choosing this option to have written longer responses since they have to name and possibly explain their choice to users who might not necessarily know who their chosen politician is.

3.2.2 Feature Selection

Because some of the feature types (n-grams, cue words, dependencies, and emotion dependencies) described in the previous subsection are expected to generate a large number of non-predictive features, we trim some of the most irrelevant features out of the feature set to avoid memory problems. Therefore, following Yang and Pedersen (1997), for each question we calculate the information gain of each feature of these types on the training set. We then remove those features having the lowest information gain as well as those features occurring less than ten times in the dataset. Early experiments showed that 1,000 was a reasonable number of features to keep, so for all experiments we keep only the top 1,000 features of these types. Note that we do not apply feature selection to emotion frequency or post information features, as each of these sets consists of a small number of real-valued features.

4 Demographic Features

As mentioned in the introduction, a major difficulty inherent to our problem is that in many cases a comment contains insufficient information for inferring the underlying vote. Aside from being short, the comments shown in Table 1 are typical of comments found in the dataset. Some comments are like the first and third in the table, requiring some obscure bit of world knowledge to understand what the writer is saying. Others like the fourth only explain why the user did not choose a particular answer, which is always potentially useful, but sufficient only if the comment excludes every other possible choice.

Because it is difficult to tell how a user voted given her comment, we exploit the demographic information users provide in their profiles as an additional source of information. Since many of the questions in our dataset deal with politics, we anticipate that information about things

such as whether a comment was written by a conservative or progressive user would be useful for predicting the answers of many comments. For each comment, we encode demographic information as features in the following way. For each field in the user's profile shown in Table 2 (aside from user ID), we construct a feature of the form F_i:V_i if the user filled in field F_i with value V_i. Thus, any comment made by user 3479864 would include the features Age:25−34, PoliticalViews:Conservative, Gender:Female, and Religion:Other.

Here is an example of a comment whose predicted vote gets corrected by adding demographic features to our system. For the question, "LPGA Decides to Allow Transgender Competitors: Good or Bad Move for Golf?", user 2252750 writes, "LPGA ...can let monkeys play if they wish....nobody gives a rip... bark". Of the three possible answers for this question, "Good move", "Bad move", and "Undecided", our baseline system without demographics believes that user 2252750 probably voted for the third, as "nobody gives a rip" makes him sound apathetic toward the issue. However, our demographic system notices that his profile contains "Religion:Christian", and users with this demographic attribute choose "Bad Move" 64% of the time. Thus, demographic features allowed our system to correctly predict his vote for "Bad Move".

Since demographics are also expected to generate a large number of non-predictive features, we apply feature selection to them as described in Section 3.2.2.

5 Enforcing Constraints

We mentioned earlier that an average SodaHead question contains 208 comments. This implies that there are only about 31−125 comments[5] in the average training set for one of our ME classifiers. It would be difficult to train a good classifier from a training set this small even if we had feature sets tailored to work well on each of the 4,803 questions. While we have already attempted to exploit user information (in the form of demographic features) to help improve our system's performance, this approach still treats the task as 4,803 separate classification problems. It does not allow for the possibility that classification on one question may be improved by exploiting information gleaned from votes on other questions.

One way we might exploit such information is by first noticing that, for any pair of questions, there may be multiple users who commented on both. This overlap between questions allows us to calculate how predictive a user's vote on one question is of how she will vote on the other. For example, on the question "Who Would You Rather Have Dinner With?", we found that users who voted for "Mitt Romney" were much more likely to choose "No, I'm still voting for him" on the question "Does Mitt Romney's 'Entitled' Remarks Change Your Opinion of Him?". Similarly, users who voted to have dinner with "Barack Obama" were much more likely to vote "Yes, I'm not voting for him anymore" on the "entitled" question. A system that somehow takes into account this information might correctly classify a difficult comment on the "entitled" question if it notices that the comment was written by a user who commented on both questions and it knows how the user voted on the "dinner" question. We call the kind of constraint described here a **QuestionPair** constraint.

We might also exploit information from other questions by noticing that there are users who share similar attitudes on a wide variety of topics in our dataset. We can gauge how often a pair of users agree with each other by comparing their votes on every question on which they have both voted where their comments appear in the training set. So for example, if we see that two users have agreed on questions about George H.W. Bush, Bill Clinton, and George W. Bush, we can guess that they will also agree on a question about Barack Obama. Similarly, if they disagreed on all those questions, they are likely to disagree on the last question. A system that takes into account this kind of information could correctly classify an otherwise difficult comment if it knows how another user voted on this question and also knows how often the two users agree on other questions. We call the kind of constraint described here a **VoterPair** constraint.

In order to enforce both kinds of constraints, we introduce a variation of the label propagation algorithm (Zhu and Ghahramani, 2002). In our version of the label propagation algorithm, each comment in our dataset is represented by a node in a graph. Each node is associated with a probability distribution indicating the likelihood that the

[5] At the low and high end of the learning curve respectively.

comment belongs to each of its question's possible answers. Thus, when we initialize the graph, each training set node's probability distribution is set to reflect its comment's actual label (with a probability of 1 for the comment's actual label and 0 for each other answer), and each development or test set node's probability distribution is set to the value predicted by another classifier such as B_2 or $B_2 + Dem$ since the algorithm is not permitted to see the comment's actual label. Lines 7−12 in Figure 1 describe the graph's initialization.

Now that we have set up the graph's nodes, we need to explain how our graph's edges work. As we discussed earlier in this section, the edges in our graph will represent two kinds of soft constraints. Each edge allows one of a node's neighbors to cast a vote (in the form of a probability distribution over possible answers) for what it thinks the node's answer should be. Let us call the comment node whose label we are trying to predict the **target** node and the comment node which casts the vote the **source** node.

Our graph contains a QuestionPair edge between any source and target comments written by the same user. Since a user cannot comment more than once on any question, the source and target comments will occur in two different questions. In order to determine how the source node votes over a QuestionPair edge, we need to calculate some probabilities. In particular, we need to determine the probability that a user will vote for possible answer k in the target question Q_I given that she voted for answer l in the source question Q_J:

$$P(Q_{I_k}|Q_{J_l}) = \frac{C(Q_{I_k},Q_{J_l})+\gamma}{\sum_{m\in A(Q_I)}(C(Q_{I_m},Q_{J_l})+\gamma)}$$

where $C(Q_{I_n}, Q_{J_l})$ is the number of users who voted for answer n in Q_I and answer l in Q_J, and $A(Q_I)$ is the set of possible answers on Q_I. We set γ, the smoothing factor, to 10 since this value worked well in earlier experiments. The source node S casts its vote on target node T for the probability distribution given by:

$$QP_{S,T}(Q_{I_k}) = \sum_{m\in A(Q_J)} P_S(Q_{J_m})P(Q_{I_k}|Q_{J_m})$$

where $P_S(Q_{J_m})$ is the probability currently associated with answer m in S's question (Q_J).

The graph contains a VoterPair edge between any source and target nodes on the same question if the users who posted these comments have both voted on at least one other question together and their comments on the other question(s) occurred in the training set. To determine how the source node votes over a VoterPair edge, we need to calculate the probability that the source and target users will agree on a generic issue:

$$P_{agr}(U_S, U_T) = \frac{C_{agr}(U_S,U_T)+1}{C_{agr}(U_S,U_T)+C_{dis}(U_S,U_T)+2}$$

where $C_{agr}(U_S, U_T)$ is the number of questions on which users U_S and U_T voted for the same answer and both their comments occurred in the training set, $C_{dis}(U_S, U_T)$ is the number of questions on which U_S and U_T voted for different answers where both their comments occurred in the training set, and the $+1$ and $+2$ are used for smoothing. The probability distribution that the source node S votes for on target node T is then given by:

$$VP_{S,T}(Q_{I_k}) = P_S(Q_{I_k})P_{agr}(U_S, U_T) + \sum_{\substack{m\in A(Q_I),\\ m\neq k}} (P_S(Q_{I_m}))\frac{1-P_{agr}(U_S,U_T)}{|A(Q_I)|-1}$$

where $P_S(Q_{I_n})$ is the probability currently associated with answer n in the source node's question (Q_I), and $|A(Q_I)|$ is the number of possible answers on Q_I. We divide the second term, which deals with disagreement, by $|A(Q_I)| - 1$ because, even if we know that the target and source users disagreed on the answer to a particular question and that the source user did not vote for answer k, there is only a $\frac{1}{|A(Q_I)|-1}$ chance that the target user voted for answer k since there are $|A(Q_I)|-1$ non-k answers to choose from.

Now that we have described how edges are added to the graph and how source comment nodes vote over the edges, we are ready to begin iterating over the label propagation algorithm (line 13 in Figure 1). For each iteration of the algorithm, we update each development or test set node's answer probability distribution by assigning it a weighted sum of (1) the initial probability distribution assigned to the node, (2) the sum of the QuestionPair edges' votes, and (3) the sum of the VoterPair edges' votes (line 16 in Figure 1). Upon completion of the algorithm, if our soft constraints work as expected, the new labeling of comment nodes should be more accurate than their initial labeling.

We tune the parameters W_I, W_V, W_Q, and $iterations$ jointly by an exhaustive search of the parameter space to maximize classification accuracy on the development set. Each of the weight parameters is allowed to take one of the values 0, 1, or 2, and the iteration parameter is allowed take one of the values 0, 1, 2, 3, 4, 5.

1: $LabelPropagation(Tr, D, Te, iterations, W_i, W_V, W_Q, I)$
2: **Inputs:**
3: Tr, D, Te: Comments in Training, Development, and Test set
4: $iterations$: The number of iterations to perform
5: W_i, W_V, W_Q: Weights assigned to initial, VoterPair, and QuestionPair constraints
6: I: Initial answer probability distribution for all comments. Should reflect actual labels for training set comments and classifier predictions for development and test set comments
7: **for all** $C \in Tr \cup D \cup Te$ **do**
8: Create node representing C
9: $C_p \leftarrow I_C$
10: // C_p: node C's current probability distribution over possible answers
11: // I_C: initial answer probability distribution for comment C
12: **end for**
13: **for** $j = 1$ to $iterations$ **do**
14: **for all** node $C \in D \cup Te$ **do**
15: Add all edges targeting node C
16: $C_p \leftarrow Norm(W_I I_C + W_V \sum_k VP_{k,C} + W_Q \sum_k QP_{k,C})$
17: // $VP_{k,C}$, $QP_{k,C}$: kth VoterPair, and kth QuestionPair votes for node C
18: Remove all edges targeting node C
19: **end for**
20: **end for**

Figure 1: Our label propagation algorithm.

One may be surprised to notice how we add edges to the graph in the algorithm only to delete them three lines later (lines 15 and 18 in Figure 1). Though edges can be added at any point in the algorithm, one benefit of using the label propagation algorithm is that it is simple enough that it is not necessary store all the edges in memory at once. The only time we need to store an edge is when its target is being voted on. This means that the label propagation algorithm can handle large datasets like ours with huge numbers of nodes and edges without being prohibitively space-expensive.

6 Evaluation

6.1 Experimental Setup

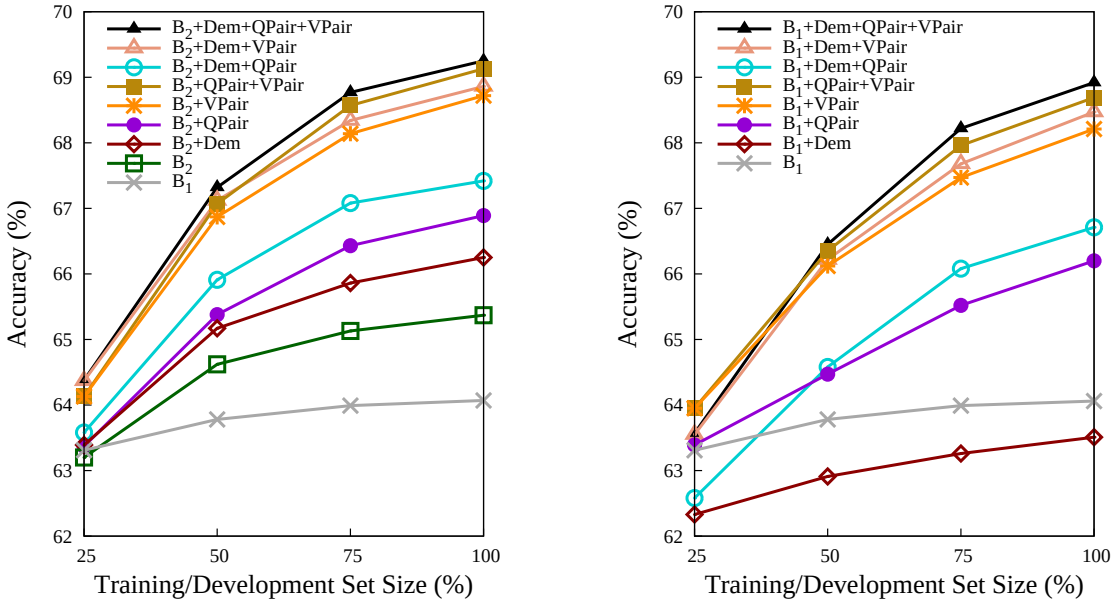
We mentioned in Section 3 that we split our dataset of 997,379 comments into a test set comprising about 20% of the dataset's comments and a training and development set comprising some fraction of the remaining 80% of the comments. We actually split the data up like this five different times so that each comment appears in an experiment's test set exactly once. In this way, through the use of five fold cross-validation, we can report our results on the entire dataset.

6.2 Results and Discussion

Figure 2a shows the accuracy of the predictions made by various systems. First, let us compare our first and second baselines. Recall that the first baseline (B_1) predicts that all test comments will have the same label as the majority of training comments, and the second baseline's (B_2) predictions are the output of ME classifiers trained with a generic feature set. As we can see from the graph, at very small training set sizes, the standard set of features supplied to B_2 does little more than confuse the ME learner, as it performs slightly but not significantly worse[6] than the first baseline when the training/development set comprises only 25% of the available data. This is understandable, as 25% of an average question's available data is only 42 comments, an extremely small number of examples to learn from for most NLP tasks. Clearly a better approach than the one provided by the second baseline is needed. Though the average training set sizes at the 50%, 75%, and 100% levels are still relatively small, B_2 significantly outperforms B_1 at all these levels.

The small improvement sizes yielded by B_2 may be attributable to some of the inherent difficulties of the problem, particularly that (1) it is composed of so many (4,803) separate subproblems that it is impractical for us to tailor a unique feature set for each one, (2) the average question is associated with a very small number of comments (about 208), making it difficult to train a reasonably good classifier for any question, and (3) many of the comments contain insufficient information for inferring the underlying votes. Perhaps some of our proposed extensions to B_2 can help address

[6] All significance tests are paired t-tests, with $p < 0.05$. Because we calculate a large number of significance results, the p values we report are obtained using Holm-Bonferroni multiple testing correction (Holm, 1979).



(a) Vote prediction.

(b) Arbitrary User Vote Prediction.

Figure 2: Five-fold cross-validation vote prediction learning curves.

some of these problems.

The first improvement we proposed involved exploiting demographic features provided by users to help with our prediction tasks. When we combine Dem and B_2's feature sets, the resulting system ($B_2 + Dem$) performs better than any of the systems discussed thus far at all four training/development set size levels, yielding significant improvements over B_2 at all four levels. This demonstrates that our demographic features are a useful complement to a standard approach like the one used by B_2.

The second improvement we proposed involved using a variation of the label propagation algorithm to enforce QuestionPair constraints. QuestionPair constraints, recall, allowed us to exploit the observed voting patterns of users who voted in the training set on any particular pair of questions. These constraints were expected to improve our predictions for any user who voted on both questions when at least one of their votes appeared in the test set. System $B_2 + QPair$ corresponds to following the algorithm in Figure 1, using system B_2's ME classifiers to initialize a label propagation graph, and then setting the VoterPair edge weight (W_V) to 0, thus allowing only QuestionPair constraints. When we compare this system to B_2, we see that the performance boost QuestionPair constraints give us over the baseline is consistently greater than the boost given by adding demographic features to it ($B_2 + Dem$) across all training/development set sizes. The improvement over B_2 is even significant at the 75% and 100% training/development set sizes.

The last improvement we proposed involved adding VoterPair constraints to the label propagation graph. Recall that VoterPair constraints allowed us to exploit how frequently we observed two users agreeing with each other to predict whether they will agree on any question they both voted on. System $B_2 + VPair$ corresponds to following the label propagation algorithm using B_2's ME classifiers to initialize the graph, then setting the QuestionPair edge weight (W_Q) to 0, thus allowing only VoterPair constraints. The addition of VoterPair constraints yields the largest significant improvements over B_2 at all four levels, indicating that, in the absence of our other proposed improvements, VoterPair edge constraints are the most important addition we can make to our baseline.

While we have now shown that each of our proposed extensions yields significant improvements over B_2, this does not necessarily mean that each one is useful in the presence of the others. For example, it might be the case that QuestionPair constraints and Demographic features correct the same kinds of classification errors, and therefore it may be sufficient to use either one or the other to obtain good results, but using both is unnecessary. To test how useful they are in each other's pres-

ence, we perform the following experiment. First, we run the algorithm using all three improvements ($B_2 + Dem + QPair + VPair$ in Figure 2a). We then run the same experiment three more times, each time removing one of the three extensions. By measuring how much performance decreases when we remove each of the three improvements, we can determine whether each improvement provides unique useful information, or whether the information it provides is already being provided by one of the other improvements.

To see what happens when we remove demographic features from the full system, we need to compare $B_2 + Dem + QPair + VPair$ and $B_2 + QPair + VPair$ in Figure 2a. While the decrease in performance after removing demographic features was modest, the difference is nevertheless significant at all four training/development set sizes, suggesting that demographic features do provide unique information to the system.

By comparing line $B_2 + Dem + QPair + VPair$ to line $B_2 + Dem + VPair$, we can determine the impact of QuestionPair constraints. Removing QuestionPair constraints also had a modest impact on the full system's performance, decreasing accuracy at all four training/development set sizes, significantly so at the 50%, 75%, and 100% levels. Interestingly, the impact of QuestionPair constraints appears to grow with the training set, while the demographic features appear to have a greater impact when the training set is small. We can see this by noting that the two lines cross at around 55%. This suggests that QuestionPair constraints are especially useful in problems where it is cheap to obtain a lot of training data, but in problems where the data has to be manually annotated, demographic features are more useful.

Finally, we can compare line $B_2 + Dem + QPair + VPair$ to line $B_2 + Dem + QPair$ to see what happens when we remove VoterPair constraints from our system. This comparison illustrates that VoterPair constraints are by far the most important improvement we removed from the full system, as removing them yielded large significant decreases at all four levels.

Though thus far we have only used it to analyze the the contributions of different individual improvements, the full system $B_2 + Dem + QPair + VPair$ is interesting in itself. Of all the systems we have constructed, it performs the best, yielding improvements of up to 5.18% and 3.88% when compared to B_1 and B_2 respectively. Its improvements over both baselines are statistically significant at all four training/development set sizes.

6.3 Arbitrary User Vote Prediction

One interesting question that we have not yet addressed is, is it possible to predict how a user would vote on a question she has not yet seen? This problem is interesting because an average question receives votes from only 0.2% of the users in our dataset, and thus a system for predicting an arbitrary user's vote would be able to predict the votes of the other 99.8% of users. A solution to this prediction problem would have practical applications in areas such as directed advertising (e.g., if we could predict how a user would vote on the magazine question in Table 1, we would have a better idea of what kinds of reading devices/services would interest her).

We can mimic this problem with our dataset by treating the comment text associated with test votes as unseen since we cannot expect an arbitrary user to have commented on any particular question we are interested in[7]. It does, however, make sense for us to expect our arbitrary user to have provided some personal demographic information, and thus a system for making these types of predictions could reasonably make use of demographic features. Similarly, in this situation we would expect to have knowledge of all users' training set voting histories. Thus, it would also be reasonable for our system to exploit the QuestionPair and VoterPair constraints described in Section 5. Thus, to test how well our system performs on this task, we repeat all experiments from the previous section while replacing B_2 (which uses a ME classifier trained on comment-based features) with B_1 (the most frequent baseline, which uses a ME classifier trained using only a bias feature). The results of these experiments are shown in Figure 2b.

If we compare the results from B_1 to $B_1 + Dem$ (which compliments B_1's bias feature with the demographic feature set), we notice that $B_1 + Dem$ is significantly worse than B_1 at all training set sizes. This confirms our suspicion from the pre-

[7] Although we are trying to mimic the situation in which we predict how an arbitrary user would vote on an arbitrary question, we caution that the vote data we train and evaluate on was not obtained from a set of arbitrary SodaHead users. It consists only of votes from users who chose which questions they wanted to answer. For this reason, the data we train and evaluate on for any question might not be a representative sample of SodaHead users as a whole.

vious section that demographic features by themselves serve only to confuse the learner, though we will see in a moment that they are a helpful supplement to more sophisticated systems.

We can evaluate QuestionPair constraints in this setting by comparing the results from B_1 to $B_1 + QPair$. $B_1 + QPair$ consistently outperforms B_1 at all four training set sizes, significantly so at the 75% and 100% levels, and thus QuestionPair constraints are also a useful addition to our system.

VoterPair constraints can be evaluated in this setting by comparing B_1 to $B_1 + VPair$. $B_1 + VPair$ significantly outperforms B_1 at all four training set sizes, and from the graph it appears to be our most beneficial improvement.

To evaluate whether demographic features are useful in the presence of the other improvements, we compare the full system, $B_1 + Dem + QPair + VPair$, to its corresponding version without demographic features, $B_1 + QPair + VPair$. Though $B_1 + QPair + VPair$ significantly outperforms the full system at the 25% training set size, the full system significantly outperforms $B_1 + QPair + VPair$ at the 75% and 100% levels, indicating that in this setting, demographic features are useful in the presence of a large training set.

We can evaluate the utility of QuestionPair constraints in this setting by comparing the full system to $B_1 + Dem + VPair$. When we remove QuestionPair constraints, accuracy is consistently lowered at all four training set sizes, significantly so at 50%, 75%, and 100%. This tells us that QuestionPair constraints are useful in this setting.

We can evaluate how useful VoterPair constraints are by checking how much $B_1 + Dem + VPair + QPair$'s performance drops when we remove VoterPair constraints from it, yielding $B_1 + Dem + QPair$. Performance drops considerably and significantly at all four training set sizes after removing VoterPair constraints, suggesting that in this setting, VoterPair constraints are still the most important of our proposed improvements.

Finally, while we have already established that all our proposed improvements can improve performance under both settings (comments visible and comments invisible), it may be worthwhile to compare the two sets of experiments to determine whether the comment features used in systems with B_2 are useful.

A casual inspection of the two figures shows that, broadly, each system that uses comment-based features in Figure 2a tends to slightly outperform the most comparable system in Figure 2b. At the low end of the curves, the two systems often differ by about 1.0% in absolute accuracy, though at the high end, the difference tends to be much smaller, with the full system with comment features outperforming the full system without comment features by only 0.3%. Since in this setting it is reasonable to assume a large training set, this last result is the one we are most interested in, and it suggests that our full system's performance does not suffer much due to the absence of comment features.

One final observation we can make is that, when comments are not visible, demographic features appear to actively harm the performance of systems trained on a small amount of data, though at larger training set sizes they are mostly helpful. We can tell this by comparing systems with demographic features to systems without them in Figure 2b (e.g., by comparing $B_1 + Dem + QPair$ to $B_1 + QPair$ or $B_1 + Dem + VPair$ to $B_1 + VPair$) at the 25% training set size. This is not the case in the setting where comments are visible, as we see that demographic features always appear helpful in Figure 2a. This reinforces the notion that demographic features provide useful information in general, but that they are by themselves too sparsely available to do more than confuse the learner. They need to be supplemented by other information sources in order for the learner to draw correct conclusions.

7 Conclusion

We examined the task of vote prediction on comments from the SodaHead website. To address this task, we exploited not only information extracted from the comments but also extra-textual information, including demographic information and two types of inter-comment constraints, QuestionPair constraints and VoterPair constraints. Our experiments involving 997,379 comments showed that each of these extensions significantly improved a baseline that exploited only textual information, with VoterPair constraints being the most effective and demographic information being the least effective. When used in combination, they obtained up to a 3.88% improvement in absolute accuracy over the baseline. To stimulate research on this task, we make our dataset publicly available.

Acknowledgments

We thank the three anonymous reviewers for their detailed and insightful comments on an earlier draft of this paper. This work was supported in part by NSF Grants IIS-1147644 and IIS-1219142. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of NSF.

References

Sitaram Asur and Bernardo A. Huberman. 2010. Predicting the future with social media. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 492–499.

Mohit Bansal, Claire Cardie, and Lillian Lee. 2008. The power of negative thinking: Exploiting label disagreement in the min-cut classification framework. In *COLING 2008: Companion Volume: Posters*, pages 15–18.

Clinton Burfoot, Steven Bird, and Timothy Baldwin. 2011. Collective classification of congressional floor-debate transcripts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1506–1515.

John D. Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Gender discrimination on twitter. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1301–1309.

Marie-Catherine de Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 449–454.

Jean E. Fox Tree and Josef C. Schrock. 1999. Discourse markers in spontaneous speech: Oh what a difference an oh makes. *Journal of Memory and Language*, 40:280–295.

Jean E. Fox Tree and Josef C. Schrock. 2002. Basic meanings of you know and i mean. *Journal of Pragmatics*, 34:427–447.

Daniel Gayo-Avello, Panagiotis Takis Metaxas, and Eni Mustafaraj. 2011. Limits of electoral predictions using twitter. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, pages 490–493.

Martin Groen, Jan Noyes, and Frans Verstraten. 2010. The effect of substituting discourse markers on their role in dialogue. *Discourse Processes: A Multidisciplinary Journal*, 47:388–420.

Kazi Saidul Hasan and Vincent Ng. 2013. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356.

Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70.

Yue Lu, Hongning Wang, ChengXiang Zhai, and Dan Roth. 2012. Unsupervised discovery of opposing opinion networks from forum discussions. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 1642–1646.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Saif Mohammad and Tony Yang. 2011. Tracking sentiment in mail: How genders differ on emotional axes. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pages 70–79.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 122–129.

Delip Rao and David Yarowsky. 2010. Detecting latent user properties in social media. In *Proceedings of the NIPS workshop on Machine Learning for Social Networks*.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 327–335.

Andranik Tumasjan, Timm Sprenger, Philipp Sandner, and Isabell Welpe. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 178–185.

Marilyn Walker, Pranav Anand, Rob Abbott, and Ricky Grant. 2012a. Stance classification using dialogic properties of persuasion. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 592–596.

Marilyn A. Walker, Pranav Anand, Rob Abbott, Jean E. Fox Tree, Craig Martell, and Joseph King. 2012b. That is your evidence?: Classifying stance in online political debate. *Decision Support Systems*, 53(4):719–729.

Yafang Wang, Bin Yang, Lizhen Qu, Marc Spaniol, and Gerhard Weikum. 2011. Harvesting facts from textual web sources by constrained label propagation. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 837–846.

Yafang Wang, Maximilian Dylla, Marc Spaniol, and Gerhard Weikum. 2012. Coupling label propagation and constraints for temporal fact extraction. In *Proceedings of the ACL 2012 Conference Short Papers*, pages 233–237.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, pages 412–420.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, CMU CALD.

Exploiting Social Relations and Sentiment for Stock Prediction

Jianfeng Si* **Arjun Mukherjee†** **Bing Liu†** **Sinno Jialin Pan*** **Qing Li‡** **Huayi Li†**
* Institute for Infocomm Research, Singapore
{ thankjeff@gmail.com, jspan@i2r.a-star.edu.sg}
†Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA
{ arjun4787@gmail.com, liub@cs.uic.edu, lhymvp@gmail.com}
‡ Department of Computer Science, City University of Hong Kong, Hong Kong, China
qing.li@cityu.edu.hk

Abstract

In this paper we first exploit cash-tags ("$" followed by stocks' ticker symbols) in Twitter to build a stock network, where nodes are stocks connected by edges when two stocks co-occur frequently in tweets. We then employ a labeled topic model to jointly model both the tweets and the network structure to assign each node and each edge a topic respectively. This Semantic Stock Network (SSN) summarizes discussion topics about stocks and stock relations. We further show that social sentiment about stock (node) topics and stock relationship (edge) topics are predictive of each stock's market. For prediction, we propose to regress the topic-sentiment time-series and the stock's price time series. Experimental results demonstrate that topic sentiments from close neighbors are able to help improve the prediction of a stock markedly.

1 Introduction

Existing research has shown the usefulness of public sentiment in social media across a wide range of applications. Several works showed social media as a promising tool for stock market prediction (Bollen et al., 2011; Ruiz et al., 2012; Si et al., 2013). However, the semantic relationships between stocks have not yet been explored. In this paper, we show that the latent semantic relations among stocks and the associated social sentiment can yield a better prediction model.

On Twitter, cash-tags (e.g., $aapl for Apple Inc.) are used in a tweet to indicate that the tweet talks about the stocks or some other related information about the companies. For example, one tweet containing cash-tags: $aapl and $goog (Google Inc.), is *"$AAPL is loosing customers. everybody is buying android phones! $GOOG"*. Such joint mentions directly reflect some kind of latent relationship between the involved stocks, which motivates us to exploit such information for the stock prediction.

We propose a notion of Semantic Stock Network (SSN) and use it to summarize the latent semantics of stocks from social discussions. To our knowledge, this is the first work that uses cash-tags in Twitter for mining stock semantic relations. Our stock network is constructed based on the co-occurrences of cash-tags in tweets. With the SSN, we employ a labeled topic model to jointly model both the tweets and the network structure to assign each node and each edge a topic respectively. Then, a lexicon-based sentiment analysis method is used to compute a sentiment score for each node and each edge topic. To predict each stock's performance (i.e., the up/down movement of the stock's closing price), we use the sentiment time-series over the SSN and the price time series in a vector autoregression (VAR) framework.

We will show that the neighbor relationships in SSN give very useful insights into the dynamics of the stock market. Our experimental results demonstrate that topic sentiments from close neighbors of a stock can help improve the prediction of the stock market markedly.

2 Related work

2.1 Social Media & Economic Indices

Many algorithms have been proposed to produce meaningful insights from massive social media data. Related works include detecting and summarizing events (Weng and Lee, 2011; Weng et al., 2011; Baldwin et al., 2012; Gao et al., 2012) and analyzing sentiments about them (Pang and Lee, 2008; Liu, 2012), etc. Some recent literature also used Twitter as a sentiment source for stock market prediction (Bollen et al., 2011; Si et al., 2013). This paper extends beyond the correlation between social media and stock market, but fur-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1139–1145,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

ther exploits the social relations between stocks from the social media context.

Topic modeling has been widely used in social media. Various extensions of the traditional LDA model (Blei et al., 2003) has been proposed for modeling social media data (Wang et al., 2011, Jo and Oh, 2011; Liu et al., 2007; Mei et al., 2007; Diao et al., 2012). Ramage et al. (2009; 2011) presented a partially supervised learning model called Labeled LDA to utilize supervision signal in topic modeling. Ma et al. (2013) predicted the topic popularity based on hash-tags on Twitter in a classification framework.

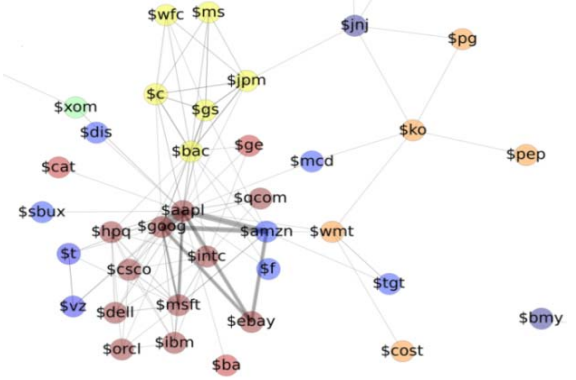


Figure 1. An example stock network.

2.2 Financial Networks for Stock

Financial network models study the correlations of stocks in a graph-based view (Tse et al., 2010; Mantegna, 1999; Vandewalle et al., 2001; Onnela et al., 2003; Bonanno et al., 2001). The usual approach is to measure the pairwise correlation of stocks' historical price series and then connect the stocks based on correlation strengths to build a correlation stock network (CSN).

However, our approach leverages social media posts on stock tickers. The rationale behind is that micro-blogging activities have been shown to be highly correlated with the stock market (Ruiz et al., 2012; Mao et al., 2012). It is more informative, granular to incorporate latest developments of the market as reflected in social media instead of relying on stocks' historical price.

3 Semantic Stock Network (SSN)

3.1 Construction of SSN

We collected five months (Nov. 2 2012 - Apr. 3 2013) of English tweets for a set of stocks in the Standard & Poor's 100 list via Twitter's REST API, using cash-tags as query keywords. For preprocessing, we removed tweets mentioning more than five continuous stock tickers as such tweets usually do not convey much meaning for our task. Finally, we obtained 629,977 tweets in total. Table 1 shows the top five most frequent stocks jointly mentioned with Apple Inc. in our dataset. Formally, we define the stock network as an undirected graph $G = \{V, E\}$. The node set V comprises of stocks, $e_{u,v} \in E$ stands for the edge between stock nodes u and v and the edge weight is the number of co-occurrences. On exploring the co-occurrence statistics in pilot studies, we set a minimum weight threshold of 400 to filter most non-informative edges. Figure 1 demonstrates a segment of the stock network constructed from our dataset.

$goog	$amzn	$ebay	$msft	$intc
43263	23266	14437	11891	2486

Table 1. co-occurrence statistics with $aapl.

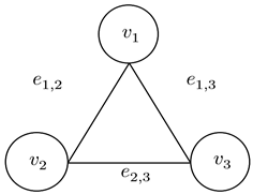


Figure 2. Tweet label design.

3.2 Semantic Topics over the Network

Figure 2 illustrates our annotation for each tweet. For a tweet, d with three cash-tags: $\{v_1, v_2, v_3\}$, we annotate d with the label set, $L_d = \{v_1, v_2, v_3, e_{1,2}, e_{1,3}, e_{2,3}\}$. ($e_{1,2}$ is "aapl_goog" if $v_1$is "aapl" and v_2 is "goog"). Then, the topic assignments of words in d are constrained to topics indexed by its label set, L_d. Given the annotations as labels, we use the Labeled LDA model (Ramage et al., 2009) to jointly learn the topics over nodes and edges. Labeled-LDA assumes that the set of topics are the distinct labels in a labeled set of documents, and each label corresponds to a unique topic. Similar to LDA (Blei et al., 2003), Labeled-LDA models each document as an admixture of latent topics and generates each word from a chosen topic. Moreover, Labeled-LDA incorporates supervision by simply constraining the model to use only those topics that correspond to a document's observed label set (Ramage et al., 2009). For model inference, we use collapsed Gibbs sampling (Bishop, 2006) and the symmetric Dirichlet Priors are set to: $\eta = 0.01, \alpha = 0.01$ as suggested in (Ramage et al., 2010). The Gibbs Sampler is given as:

$$p(z_i = k|z_{-i}) \sim \frac{N(d_i,k)_{-1} + \alpha}{N(d_i,*)_{-1} + |L_{d_i}|*\alpha} * \frac{N(k,w_i)_{-1} + \eta}{N(k,*)_{-1} + |V|*\eta} \quad (1)$$

where $N(d_i, k)$ is the number of words in d_i assigned to topic k, while $N(d_i,*)$ is the marginalized sum. $|L_{d_i}|$ is the size of label subset of d_i.

$N(k,w)$ is the term frequency of word w in topic k. $|V|$ is the vocabulary size. The subscript $_{-1}$ is used to exclude the count assignment of the current word w_i. The posterior on the document's topic distribution $\{\theta_{d,k}\}$ and topic's word distribution $\{\beta_{k,w}\}$ can be estimated as follows:

$$\theta_{d,k} = \frac{N(d_i,k)+\alpha}{N(d_i,*)+|L_{d_i}|*\alpha} \quad (2)$$

$$\beta_{k,w} = \frac{N(k,w_i)+\eta}{N(k,*)+|V|*\eta} \quad (3)$$

Later, parameters $\{\beta_{k,w}\}$ will be used to compute the sentiment score for topics.

3.3 Leveraging Sentiment over SSN for Stock Prediction

We define a lexicon based sentiment score in the form of opinion polarity for each node-indexed and edge-indexed topic as follows:

$$S(k) = \sum_{w=1}^{|V|} \beta_{k,w}\, l(w),\ S(k) \in [-1,1] \quad (4)$$

where $l(w)$ denotes the opinion polarity of word w. $\beta_{k,w}$ is the word probability of w in topic k (Eq.3). Based on an opinion lexicon O, $l(w)$ = +1 if $w \in O_{pos}$, $l(w)$ = -1 if $w \in O_{neg}$ and $l(w)$ = 0 otherwise. We use the opinion English lexicon contributed by Hu and Liu (2004).

Considering the inherent dynamics of both the stock markets and social sentiment, we organize the tweets into daily based sub-sets according to their timestamps to construct one SSN_t ($t \in [1,T]$) for each day. Then, we apply a Labeled LDA for each SSN_t and compute the sentiment scores for each SSN_t's nodes and edges. This yields a sentiment time series for the node, v, $\{S(v)_1, S(v)_2, \ldots, S(v)_T\}$ and for the edge, $e_{u,v}$, $\{S(e_{u,v})_1, S(e_{u,v})_2, \ldots, S(e_{u,v})_T\}$. We introduce a vector autoregression model (VAR) (Shumway and Stoffer, 2011) by regressing sentiment time series together with the stock price time series to predict the up/down movement of the stock's daily closing price.

As usual in time series analysis, the regression parameters are learned during a training phase and then are used for forecasting under sliding windows, i.e., to train in period $[t, t+w]$ and to predict on time $t+w+1$. Here the window size w refers to the number of days in series used in model training. A VAR model for two variables $\{x_t\}$ and $\{y_t\}$ can be written as:

$$y_t = \sum_{i=1}^{lag} (\vartheta_i^x x_{t-i} + \vartheta_i^y y_{t-i}) + \varepsilon_t \quad (5)$$

where $\{\varepsilon\}$ are white noises, $\{\vartheta\}$ are model parameters, and lag notes the time steps of historical information to use. In our experiment, $\{y_t\}$ is the target stock's price time series, $\{x_t\}$ is the covariate sentiment/price time series, and we will try $lag \in \{2,3\}$. We use the "dse" library in R language to fit our VAR model based on least square regression.

4 Experiments

4.1 Tweets in Relation to the Stock Market

Micro-blogging activities are well correlated with the stock market. Figure 3 shows us how the Twitter activities response to a report announcement of $aapl (Jan. 23 2013). The report was made public soon after the market closed at 4:00pm, while the tweets volume rose about two hours earlier and reached the peak at the time of announcement, then it arrived the second peak at the time near the market's next opening (9:30am). By further accumulating all days' tweet volume in our dataset as hourly based statistics, we plot the volume distribution in Figure 4. Again, we note that trading activities are well reflected by tweet activities. The volume starts to rise drastically two or three hours before the market opens, and then reaches a peak at 9:00pm. It drops during the lunch time and reaches the second peak around 2:00pm (after lunch). Above observations clearly show that market dynamics are discussed in tweets and the content in tweets' discussion very well reflects the fine-grained aspects of stock market trading, opening and closing.

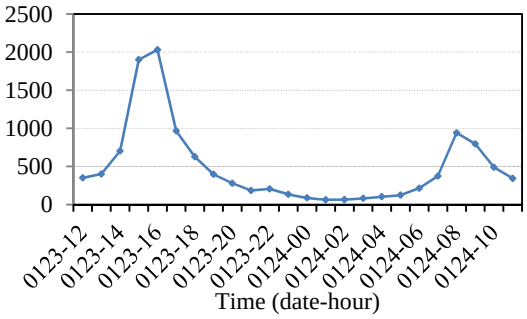


Figure 3. Tweet activity around $aapl's earnings report date on Jan. 23 2013.

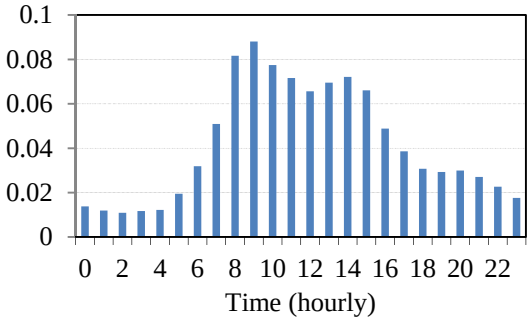


Figure 4. Tweet volume distribution in our data over hours averaged across each day.

4.2 Stock Prediction

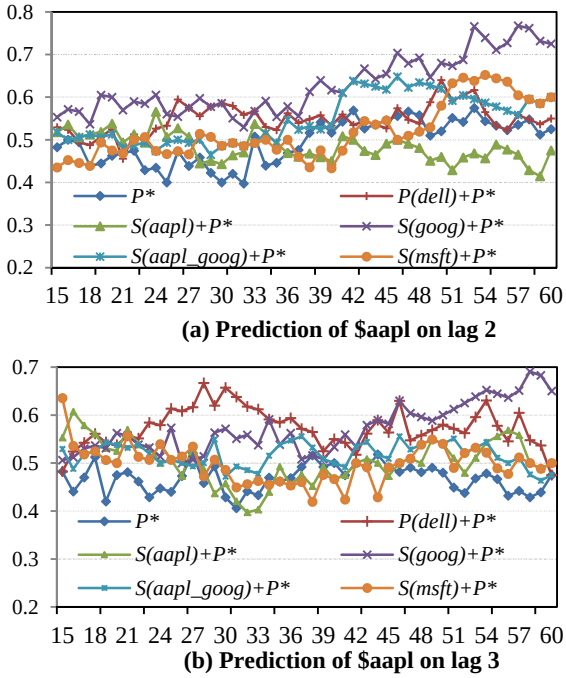
This section demonstrates the effectiveness of our SSN based approach for stock prediction. We leverage the sentiment time-series on two kinds of topics from SSN: 1). Node topic from the target stock itself, 2). Neighbor node/edge topics. We note that the price correlation stock network (CSN) (e.g., Bonanno et al., 2001; Mantegna, 1999) also defines neighbor relationships based on the Pearson's correlation coefficient (Tse et al., 2010) between pair of past price series (We get the stock dataset from Yahoo! Finance, between Nov. 2 2012 and Apr. 3 2013).

We build a two variables VAR model to predict the movement of a stock's daily closing price. One variable is the price time series of the target stock ($\{y_t\}$ in Eq.5); another is the covariate sentiment/price time series ($\{x_t\}$ in Eq.5). We setup two baselines according to the sources of the covariate time series as follows:

1. Covariate price time series from CSN, we try the price time series from the target stock's closest neighbor which takes the maximum historical price correlation in CSN.
2. With no covariate time series, we try the target stock's price only based on the univariate autoregression (AR) model.

To summarize, we try different covariate sentiment ($S(.)$) or price ($P(.)$) time series from SSN or CSN together with the target stock's price time series (P^*) to predict the movement of one day ahead price (P^{**}). The accuracy is computed based on the correctness of the predicted directions as follows, i.e., if the prediction P^{**} takes the same direction as the actual price value, we increment $\#(posPred)$ by 1, $\#(totalTest)$ is the total number of test.

$$Accuracy = \frac{\#(posPred)}{\#(totalTest)} \quad (6)$$



(a) Prediction of $aapl on lag 2

(b) Prediction of $aapl on lag 3

Figure 5. Prediction on $aapl. (x-axis is the training window size, y-axis is the prediction accuracy) with different covariate sources.

	Source	Lag = 2	Lag = 3
P^* only	self	0.49(0.57)	0.47(0.52)
CSN: $P(.)+P^*$	dell	0.55(0.64)	0.57(0.67)
SSN: $S(.)+P^*$	aapl	0.48(0.56)	0.50(0.61)
	goog	**0.62(0.78)**	**0.57(0.69)**
	aapl_goog	0.55(0.65)	0.52(0.56)
	msft	0.52(0.65)	0.54(0.61)

Table 2. Performance comparison of the average and best (in parentheses) prediction accuracies over all training window sizes for prediction on $aapl.

Figure 5 details the prediction of $aapl on different training window sizes of [15, 60] and lags. $\{S(aapl), S(goog), S(msft), S(aapl\_goog)\}$ are from SSN, $P(dell)$ is from CSN ($dell (Dell Inc.) takes the maximum price correlation score of 0.92 with $aapl), and $P^* = P(aapl)$ is the univariate AR model, using the target stock's price time series only. Table 2 further summarizes the performance comparison of different approaches reporting the average (and best) prediction accuracies over all time windows and different lag settings. Comparing to the univariate AR model (P^* only), we see that the sentiment based time-series improve performances significantly. Among SSN sentiment based approaches, the $S(goog)$ helps improve the performance mostly and gets the best accuracy of 0.78 on lag 2 and training window size of 53. On average, $S(goog)$ achieves a net gain over $S(aapl)$ in the range of 29% with lag 2 (0.62 = 1.29 x 0.48) and 14% with lag 3 (0.57 = 1.14 x 0.50). Also, $S(aapl\_goog)$ performs better than $S(aapl)$. The result indicates that $aapl's stock performance is highly influenced by its competitor. $P(dell)$ also performs well, but we will see relationships from CSN may not be so reliable.

We further summarize some other prediction cases in Table 3 to show how different covariate sentiment sources ($S(.)$) and price sources ($P(.)$) from their closest neighbor nodes help predict their stocks, which gives consistent conclusions. We compute the t-test for SSN based prediction accuracies against that of CSN or price only based approaches among all testing

window sizes ([15, 60]), and find that SSN based approaches are significantly (p-value < 0.001) better than others.

We note that tweet volumes of most S&P100 stocks are too small for effective model building, as tweets discuss only popular stocks, other stocks are not included due to their deficient tweet volume.

We make the following observations:

1. CSN may find some correlated stock pairs like $ebay and $amzn, $wmt and $tgt, but sometimes, it also produces pairs without real-world relationships like $tgt and $vz, $qcom and $pfe, etc. In contrast, SSN is built on large statistics of human recognition in social media, which is likely to be more reliable as shown.

2. Sentiment based approaches $\{S(\cdot)\}$ consistently perform better than all price based ones $\{P^*,\ P(\cdot)\}$. For $S(\cdot)$ based predictions, sentiment discovered from the target stock's closest neighbors in SSN performs best in general. This empirical finding dovetails with qualitative results in the financial analysis community (Mizik & Jacobson, 2003; Porter, 2008), where companies' market performances are more likely to be influenced by their competitors. But for Google, its stock market is not so much influenced by other companies (it gets the best prediction accuracy on $S(goog)$, i.e., the internal factor). It can be explained by Google Inc.'s relatively stable revenue structure, which is well supported by its leading position in the search engine market.

3. The business of offline companies like Target Corp. ($tgt) and Wal-Mart Stores Inc. ($wmt) are highly affected by online companies like $amzn. Although competition exists between $tag and $wmt, their performances seem to be affected more by a third-party like $amzn (In Table 3, $S(amzn)$ predicts the best for both). Not surprisingly, these offline companies have already been trying to establish their own online stores and markets.

5 Conclusion

This paper proposed to build a stock network from co-occurrences of ticker symbols in tweets. The properties of SSN reveal some close relationships between involved stocks, which provide good information for predicting stocks based on social sentiment. Our experiments show that SSN is more robust than CSN in capturing the neighbor relationships, and topic sentiments from close neighbors of a stock significantly improve the prediction of the stock market.

Acknowledgments

This work was supported in part by a grant from the National Science Foundation (NSF) under grant no. IIS-1111092).

Target	*lag*	P^* only	CSN: $P(.)+P^*$	SSN: $S(.)+P^*$		
goog			*dis*(0.96)	***goog***	*aapl*	*amzn*
	2	0.48(0.59)	0.53(0.60)	**0.59(0.65)**	0.44(0.53)	0.42(0.49)
	3	0.46(0.54)	0.53(0.62)	**0.56(0.67)**	0.50(0.59)	0.43(0.49)
amzn			*csco*(0.90)	*amzn*	*goog*	***msft***
	2	0.48(0.54)	0.48(0.55)	0.47(0.54)	0.57(0.66)	**0.60(0.68)**
	3	0.46(0.53)	0.49(0.53)	0.43(0.50)	0.55(0.63)	**0.57(0.66)**
ebay			*amzn*(0.81)	*ebay*	***amzn***	*goog*
	2	0.49(0.55)	0.51(0.57)	0.44(0.53)	**0.57(0.64)**	0.56(0.62)
	3	0.48(0.58)	0.49(0.54)	0.45(0.58)	**0.54(0.64)**	0.54(0.61)
tgt			*vz*(0.88)	*tgt*	*wmt*	***amzn***
	2	0.43(0.53)	0.43(0.54)	0.46(0.55)	0.49(0.56)	**0.49(0.59)**
	3	0.44(0.50)	0.40(0.53)	0.44(0.48)	0.41(0.48)	**0.48(0.54)**
wmt			*tgt*(0.86)	*wmt*	*tgt*	***amzn***
	2	0.53(0.59)	0.53(0.63)	0.52(0.61)	0.52(0.60)	**0.60(0.65)**
	3	0.53(0.64)	0.48(0.57)	0.55(0.66)	0.48(0.58)	**0.58(0.66)**
qcom			*pfe*(0.88)	*qcom*	*aapl*	***intc***
	2	0.53(0.6)	0.55(0.63)	0.57(0.61)	0.46(0.54)	**0.63(0.70)**
	3	0.54(0.61)	0.48(0.55)	0.56(0.65)	0.51(0.61)	**0.61(0.67)**

Table 3. Average and best (in parentheses) prediction accuracies (over window sizes of [15, 60]) of some other cases with different covariates, cell of *dis*(0.96) means "$dis" takes the maximum price correlation strength of 0.96 with "$goog" (similar for others in column CSN). The best performances are highlighted in **bold**.

References

Baldwin T., Cook P., Han B., Harwood A., Karunasekera S., and Moshtaghi M. 2012. A support platform for event detection using social intelligence. In Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL '12). Association for Computational Linguistics, Stroudsburg, PA, USA, 69-72.

Bishop C.M. 2006. Pattern Recognition and Machine Learning. Springer.

Blei D., NG A., and Jordan M. 2003. Latent Dirichlet allocation. Journal of Machine Learning Research 3:993-1022.

Bollen J., Mao H., and Zeng X.J. 2011. Twitter mood predicts the stock market. Journal of Computer Science 2(1):1-8.

Bonanno G., Lillo F., and Mantegna R.N. 2001. High-frequency cross-correlation in a set of stocks, Quantitative Finance, Taylor and Francis Journals, vol. 1(1), 96-104.

Cohen J., Cohen P., West S.G., and Aiken L.S. 2003. Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences, (3rd ed.) Hillsdale, NJ: Lawrence Erlbaum Associates.

Diao Q., Jiang J., Zhu F., and Lim E.P. 2012. Finding bursty topics from microblogs. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1 (ACL '12), Vol. 1. Association for Computational Linguistics, Stroudsburg, PA, USA, 536-544.

Gao W., Li P., and Darwish K. 2012. Joint topic modeling for event summarization across news and social media streams. CIKM 2012: 1173-1182

Hu M. and Liu B. 2004. Mining and summarizing customer reviews. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 22-25. Seattle, Washington (KDD-2004).

Jo Y. and Oh A. 2011. Aspect and sentiment unification model for online review analysis. In ACM Conference in Web Search and Data Mining (WSDM-2011).

Liu B. 2012. Sentiment analysis and opinion mining. Morgan & Claypool Publishers.

Liu Y., Huang X., An A., and Yu X. 2007. ARSA: a sentiment-aware model for predicting sales performance using blogs. In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 607-614. ACM, New York, NY.

Ma Z., Sun A., and Cong G. 2013. On predicting the popularity of newly emerging hashtags in Twitter. In Journal of the American Society for Information Science and Technology, 64(7): 1399-1410 (2013)

Mantegna R. 1999. Hierarchical structure in financial markets, The European Physical Journal B - Condensed Matter and Complex Systems, Springer, vol. 11(1), pages 193-197, September.

Mao Y., Wei W., Wang B., and Liu B. 2012. Correlating S&P 500 stocks with Twitter data. In Proceedings of the First ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research (HotSocial '12). ACM, New York, NY, USA, 69-72

Mei Q., Ling X., Wondra M., Su H., and Zhai C. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In Proceedings of International Conference on World Wide Web (WWW-2007).

Mizik N. and Jacobson R. 2003. Trading off between value creation and value appropriation: The financial implications of shifts in strategic emphasis. Journal of Marketing, 63-76.

Onnela J.P., Chakraborti A., and Kaski K. 2003. Dynamics of market correlations: taxonomy and portfolio analysis, Phys. Rev. E 68, 056110.

Pang B. and Lee L. 2008. Opinion Mining and Sentiment Analysis. Now Publishers Inc.

Porter M.E. 2008. The Five Competitive Forces That Shape Strategy.HBR, Harvard Business Review.

Ramage D., Dumais S.T., and Liebling D. 2010. Characterizing microblogging using latent topic models. In Proceedings of ICWSM 2010.

Ramage D., Hall D., Nallapati R., and Manning C.D. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009).

Ramage D., Manning C.D., and Dumais S.T. 2011. Partially labeled topic models for interpretable text mining. In Proceedings of KDD 2011

Ruiz E.J., Hristidis V., Castillo C., Gionis A., and Jaimes A. 2012. Correlating financial time series with micro-blogging activity. In Proceedings of the fifth ACM international conference on Web search and data mining, pp. 513-522. ACM Press, NY (WSDM-2012).

Shumway R.H. and Stoffer D.S. 2011. Time Series Analysis and Its Applications: With R Examples, 3rd ed.

Si J., Mukherjee A., Liu B., Li Q., Li H., and Deng X. 2013. Exploiting Topic based Twitter Sentiment for Stock Prediction. In Proceedings of the 51st

Annual Meeting of the Association for Computational Linguistics. ACL'13, Sofia, Bulgaria, 24-29.

Tse C.K., Liu J., and Lau F.C.M. 2010. A network perspective of the stock market, Journal of Empirical Finance. 17(4): 659-667.

Vandewalle N., Brisbois F., and Tordoir X. 2001. Self-organized critical topology of stock markets, Quantit. Finan., 1, 372–375.

Wang X., Wei F., Liu X., Zhou M., and Zhang M. 2011. Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. CIKM 2011: 1031-1040

Weng J. and Lee B.S. 2011. Event Detection in Twitter. In Proceedings of the International AAAI Conference on Weblogs and Social Media 2011.

Weng J.Y., Yang C.L., Chen B.N., Wang Y.K., and Lin S.D. 2011. IMASS: An Intelligent Microblog Analysis and Summarization System. ACL (System Demonstrations) 2011: 133-138.

Developing Age and Gender Predictive Lexica over Social Media

Maarten Sap[1] **Gregory Park**[1] **Johannes C. Eichstaedt**[1] **Margaret L. Kern**[1]
David Stillwell[3] **Michal Kosinski**[3] **Lyle H. Ungar**[2] and **H. Andrew Schwartz**[2]
[1]Department of Psychology, University of Pennsylvania
[2]Computer & Information Science, University of Pennsylvania
[3]Psychometrics Centre, University of Cambridge
maarten@sas.upenn.edu

Abstract

Demographic lexica have potential for widespread use in social science, economic, and business applications. We derive predictive lexica (words and weights) for age and gender using regression and classification models from word usage in Facebook, blog, and Twitter data with associated demographic labels. The lexica, made publicly available,[1] achieved state-of-the-art accuracy in language based age and gender prediction over Facebook and Twitter, and were evaluated for generalization across social media genres as well as in limited message situations.

1 Introduction

Use of social media has enabled the study of psychological and social questions at an unprecedented scale (Lazer et al., 2009). This allows more data-driven discovery alongside the typical hypothesis-testing social science process (Schwartz et al., 2013b). Social media may track disease rates (Paul and Dredze, 2011; Google, 2014), psychological well-being (Dodds et al., 2011; De Choudhury et al., 2013; Schwartz et al., 2013a), and a host of other behavioral, psychological and medical phenomena (Kosinski et al., 2013).

Unlike traditional hypothesis-driven social science, such large-scale social media studies rarely take into account—or have access to—age and gender information, which can have a major impact on many questions. For example, females live almost five years longer than males (cdc, 2014; Marengoni et al., 2011). Men and women, on average, differ markedly in their interests and work preferences (Su et al., 2009). With age, personalities gradually change, typically becoming less open to experiences but more agreeable and conscientious (McCrae et al., 1999). Additionally, social media language varies by age (Kern et al., 2014; Pennebaker and Stone, 2003) and gender (Huffaker and Calvert, 2005). Twitter may have a male bias (Mislove et al., 2011), while social media in general skew towards being young and female (pew, 2014).

Accessible tools to predict demographic variables can substantially enhance social media's utility for social science, economic, and business applications. For example, one can post-stratify population-level results to reflect a representative sample, understand variation across age and gender groups, or produce personalized marketing, services, and sentiment recommendations; a movie may be generally disliked, except by people in a certain age group, whereas a product might be used primarily by one gender.

This paper describes the creation of age and gender predictive lexica from a dataset of Facebook users who agreed to share their status updates and reported their age and gender. The lexica, in the form of words with associated weights, are derived from a penalized linear regression (for continuous valued age) and support vector classification (for binary-valued gender). In this modality, the lexica are simply a transparent and portable means for distributing predictive models based on words. We test generalization and adapt the lexica to blogs and Twitter, plus consider situations when limited messages are available. In addition to use in the computational linguistics community, we believe the lexicon format will make it easier for social scientists to leverage data-driven models where manually created lexica currently dominate[2] (Dodds et al., 2011; Tausczik and Pennebaker, 2010).

2 Related Work

Online behavior is representative of many aspects of a user's demographics (Pennacchiotti and Popescu, 2011; Rao et al., 2010). Many studies have used linguistic cues (such as ngrams) to determine if someone belongs to a certain age group, be it on Twitter or another social media platform (Al Zamal et al., 2012; Argamon et al., 2009; Nguyen et al., 2013; Rangel and Rosso, 2013). Gender prediction has been studied across blogs (Burger and Henderson, 2006; Goswami et al., 2009), Yahoo! search queries (Jones et al., 2007), and Twitter (Burger et al., 2011; Nguyen et al., 2013; Liu and Ruths, 2013; Rao et al., 2010). Because Twitter does not make gender or age available, such work infers gender and age by leveraging profile information, such as gender-discriminating names or crawling for links to publicly available data (e.g. Burger et al.,

[1]download at http://www.wwbp.org/data.html

[2]The LIWC lexicon, derived manually based on psychological theory, (Pennebaker et al., 2001) had 1136 citations in 2013 alone.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1146–1151,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

2011).

While many studies have examined prediction of age or gender, none (to our knowledge) have released a model to the public, much less in the form of a lexicon. Additionally, most works in age prediction classify users into bins rather than predicting a continuous real-valued age as we do (exceptions: Nguyen et al., 2013; Jones et al., 2007). People have also used online media to infer other demographic-like attributes such as native language (Argamon et al., 2009), origin (Rao et al., 2010), and location (Jones et al., 2007). An approach similar to the one presented here could be used to create lexica for any of these outcomes.

While lexica are not often used for demographics, data-driven lexicon creation over social media has been well studied for sentiment, in which univariate techniques (e.g. *point-wise mutual information*) dominate[3]. For example, Taboada et al. (2011) expanded an initial lexicon by adding on co-occurring words. More recently, Mohammad's sentiment lexicon (Mohammad et al., 2013) was found to be the most informative feature for the top system in the SemEval-2013 social media sentiment analysis task (Wilson et al., 2013). Approaches like point-wise mutual information take a univariate view on words–i.e. the weight given to one feature (word) is not affected by other features. Since language is highly collinear, we take a multivariate lexicon development approach, which takes covariance into account (e.g. someone who mentions 'hair' often is more likely to mention 'brushing', 'style', and 'cut'; weighting these words in isolation might "double-count" some information).

3 Method

Primary data. Our primary dataset consists of Facebook messages from users of the MyPersonality application (Kosinski and Stillwell, 2012). Messages were posted between January 2009 and October 2011. We restrict our analysis to those Facebook users meeting certain criteria: they must indicate English as a primary language, have written at least 1,000 words in their status updates, be younger than 65 years old (data beyond this age becomes very sparse), and indicate their gender and age. This resulted in a dataset of N = 75,394 users, who wrote over 300 million words collectively. We split our sample into training and test sets. Our primary *test set* consists of a 1,000 randomly selected Facebook users, while the *training set* that we used for creating the lexica was a subset (N = 72,874) of the remaining users.

Additional data To evaluate our predictive lexica in differing situations, we utilize three additional datasets: stratified Facebook data, blogs, and tweets. The stratified Facebook data (exclusively used for testing) consists of equal proportions of 1,520 males and females across 12 4-year age bins starting at 13 and ending at 60.[4] This roughly matchs the size of the main *test set*.

Seeking out-of-domain data, we downloaded age and gender annotated blogs from 2004 (Schler et al., 2006) (also used in Goswami et al., 2009) and gender labeled tweets (Volkova et al., 2013). Limiting the sample to users who wrote at least 1000 words, the total number of bloggers is 15,006, of which 50.6% are female and only 15% are over 27 (reflecting the younger population standard in social media). From this we use a randomly selected 1,000 bloggers as a blogger test set and the remaining 14,006 bloggers for training. Similarly for the Twitter dataset, we use 11,000 random gender-only annotated users, in which 51.9% are female. We again randomly select 1,000 users as a test set for gender prediction and use the remaining 10,000 for training.

3.1 Lexicon Creation

We present a method of weighted lexicon creation by using the coefficients from linear multivariate regression and classification models. Before delving into the creation process, consider that a weighted lexicon is often applied as the sum of all weighted word relative frequencies over a document:

$$usage_{lex} = \sum_{word \in lex} w_{lex}(word) * \frac{freq(word, doc)}{freq(*, doc)}$$

where $w_{lex}(word)$ is the lexicon (lex) weight for the $word$, $freq(word, doc)$ is frequency of the word in the document (or for a given user), and $freq(*, doc)$ is the total word count for that document (or user).

Further consider how one applies linear multivariate models in which the goal is to optimize feature coefficients that best fit the continuous outcome (regression) or separate two classes (classification):

$$y = (\sum_{f \in features} w_f * x_f) + w_0$$

where x_f is the value for a feature (f), w_f is the feature coefficient, and w_0 is the intercept (a constant fit to shift the data such that it passes through the origin). In the case of regression, y is the outcome value (e.g. age) while in classification y is used to separate classes (e.g. $>= 0$ is female, < 0 is male). If all features are word relative frequencies ($\frac{freq(word,doc)}{freq(*,doc)}$) then many multivariate modeling techniques can simply be seen as learning a weighted lexicon plus an intercept[5].

[3]Note that the point-wise information-derived sentiment lexica are often used as features in a supervised model, essentially dimensionally reducing a large set of words into positive and negative sentiment, while our lexica represent the predictive model itself.

[4]65 females and 65 males in each of the first 11 bins: [13,16], [17,20], ..., [53, 56]; the last bin ([57, 60]) contained 45 males and 45 females. The [61.64] bin was excluded as it was much smaller.

[5]included in the lexicon distribution

	age						gender			
model\corpus	**randFB**		**stratFB**		**randBG**		**randFB**	**stratFB**	**randBG**	**randT**
	r	mae	r	mae	r	mae	acc	acc	acc	acc
baseline	0	6.14	0	11.62	0	6.11	.617	.500	.508	.518
FB_{lex}	.835	3.40	.801	6.94	.710	5.76	.917	.913	.774	.856
BG_{lex}	.664	4.26	.656	11.39	.768	3.63	.838	.803	.824	.834
FB+BG_{lex}	.831	3.42	.795	7.06	.762	3.76	.913	.909	.822	.858
T_{lex}							.816	.820	.763	.889
FB+BG+T_{lex}							.919	.910	.820	.900

Table 1: Prediction accuracies for age (Pearson correlation coefficient(r); mean absolute error (mae) in years) and gender (accuracy %). Baseline for age is mean age of training sample; for gender, it is the most frequent class (female). Lexica tested include those derived from Facebook (FB_{lex}), blogs (BG_{lex}), and Twitter (T_{lex}). We evaluate over a random Facebook sample (randFB), a stratified Facebook sample (stratFB), a random blogger sample (randBG), and a random twitter sample (randT). All results were a significant ($p < 0.001$) improvement over the baseline.

In practice, we learn our 1gram coefficients (i.e. lexicon weights) from ridge regression (Hoerl and Kennard, 1970) for age (continuous variable) and from support vector classification (Fan et al., 2008) for gender (binary variable). Ridge regression uses an L2 ($\alpha||\beta||^2$) penalization to avoid overfitting (Hoerl and Kennard, 1970). Although some words no doubt have a non-linear relationship with age (e.g., 'fiance' peaks in the 20s), we still find high accuracy from a linear model (see Table 1) and it allows for a distribution of the model in the accessible form of a lexicon. For gender prediction, we use an SVM with a linear kernel with L1 penalization ($\alpha||\beta||_1$) (Tibshirani, 1996). Because the L1 penalization zeros-out many coefficients, it has the added advantage of effectively reducing the size of the lexica. Using the training data, we test a variety algorithms including the lasso, elastic net regression, and L2 penalized SVMs in order to decide which learning algorithms to use.

To extract the words (1grams) to use as features and which make up lexica, we use the Happier Fun Tokenizer,[6] which handles social media content and markup such as emoticons or hashtags. For our main user-level models, word usage is aggregated as the relative frequency ($\frac{freq(word,user)}{freq(*,user)}$). Due to the sparse and large vocabulary of social media data, we limit the 1grams to those used by at least 1% of users.

4 Evaluation

We evaluate our predictive lexica across held-out user data. First, we see how well lexica derived from Facebook users predict a random set of additional users. Then, we explore generalization of the models in various other settings: on a stratified Facebook test sample, blogs, and Twitter. Finally, we compare lexica fit to a restricted number of messages per user.

Results of our evaluation over Facebook users are shown in Table 1 (randFB columns). Accuracies for age are reported as Pearson correlation coefficients (r) and mean absolute errors (mae), measured in years. For gender, we use an accuracy % (number-correct over test-size). As baselines, we use the mean for age (23.0 years old) and the most frequent class (female) for gender. We see that for both age and gender, accuracies are substantially higher than the baseline. These accuracies were just below with no significant difference previous state-of-the-art results (Schwartz et al., 2013; r = 0.84 for age and 91.9% accuracy for gender). [7]

Because of the nature of our datasets (the Facebook data is private) and task (*user-level* predictions), comparable previous studies are nearly nonexistent. Nonetheless, the Twitter data was a random subset of users based on the (Burger et al., 2011) dataset excluding non-English tweets, making it somewhat comparable. In this case, the lexica outperformed previous results for gender prediction of Twitter users, which ranged from 75.5% to 87% (Burger et al., 2011; Ciot et al., 2013; Liu and Ruths, 2013; Al Zamal et al., 2012). However, the lexica were unable to match the 92.0% accuracy Burger et al. (2011) achieved when using profile information in addition to text. No other similar studies — to the best of our knowledge — have been conducted.

Application in other settings. While Facebook is the ideal setting to apply our lexica, we hope that they generalize to other situations. To evaluate their utility in other settings, we first tested them over a gender and age stratified Facebook sample. Our random sample, like all of Facebook, is biased toward the young; this stratified test sample contains equal numbers of males and females, ages 13 to 60. Next, we use the lexica to predict data from other domains: blogs (Schler et al., 2006) and Twitter (Volkova et al., 2013). In this case, our goal was to account for the content and stylistic variation that may be specific to Facebook.

[6] downloaded from http://www.wwbp.org/data.html

[7] Adding 2 and 3-grams increases the performance of our model (r = 0.85, 92.7%), just above our previous results (Schwartz et al., 2013b). However, with the accessibility of single word lexica in mind, this current work focuses on features based entirely on 1grams.

# Msgs:	**all**	**100**	**20**	**5**	**1**
age	**.831**	.820	.688	.454	.156
gender	**.919**	.901	.796	.635	.554

Table 2: Prediction accuracies for age (Pearson correlation) and gender (accuracy %) when reducing the number of messages from each user.

Results over these additional datasets are shown in Table 1 (stratFB, randBG, and randT columns). The performance decreases as expected since these datasets have differing distributions, but it is still substantially above mean and most frequent class baselines on the stratified dataset. Over blogs and Twitter, both age and gender prediction accuracies drop to a greater degree (when only using the Facebook-trained models), suggesting stylistic or content differences between the domains. However, when using lexica created with data from across multiple domains, the results in Facebook, blogs, and Twitter remain in line with results from models created specifically over their respective domains. In light of this result, we release the FB+BG age & FB+BG+T gender models as lexica (available at www.wwbp.org/data.html).

Limiting messages per user. As previously noted, some applications of demographic estimation require predictions over more limited messages. We explore the accuracy of user-level age and gender predictions as the number of messages per user decreases in Table 2. For these tests we used the FB+BG age & FB+BG+T gender lexica. Confirming findings by Van Durme (2012), the fewer posts one has for each user, the less accurate the gender and age predictions. Still, given the average user posted 205 messages, it seems that not all messages from a user are necessary to make a decent inference on their age and gender. Future work may explore models developed specifically for these limited situations.

5 Conclusion

We created publicly available lexica (words and weights) using regression and classification models over language usage in social media. Evaluation of the lexica over Facebook yielded accuracies in line with state-of-the-art age ($r = 0.831$) and gender (91.9% accuracy) prediction. By deriving the lexica from Facebook, blogs, and Twitter, we found the predictive power generalized across all three domains with little sacrifice to any one domain, suggesting the lexica may be used in additional social media domains. We also found the lexica maintain reasonable accuracy when writing samples were somewhat small (e.g. 20 messages) but other approaches may be best when dealing with more limited data.

Given that manual lexica are already extensively employed in social sciences such as psychology, economics, and business, using lexical representations of data-driven models allows the utility of our models to extend beyond the borders of the field of NLP.

Acknowledgement

Support for this work was provided by the Templeton Religion Trust and by Martin Seligman of the University of Pennsylvania's Positive Psychology Center.

References

Faiyaz Al Zamal, Wendy Liu, and Derek Ruths. 2012. Homophily and latent attribute inference: Inferring latent attributes of twitter users from neighbors. In *ICWSM*.

Shlomo Argamon, Moshe Koppel, James W Pennebaker, and Jonathan Schler. 2009. Automatically profiling the author of an anonymous text. *Communications of the ACM*, 52(2):119–123.

John D Burger and John C Henderson. 2006. An exploration of observable features related to blogger age. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, pages 15–20.

John D Burger, John C Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1301–1309. Association for Computational Linguistics.

2014. Faststats: How healthy are we. `http://www.cdc.gov/nchs/fastats/healthy.htm`. Accessed on March 12, 2014.

Morgane Ciot, Morgan Sonderegger, and Derek Ruths. 2013. Gender inference of twitter users in non-english contexts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, Wash*, pages 18–21.

Munmun De Choudhury, Scott Counts, and Eric Horvitz. 2013. Predicting postpartum changes in emotion and behavior via social media. In *Proceedings of the 2013 ACM annual conference on Human factors in computing systems*, pages 3267–3276. ACM.

Peter Sheridan Dodds, Kameron Decker Harris, Isabel M Kloumann, Catherine A Bliss, and Christopher M Danforth. 2011. Temporal patterns of happiness and information in a global social network: Hedonometrics and twitter. *PloS one*, 6(12):e26752.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Inc. Google. 2014. Google flu trends. `http://www.google.org/flutrends`. Accessed on March 12, 2014.

Sumit Goswami, Sudeshna Sarkar, and Mayur Rustagi. 2009. Stylometric analysis of bloggers age and gender. In *Third International AAAI Conference on Weblogs and Social Media*.

Arthur E Hoerl and Robert W Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

David A Huffaker and Sandra L Calvert. 2005. Gender, identity, and language use in teenage blogs. *Journal of Computer-Mediated Communication*, 10(2):00–00.

Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. 2007. I know what you did last summer: query logs and user privacy. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 909–914. ACM.

Margaret L Kern, Johannes C Eichstaedt, H Andrew Schwartz, Gregory Park, Lyle H Ungar, David J Stillwell, Michal Kosinski, Lukasz Dziurzynski, and Martin EP Seligman. 2014. From sooo excited!!! to so proud: Using language to study development. *Developmental psychology*, 50(1):178–188.

Michal Kosinski and David J Stillwell. 2012. mypersonality project. *http://www.mypersonality.org/wiki/*.

Michal Kosinski, David J Stillwell, and Thore Graepel. 2013. Private traits and attributes are predictable from digital records of human behavior. volume 110, pages 5802–5805. National Acad Sciences.

David Lazer, Alex Pentland, Lada Adamic, Sinan Aral, Albert-Laszlo Barabasi, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, Tony Jebara, Gary King, Michael Macy, Deb Roy, and Marshall Van Alstyne. 2009. Computational social science. *Science*, 323(5915):721–723.

Wendy Liu and Derek Ruths. 2013. Whats in a name? using first names as features for gender inference in twitter. In *Analyzing Microtext: 2013 AAAI Spring Symposium*.

Alessandra Marengoni, Sara Angleman, René Melis, Francesca Mangialasche, Anita Karp, Annika Garmen, Bettina Meinow, and Laura Fratiglioni. 2011. Aging with multimorbidity: a systematic review of the literature. *Ageing research reviews*, 10(4):430–439.

Robert R McCrae, Paul T Costa, Margarida Pedroso de Lima, António Simões, Fritz Ostendorf, Alois Angleitner, Iris Marušić, Denis Bratko, Gian Vittorio Caprara, Claudio Barbaranelli, et al. 1999. Age differences in personality across the adult life span: parallels in five cultures. *Developmental Psychology*, 35(2):466–477.

Alan Mislove, Sune Lehmann, Yong-Yeol Ahn, Jukka-Pekka Onnela, and J Niels Rosenquist. 2011. Understanding the demographics of twitter users. *ICWSM*, 11:5th.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*.

Dong Nguyen, Rilana Gravel, Dolf Trieschnigg, and Theo Meder. 2013. how old do you think i am?: A study of language and age in twitter. In *Proceedings of the Seventh International AAAI Conference on Weblogs and Social Media*.

Michael J Paul and Mark Dredze. 2011. You are what you tweet: Analyzing twitter for public health. In *ICWSM*.

Marco Pennacchiotti and Ana-Maria Popescu. 2011. A machine learning approach to twitter user classification. In *ICWSM*.

James W Pennebaker and Lori D Stone. 2003. Words of wisdom: language use over the life span. *Journal of Personality and Social Psychology*, 85(2):291–301.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. 71:2001.

2014. Social networking fact sheet. `http://www.pewinternet.org/fact-sheets/social-networking-fact-sheet/`. Accessed on August 26, 2014.

Francisco Rangel and Paolo Rosso. 2013. Use of language and author profiling: Identification of gender and age. *Natural Language Processing and Cognitive Science*, page 177.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying latent user attributes in twitter. In *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, pages 37–44. ACM.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. 2006. Effects of age and gender on blogging. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, volume 6, pages 199–205.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Megha Agrawal, Gregory J Park, Shrinidhi K Lakshmikanth, Sneha Jha, Martin EP Seligman, Lyle Ungar, et al. 2013a. Characterizing geographic variation in well-being using tweets. In *ICWSM*.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David J Stillwell, Martin EP Seligman, et al. 2013b. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS one*, 8(9):e73791.

Rong Su, James Rounds, and Patrick Ian Armstrong. 2009. Men and things, women and people: a meta-analysis of sex differences in interests. *Psychological Bulletin*, 135(6):859–884.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Benjamin Van Durme. 2012. Streaming analysis of discourse participants. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 48–58. Association for Computational Linguistics.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *Proceedings of the 2013 Conference on Empirical Methods on Natural Language Processing*.

Theresa Wilson, Zornitsa Kozareva, Preslav Nakov, Sara Rosenthal, Veselin Stoyanov, and Alan Ritter. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Proceedings of the International Workshop on Semantic Evaluation, SemEval*, volume 13.

Dependency Parsing for Weibo: An Efficient Probabilistic Logic Programming Approach

William Yang Wang, Lingpeng Kong, Kathryn Mazaitis, William W. Cohen
Language Technologies Institute & Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA.
{yww,lingpenk,krivard,wcohen}@cs.cmu.edu

Abstract

Dependency parsing is a core task in NLP, and it is widely used by many applications such as information extraction, question answering, and machine translation. In the era of social media, a big challenge is that parsers trained on traditional newswire corpora typically suffer from the domain mismatch issue, and thus perform poorly on social media data. We present a new GFL/FUDG-annotated Chinese treebank with more than 18K tokens from Sina Weibo (the Chinese equivalent of Twitter). We formulate the dependency parsing problem as many small and parallelizable arc prediction tasks: for each task, we use a programmable probabilistic first-order logic to infer the dependency arc of a token in the sentence. In experiments, we show that the proposed model outperforms an off-the-shelf Stanford Chinese parser, as well as a strong MaltParser baseline that is trained on the same in-domain data.

1 Introduction

Weibo, in particular Sina Weibo[1], has attracted more than 30% of Internet users (Yang et al., 2012), making it one of the most popular social media services in the world. While Weibo posts are abundantly available, NLP techniques for analyzing Weibo posts have not been well-studied in the past.

Syntactic analysis of Weibo is made difficult for three reasons: first, in the last few decades, Computational Linguistics researchers have primarily focused on building resources and tools using standard English newswire corpora[2], and thus, there are fewer resources in other languages in general. Second, microblog posts are typically short, noisy (Gimpel et al., 2011), and can be considered as a "dialect", which is very different from news data. Due to the differences in genre, part-of-speech taggers and parsers trained on newswire corpora typically fail on social media texts. Third, most existing parsers use language-independent standard features (McDonald et al., 2005), and these features may not be optimal for Chinese (Martins, 2012). To most of the application developers, the parser is more like a blackbox, which is not directly programmable. Therefore, it is non-trivial to adapt these generic parsers to language-specific social media text.

In this paper, we present a new probabilistic dependency parsing approach for Weibo, with the following contributions:

- We present a freely available Chinese Weibo dependency treebank[3], manually annotated with more than 18,000 tokens;
- We introduce a novel probabilistic logic programming approach for dependency arc prediction, making the parser directly programmable for theory engineering;
- We show that the proposed approach outperforms an off-the-shelf dependency parser, as well as a strong baseline trained on the same in-domain data.

In the next section, we describe existing work on dependency parsing for Chinese. In Section 3, we present the new Chinese Weibo Treebank to the research community. In Section 4, we introduce the proposed efficient probabilistic programming approach for parsing Weibo. We show the experimental results in Section 5, and conclude in Section 6.

[1] http://en.wikipedia.org/wiki/Sina_Weibo

[2] For example, Wall Street Journal articles are used for building the Penn Treebank (Marcus et al., 1993).

[3] http://www.cs.cmu.edu/~yww/data/WeiboTreebank.zip

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1152–1158,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

2 Related Work

Chinese dependency parsing has attracted many interests in the last fifteen years. Bikel and Chiang (2000; 2002) are among the first to use Penn Chinese Tree Bank for dependency parsing, where they adapted Xia's head rules (Xia, 1999). An important milestone for Chinese dependency parsing is that, a few years later, the CoNLL shared task launched a track for multilingual dependency parsing, which also included Chinese (Buchholz and Marsi, 2006; Nilsson et al., 2007). These shared tasks soon popularized Chinese dependency parsing by making datasets available, and there has been growing amount of literature since then (Zhang and Clark, 2008; Nivre et al., 2007; Sagae and Tsujii, 2007; Che et al., 2010; Carreras, 2007; Duan et al., 2007).

Besides the CoNLL shared tasks, there are also many interesting studies on Chinese dependency parsing. For example, researchers have studied case (Yu et al., 2008) and morphological (Li and Zhou, 2012) structures for learning a Chinese dependency parser. Another direction is to perform joint learning and inference for POS tagging and dependency parsing (Li et al., 2011; Hatori et al., 2011; Li et al., 2011; Ma et al., 2012). In recent years, there has been growing interests in dependency arc prediction in Chinese (Che et al., 2014), and researchers have also investigated character-level Chinese dependency parsing (Zhang et al., 2014). However, even though the above methods all have merits, the results are reported only on standard newswire based Chinese Treebank (e.g. from People's Daily (Liu et al., 2006)), and it is unclear how they would perform on Weibo data.

To the best of our knowledge, together with the recent study on parsing tweets (Kong et al., 2014), we are among the first to study the problem of dependency parsing for social media text.

3 The Chinese Weibo Treebank

We use the publicly available μtopia dataset (Ling et al., 2013) for dependency annotation. An interesting aspect of this Weibo dataset is that, besides the Chinese posts, it also includes a copy of the English translations. This allows us to observe some interesting phenomena that mark the differences of the two languages. For example:

- Function words are more frequently used in English than in Chinese. When examining this English version of the Weibo corpus for the total counts of the word "the", there are 2,084 occurrences in 2,003 sentences. Whereas in Chinese, there are only 52 occurrences of the word "the" out of the 2,003 sentences.

- The other interesting thing is the position of the head. In English, the head of the tree occurs more frequent on the left-to-middle of the sentence, while the distribution of the head is more complicated in Chinese. This is also verified from the parallel Weibo data.

- Another well-known issue in Chinese is that Chinese is a pro-drop topical language. This is extremely prominent in the short text, which clearly creates a problem for parsing. For example, in the Chinese Weibo data, we have observed the sentence in Figure 1.

想吃 鸡蛋 就 得 忍受 鸡叫

(If you) Want to eat chicken eggs,
(you) have to bear the chicken sound

Figure 1: An example of pro-drop phenomenon from the Weibo data.

To facilitate the annotation process, we first preprocess the Weibo posts using the Stanford NLP pipeline, including a Chinese Word Segmenter (Tseng et al., 2005) and a Chinese Part-of-Speech tagger (Toutanova and Manning, 2000). Two native speakers of Chinese with strong linguistic backgrounds have annotated the dependency relations from 1,000 posts of the μtopia dataset, using the FUDG (Schneider et al., 2013) and GFL annotation tool (Mordowanec et al., 2014). The annotators communicate regularly during the annotation process, and a coding manual that relies majorly on the Stanford Dependencies (Chang et al., 2009) is designed. The annotation process has two stages: in the first stage, we rely on the word segmentation produced by the segmenter, and produce a draft version of the treebank; in the second stage, the annotators actively discuss the difficult cases to reach agreements, manually correct the mis-segmented word tokens, and revise the annotations of the tricky cases. The final inter-annotator agreement rate on a randomly-selected subset of 373 tokens in this

treebank is 82.31%.

Fragmentary Unlabeled Dependency Grammar (FUDG) is a newly proposed flexible framework that offers a relative easy way to annotate the syntactic structure of text. Beyond the traditional tree view of dependency syntax in which the tokens of a sentence form nodes in a tree, FUDG also allows the annotation of additional lexical items such as multiword expressions. It provides special devices for coordination and coreference; and facilitates underspecified (partial) annotations where producing a complete parse would be difficult. Graph Fragment Language (GFL) is an implementation of unlabeled dependency annotations in the FUDG framework, which fully supports Chinese, English and other languages. The training set of our Chinese Weibo Treebank[4] includes 14,774 tokens, while the development and test sets include 1,846 and 1,857 tokens respectively.

4 A Programmable Parser with Personalized PageRank Inference

A key problem in multilingual dependency parsing is that generic feature templates may not work well for every language. For example, Martins (2012) shows that for Chinese dependency parsing, when adding the generic grandparents and siblings features, the performance was worse than using the standard bilexical, unilexical, and part-of-speech features. Unfortunately, for many parsers such as Stanford Chinese Parser (Levy and Manning, 2003) and MaltParser (Nivre et al., 2007), it is very difficult for programmers to specify the feature templates and inference rules for dependency arc prediction.

In this work, we present a Chinese dependency parsing method for Weibo, based on efficient probabilistic first-order logic programming (Wang et al., 2013). The advantage of probabilistic programming for parsing is that, software engineers can simply conduct theory engineering, and optimize the performance of the parser for a specific genre of the target language. Recently, probabilistic programming approaches (Goodman et al., 2012; Wang et al., 2013; Lloyd et al., 2014) have demonstrated its efficiency and effectiveness in many areas such as information extraction (Wang et al., 2014), entity linking, and text classification (Wang et al., 2013).

[4]The corpus is freely available for download at the URL specified in Section 1.

Algorithm 1 A Dependency Arc Inference Algorithm for Parsing Weibo

Given:
(1) a sentence with tokens T_i, where i is the index, and L is the length;
(2) a database D of token relations from the corpus;
(3) first-order logic inference rule set R.

for $i = 1 \rightarrow L$ tokens **do**
  $\mathbb{S} \leftarrow ConstructSearchSpace(T_i, R, D)$;
  $\vec{P}_i \leftarrow InferParentUsingProPPR(T_i, \mathbb{S})$;
end for

Greedy Global Inference
for $i = 1 \rightarrow L$ tokens **do**
  $Y_i = \arg\max \vec{P}_i$;
end for

4.1 Problem Formulation

We formulate the dependency parsing problem as many small dependency arc prediction problems. For each token, we form the parent inference problem of a token T_i as solving a query $edge(T_i, ?)$ using stochastic theorem proving on a search graph. Our approach relies on a database D of inter-token relations. To construct the database, we automatically extract the token relations from the text data. For example, to denote the adjacency of two tokens T_1 and T_2, we store the entry $adjacent(T_1, T_2)$ in D. One can also store the part-of-speech tag of a token in the form $haspos(T_1, DT)$. There is no limitations on the arity and the types of the predicates in the database.

Given the database of token relations, one then needs to construct the first-order logic inference theory R for predicting dependency arcs. For example, to construct simple bilexical and bi-POS inference rules to model the dependency of an adjacent head and a modifier, one can write first-order clauses such as:

```
edge(V1,V2) :-
  adjacent(V1,V2),hasword(V1,W1),
  hasword(V2,W2),keyword(W1,W2) #adjWord.

edge(V1,V2) :-
  adjacent(V1,V2),haspos(V1,W1),
  haspos(V2,W2),keypos(W1,W2) #adjPos.

keyword(W1,W2) :- # kw(W1,W2).
keypos(W1,W2) :- # kp(W1,W2).
```

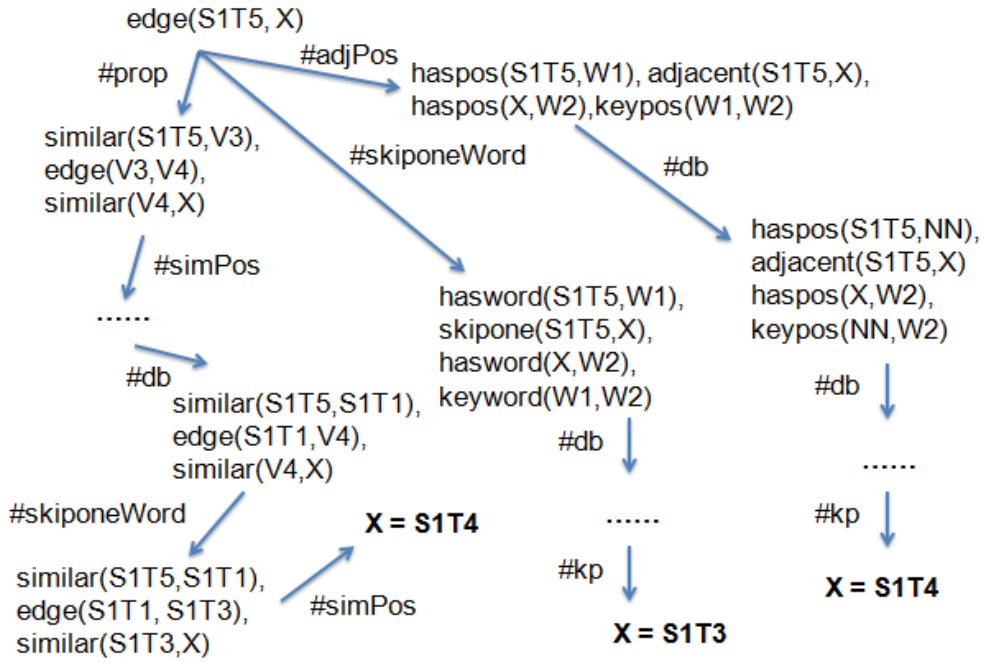


Figure 2: After mapping the database D to theory R, here is an example of search space for dependency arc inference. The query is $edge(S_1T_5, X)$, and there exists one correct and multiple incorrect solutions (highlighted in **bold**).

Here, we associate a feature vector ϕ_c with each clause, which is annotated using the # symbol after each clause in the theory set. Note that the last two (*keyword* and *keypos*) clauses are feature templates that allow us to learn the specific bi-POS tags and bilexical words from the data. In order for one to solve the query $edge(T_i, ?)$, we first need to map the entities from D to R to construct the search space. The details for constructing and searching in the graph can be found in previous studies on probabilistic first-order logic (Wang et al., 2013) and stochastic logic programs (Cussens, 2001). An example search space is illustrated in Figure 2. Note that now the edges in the search graph correspond to the feature vector ϕ_c in R.

The overall dependency arc inference algorithm can be found in Algorithm 1. For each of the parent inference subtask, we use ProPPR (Wang et al., 2013) to perform efficient personalized PageRank inference. Note that to ensure the validity of the dependency tree, we break the loops in the final parse graph into a parse tree using the maximum personalized PageRank score criteria. When multiple roots are predicted, we also select the most likely root by comparing the personalized PageRank solution scores.

To learn the more plausible theories, one needs to upweight weights for relevant features, so that they have higher transition probabilities on the corresponding edges. To do this, we use stochastic gradient descent to learn from training queries, where the correct and incorrect solutions are known. The details of the learning algorithm are described in the last part of this section.

4.2 Personalized PageRank Inference

For the inference of the parent of each token, we utilize ProPPR (Wang et al., 2013). ProPPR allows a fast approximate proof procedure, in which only a small subset of the full proof graph is generated. In particular, if α upper-bounds the reset probability, and d upperbounds the degree of nodes in the graph, then one can efficiently find a subgraph with $O(\frac{1}{\alpha\epsilon})$ nodes which approximates the weight for every node within an error of $d\epsilon$ (Wang et al., 2013), using a variant of the PageRank-Nibble algorithm of Andersen *et al* (2008).

4.3 Parameter Estimation

Our parameter learning algorithm is implemented using a parallel stochastic gradient descent variant to optimize the log loss using the supervised personalized PageRank algorithm (Backstrom and

Method	Dev.	Test
Stanford Parser (Xinhua)	0.507	0.489
Stanford Parser (Chinese)	0.597	0.581
MaltParser (Full)	**0.669**	0.654
Our methods — ProPPR		
ReLU (Bi-POS)	0.506	0.517
ReLU (Bilexical)	0.635	0.616
ReLU (Full)	0.668	0.666
Truncated $tanh$ (Bi-POS)	0.601	0.594
Truncated $tanh$ (Bilexical)	0.650	0.634
Truncated $tanh$ (Full)	0.667	**0.675***

Table 1: Comparing our Weibo parser to other baselines (UAS). The off-the-shelf Stanford parser uses its attached Xinhua and Chinese factored models, which are trained on external Chinese treebank of newswire data. MaltParser was trained on the same in-domain data as our proposed approach. * indicates $p < .001$ comparing to the MaltParser.

Leskovec, 2011). The idea is that, given the training queries, we perform a random walk with restart process, and upweight the edges that are more likely to end up with a known correct parent. We learn the transition probability from two nodes (u, v) in the search graph using: $\Pr_{\mathbf{w}}(v|u) = \frac{1}{Z} f(\mathbf{w}, \Phi^c_{\text{restart}})$, where we use two popular non-linear parameter learning functions from the deep learning community:

- Rectified Linear Unit (ReLU) (Nair and Hinton, 2010): $max(0, x)$;
- The Hyperbolic Function (Glorot and Bengio, 2010): $tanh(x)$.

as the f in this study. ReLU is a desirable non-linear function, because it does not have the vanishing gradient problem, and produces sparse weights. For the weights learned from $tanh(x)$, we truncate the negative weights on the edges, since the default weight on the feature edges is $\mathbf{w} = 1.0$ (existence), and $\mathbf{w} = 0.0$ means that the edge does not exist in the inference stage.

5 Experiments

In this experiment, we compare the proposed parser with two well-known baselines. First, we compare with an off-the-shelf Stanford Chinese Parser (Levy and Manning, 2003). Second, we compare with the MaltParser (Nivre et al., 2007) that is trained on the same in-domain Weibo dataset. The train, development, and test splits are described in Section 3. We tune the regularization hyperparameters of the models on the dev. set, and report Unlabeled Attachment Score (UAS) results for both the dev. set and the hold-out test set. We experiment with the bilexical and bi-POS first-order logic theory separately, as well as a combined full model with directional and distance features.

The results are shown in Table 1. We see that both of the two attached pre-trained models from the Stanford parser do not perform very well on this Weibo dataset, probably because of the mismatched training and test data. MaltParser is widely considered as one of the most popular dependency parsers, not only because of its speed, but also the acclaimed accuracy. We see that when using the full model, the UAS results between our methods and MaltParser are very similar on the development set, but both of our approaches outperform the Maltparser in the holdout test set. The truncated $tanh$ variant of ProPPR obtains the best UAS score of 0.675.

6 Conclusion

In this paper, we present a novel Chinese dependency treebank, annotated using Weibo data. We introduce a probabilistic programming dependency arc prediction approach, where theory engineering is made easy. In experiments, we show that our methods outperform an off-the-shelf Stanford Chinese Parser, as well a strong MaltParser that is trained on the same in-domain data. The Chinese Weibo Treebank is made freely available to the research community. In the future, we plan to apply the proposed approaches to dependency and semantic parsing of other languages.

Acknowledgements

We are grateful to anonymous reviewers for useful comments. This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the DEFT program, and a Google Research Award. The authors are solely responsible for the contents of the paper, and the opinions expressed in this publication do not reflect those of the funding agencies.

References

Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. 2008. Local partitioning for directed graphs using pagerank. *Internet Mathematics*, 5(1):3–22.

Lars Backstrom and Jure Leskovec. 2011. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 635–644. ACM.

Daniel M Bikel and David Chiang. 2000. Two statistical parsing models applied to the chinese treebank. In *Proceedings of the second workshop on Chinese language processing: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 12*, pages 1–6. Association for Computational Linguistics.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164. Association for Computational Linguistics.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *EMNLP-CoNLL*, pages 957–961.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D Manning. 2009. Discriminative reordering with chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*, pages 51–59. Association for Computational Linguistics.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 13–16. Association for Computational Linguistics.

Wanxiang Che, Jiang Guo, and Ting Liu. 2014. Reliable dependency arc recognition. *Expert Systems with Applications*, 41(4):1716–1722.

David Chiang and Daniel M. Bikel. 2002. Recovering latent information in treebanks. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

James Cussens. 2001. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic parsing action models for multi-lingual dependency parsing. In *EMNLP-CoNLL*, pages 940–946.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics.

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

Noah Goodman, Vikash Mansinghka, Daniel Roy, Keith Bonawitz, and Daniel Tarlow. 2012. Church: a language for generative models. *arXiv preprint arXiv:1206.3255*.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint pos tagging and dependency parsing in chinese. In *IJCNLP*, pages 1216–1224.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, Doha, Qatar, October. ACL.

Roger Levy and Christopher Manning. 2003. Is it harder to parse chinese, or the chinese treebank? In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 439–446. Association for Computational Linguistics.

Zhongguo Li and Guodong Zhou. 2012. Unified dependency parsing of chinese morphological and syntactic structures. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1445–1454. Association for Computational Linguistics.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese pos tagging and dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1180–1191. Association for Computational Linguistics.

Wang Ling, Guang Xiang, Chris Dyer, Alan Black, and Isabel Trancoso. 2013. Microblogs as parallel corpora. In *Proceedings of the 51st Annual Meeting on Association for Computational Linguistics*, ACL '13. Association for Computational Linguistics.

Ting Liu, Jinshan Ma, and Sheng Li. 2006. Building a dependency treebank for improving chinese parser. *Journal of Chinese Language and Computing*, 16(4):207–224.

James Robert Lloyd, David Duvenaud, Roger Grosse, Joshua B Tenenbaum, and Zoubin Ghahramani. 2014. Automatic construction and natural-language description of nonparametric regression models. *arXiv preprint arXiv:1402.4304*.

Ji Ma, Tong Xiao, Jingbo Zhu, and Feiliang Ren. 2012. Easy-first chinese pos tagging and dependency parsing. In *COLING*, pages 1731–1746.

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

André Filipe Torres Martins. 2012. *The Geometry of Constrained Structured Prediction: Applications to Inference and Learning of Natural Language Syntax*. Ph.D. thesis, Columbia University.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98. Association for Computational Linguistics.

Michael T. Mordowanec, Nathan Schneider, Chris Dyer, and Noah A. Smith. 2014. Simplified dependency annotations with gfl-web. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. ACL.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The conll 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 915–932. sn.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with lr models and parser ensembles. In *EMNLP-CoNLL*, volume 2007, pages 1044–1050.

Nathan Schneider, Brendan O'Connor, Naomi Saphra, David Bamman, Manaal Faruqui, Noah A Smith, Chris Dyer, and Jason Baldridge. 2013. A framework for (under) specifying dependency syntax without overloading annotators. *arXiv preprint arXiv:1306.2091*.

Kristina Toutanova and Christopher D Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pages 63–70. Association for Computational Linguistics.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 171.

William Yang Wang, Kathryn Mazaitis, and William W Cohen. 2013. Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2129–2138. ACM.

William Yang Wang, Kathryn Mazaitis, Ni Lao, Tom Mitchell, and William W Cohen. 2014. Efficient inference and learning in a large knowledge base: Reasoning with extracted information using a locally groundable first-order probabilistic logic. *arXiv preprint arXiv:1404.3301*.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, pages 398–403.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13. ACM.

Kun Yu, Daisuke Kawahara, and Sadao Kurohashi. 2008. Chinese dependency parsing with large scale automatically constructed case structures. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 1049–1056. Association for Computational Linguistics.

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

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Character-level chinese dependency parsing. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, Baltimore, MD, USA, June. ACL.

Exploiting Community Emotion for Microblog Event Detection

Gaoyan Ou[1,2], **Wei Chen**[1,2], **Tengjiao Wang**[1,2], **Zhongyu Wei**[1,3],
Binyang Li[4], **Dongqing Yang**[1,2] **and Kam-Fai Wong**[1,3]
[1]Key Laboratory of High Confidence Software Technologies, Ministry of Education, China
[2]School of Electronics Engineering and Computer Science, Peking University, China
[3]Shenzhen Research Institute, The Chinese University of Hong Kong, China
[4]Dept. of Information Science & Technology, University of International Relations, China
pekingchenwei@pku.edu.cn

Abstract

Microblog has become a major platform for information about real-world events. Automatically discovering real-world events from microblog has attracted the attention of many researchers. However, most of existing work ignore the importance of emotion information for event detection. We argue that people's emotional reactions immediately reflect the occurring of real-world events and should be important for event detection. In this study, we focus on the problem of community-related event detection by community emotions. To address the problem, we propose a novel framework which include the following three key components: *microblog emotion classification*, *community emotion aggregation* and *community emotion burst detection*. We evaluate our approach on real microblog data sets. Experimental results demonstrate the effectiveness of the proposed framework.

1 Introduction

Microblog has become a popular and convenient platform for people to share information about social events in real time. When an external event occurs, it will be quickly propagated between microblog users. During propagation process of an event, sufficient amount of users will express their emotions. Taking Sina Weibo[1] as an example, more than 12 percent of users use emoticons[2] when reposting an event-related microblog message.

The emotion information can not only help us better understand a given event, but also be utilized to discover new events. Figure 1 shows

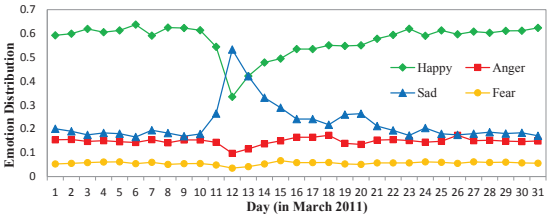


Figure 1: The global emotion dynamics

the emotional distribution dynamics of the overall microblog messages in March 2011. The sudden change of the public emotion distribution on March 12 indicates a public event: *3.11 Earthquake in Japan*. We can see that emotional changes immediately reflect the occurring of real-world events, thus it is reasonable to use them to perform event detection.

Most existing research on microblog event detection (Weng and Lee, 2011; Sakaki et al., 2010; Becker et al., 2010) only account for the factual information (e.g., burstness of topic keywords). They usually ignore the importance of emotion information for event detection. Although there have recently been a few papers (Zhao et al., 2012a; Nguyen et al., 2013; Akcora et al., 2010) in this direction, they have a number of disadvantages. Firstly, they can not detect community-related events. Since they all aggregate emotion at global level, they can only discover national attention events, such as public holidays ("Christmas" and "Spring Festival") or natural disasters. In many applications, discovering events related to a certain group of users or a certain topic is more meaningful. Consider the following questions: "*what happened in the football community last week?*" and "*what are the most significant events in the lawyer community last month?*" Secondly, they assign equal weight to each microblog message or user when aggregating them to a global emotion score. Such approaches may lead to incorrect results when a user posts emotion spam

[1]http://weibo.com/
[2]An icon to indicate user's emotion, as shown in Table 1.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1159–1168,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

in the community. Thirdly, there is a lack of quantitative evaluation using real-world events in existing work. For example, there is only case study in (Zhao et al., 2012a) and (Akcora et al., 2010).

In this study, we focus on a new task of detecting community-related events via community emotion. Our intuition is inspired from the social psychology theory of *group emotion*. In social psychology, *group emotion* refers to the moods, emotions and dispositional affects of a group of people [3]. It can arise as a result of group-relevant events (Smith et al., 2007). For example, people will feel happy when their favorite football team wins a game and feel sad when their team loses a game. Thus we can use the community emotion as signals to detect community-related events. To achieve good performance of this new task, the following two factors must be considered: 1) *how to measure the community emotion based on microblog message or user emotion* and 2) *how to perform event detection based on the sequence of community emotion*.

To measure community emotion, we argue that in a given community, different users have different emotional authorities. The emotion of highly influential people in the community may be more important in determining the community emotion (Barsäde and Gibson, 1998). We propose to use the user's emotion authority computed by the social network of the community to weight the user when aggregating community emotion. Since spam user is unlikely to have high emotion authority in the community, our method can reduce the effect of emotion spam. Based on the community emotion, we present an emotion burst detection algorithm for the community emotion distribution sequence. We identify two emotional states of the community: "emotion stable" state and "emotion burst" state. We use the Dirichlet distribution to model the generation process of the community emotion distribution. An efficient emotion burst detection algorithm is presented to detect community-related events.

We evaluate our approach on large-scale microblog data sets by using real-world event list for each community. Experimental results demonstrate that the community emotion is an effective signal for community-related event detection. In comparison with several baseline methods, our emotion burst detection algorithm also improves the event detection performance in terms of precision, recall and F-measure.

[3] http://en.wikipedia.org/wiki/Group_emotion

2 Related Work

In this section, we review the related work on sentiment analysis and event detection in microblog, respectively.

Sentiment Analysis in Microblog: Sentiment analysis (Pang and Lee, 2008; Liu, 2012) is mainly about analyzing people's opinions, sentiments and emotions towards a given event, topic, product, etc. Microblog platforms like Twitter and Weibo, provide people a convenient way to post their emotional reactions towards social events in almost real time. This leads to increasing number of interests on sentiment analysis in microblog data (Davidov et al., 2010; Liu et al., 2012; Go et al., 2009; Agarwal et al., 2011; Pak and Paroubek, 2010; Jiang et al., 2011; Speriosu et al., 2011; Bermingham and Smeaton, 2010). The training data for microblog sentiment analysis are usually obtained in an automatic manner by utilizing emoticons, hashtags and smileys. Davidov et al. (2010) propose an approach to automatically obtain labeled training examples by exploiting hashtags and smileys. Liu et al. (2012) proposed an emoticon smoothed method to integrate both manually labeled data and noisy labeled data for Twitter sentiment classification.

Different from existing microblog sentiment analysis work, which aims at discovering sentiments and emotions for an event or topic given in advance, we are interested in utilizing the emotion information in microblog messages for real-world event detection. Our work use sentiment analysis as a tool to perform microblog emotion classification. Then we propose an event detection algorithm based on the sequence of community level emotion distribution.

Event Detection in Microblog: Event detection from microblog data has attracted increasing attention recently. We divide existing work into the following two categories: *non-emotion-based* methods and *emotion-based* methods. *Non-emotion-based* methods try to exploit keyword or activity patterns to discover events (Weng and Lee, 2011; Sakaki et al., 2010; Becker et al., 2010; Mathioudakis and Koudas, 2010). Mathioudakis and Koudas (2010) first identify "bursty" keywords and then discover events by grouping bursty keywords together. Zhao et al. (2012b) focuses on

identifying event-related burst from social media based on multiple activities. They propose a unified optimization model to integrate multiple correlated activity streams based on the state machine in (Kleinberg, 2003).

Emotion-based methods try to exploit the emotional reactions to discover events (Zhao et al., 2012a; Akcora et al., 2010). Akcora et al. (2010) model public emotion as an emotion vector computed by counting emotion words and then use a rule-based method to identify public breakpoints. Zhao et al. (2012a) use a simple top-k method to detect abnormal events based on sequence of sentiment variance.

Our method is also an emotion-based method. However, our approach is different from existing emotion-based methods in the following aspects. Firstly, while existing work aggregates emotion for all users, we focus on emotion for a certain community to discover community-related events. Secondly, existing methods assume that the emotions of different users are of equal importance. We distinguish user's emotional authority based on the community structure of users.

3 Preliminary Definitions

In this section, we first give some basic concepts before formally defining our problem.

Topical Community: A group of connected users that are interested in a specific topic. A topical community can be denoted as $C = \{V, E, M\}$, where V is the set of community users, E is the set of relationships of between users. $M = \{M_1, M_2, ..., M_T\}$ is the microblog message collection in the community, which is segmented according to time. M_t is the microblog message collection in time interval t.

Emotion Distribution: An emotion distribution e_t is a real-value N-dimension vector sampled from the emotion space, satisfying the constraint $\sum_{i=1}^{N} e_{ti} = 1$. It indicates the emotional state of a microblog message, a user or a community at time t. At a given time interval t, a user emotion distribution $e_t(u)$ is computed by an aggregation function over the emotion distribution of the microblog messages posted by u in time interval t. Community emotion distribution $e_t(c)$ is computed by an aggregation function over the emotion distribution of the community users in c at time interval t.

Community Emotion Burst: Given an emotion distribution sequence e_t for community c, an emotion burst is defined as period $[t_s, t_e]$ in which some event relevant to c takes places. During the time period, the emotion distribution of c is :1) significantly different from its average emotion distribution, or 2) extremely uneven distributed.

Given the above definitions, our object is then to detect community-related events from the emotion distribution sequence of the community.

4 The Proposed Framework

In this section, we describe our microblog event detection framework. The framework aims to detect community-related events based on the community emotion. The overview of our framework is shown in Figure 2. In particular, we define the following four main components:

1) **Microblog emotion classification:** We train emotion classification model by automatically acquiring training data using the emoticons.

2) **Community emotion aggregation:** We assume that in a given community, different users have different weights when aggregating community emotion. Thus we propose a novel weighting approach based on the user's authority.

3) **Community emotion burst detection:** Given the community emotion, we propose an emotion burst detection algorithm to detect community emotion bursts.

4) **Event extraction:** The function of this component is to extract event keywords for each community emotion burst. We count the document frequency (DF) of each term contained in the microblog messages in the burst period. Then the top-5 DF terms are used to describe the event occurring during the burst period, although there exist alternative techniques (Ritter et al., 2012; Li et al., 2010).

Since the last component is not the main focus of this work, we only introduce the first three components in detail in the following subsections.

4.1 Microblog Emotion Classification

We build an emotion classifier using the multinomial Naïve Bayes classifier for each community to classify microblog messages into different emotion classes. Here we are interested in the setting where the microblog messages arrive as stream, thus it is not appropriate to use a constant set of manually labeled messages. To avoid manual annotation cost, we adopt a distant supervision approach (Go et al., 2009; Davidov et al., 2010; Hu

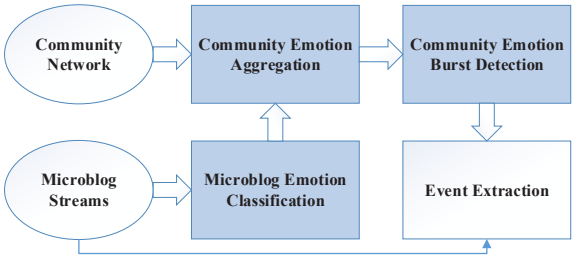


Figure 2: Overview of our community-related event detection framework

Table 1: List of emoticons

Happy(26)	good	laugh	love	smile	cute
Anger *(19)*	anger	hate	curse	despise	crazy
Sad(13)	sad	disappoint	cry	unhappy	unwell
Fear(11)	fear	surprise	shame	doubt	fright

et al., 2013) to acquire labeled microblog messages automatically by using the emoticons. We first select a set of emoticons which are most frequently used to express emotion in microblog. Then we manually map each emoticon into four emotion classes (26 for *happy*, 19 for *anger*, 13 for *sad* and 11 for *fear*). We only show the top five emoticons for each emotion type in Table 1. The labeled emoticons can then be used to acquire training data to train an emotion classifier for a community in any time period.

We combine the features which have been proven effective by previous work, such as punctuation and emotion lexicons. Specifically, we use the following features : 1) Words appearing in the microblog message serve as word features, 2) Number of "!" characters and "?" in the microblog message, and 3) Each term in a general emotion lexicon serves as an emotion lexicon feature.

The emotion distribution $e(m)$ of a microblog message m is represented by a N-dimension distribution vector. For example, if a microblog message m is classified as *happy*, then its emotion distribution $e(m)$ is $[1, 0, 0, 0]$. The emotion distribution of $e(u)$ of user u at time t is average emotion distribution of the microblog messages posted by him during time t.

$$e_t(u) = \frac{1}{N_{ut}} \sum_{m=1}^{N_{ut}} e(m) \quad (1)$$

where N_{ut} is the number of microblog messages posted by user u at time t.

4.2 Community-level Emotion Aggregation

A common strategy to measure community emotion is to compute the average emotion of the community users. It is based on the assumption that the emotion of different community users are with equal importance. This is implemented by employing an *average* aggregation function on the individual emotion distribution $e_t(u)$:

$$e_t(c) = \frac{1}{N_c} \sum_{u \in c} e_t(u) \quad (2)$$

where N_c is the number of users in community c.

Intuitively, the emotion of user with higher authority in community should be more important in determining community emotion. Thus we estimate community emotion by taking into account user's emotional authority, which is based on the assumption that different users in a community have different emotional authorities.

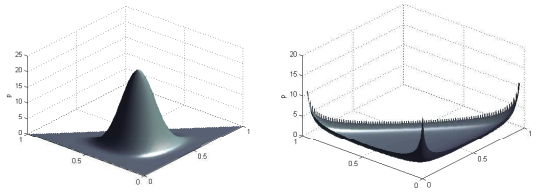
We employ HITS algorithm(Kleinberg, 1999) to compute the user authority $auth(u)$ based on the user network $\{V, E\}$ of the community. Then $auth(u)$ is used to represent the emotional authority of user u. This authority-based community emotion aggregation approach can also reduce the effect of spam users, since they usually have low authorities in the community network. For simplicity and computation efficiency, we assume that $auth(u)$ is time independent, which means that we only need to run the HITS algorithm once for each community. Given the user emotion distributions and the user emotional authorities, the emotion distribution for a community c in time interval t can be measured as:

$$e_t(c) = \frac{1}{A_{ct}} \sum_{u \in c} auth(u) e_t(u) \quad (3)$$

where $A_{ct} = \sum_{u \in c_t} auth(u)$ is the normalization term.

4.3 Community Emotion Burst Detection

We formulate our problem into the binary state model framework (Kleinberg, 2003). For a given community c, there are T time intervals in total, with community emotion distribution sequence $e = (e_1, e_2, ..., e_T)$ and state sequence $q = (q_1, q_2, ..., q_T)$. Each q_t can be one of the following two states: $q_t = 0$ ("emotion stable" state) and $q_t = 1$ ("emotion burst" state). Our objective



(a) "emotion stable" state (b) "emotion burst" state

Figure 3: Emotion probability density for "emotion stable" state and "emotion burst" state

is to find an optimal state sequence q^* given the emotion distribution sequence e.

Since each e_t is a distribution rather than positive integer, the emotion generation process can no longer be modeled by a Poisson distribution. We choose to model the emotion generation process by the Dirichlet distribution. This process is analogous to the document-level topic generation in the LDA topic model (Blei et al., 2003).

If community c is in an "emotion stable" state in time interval t, its emotion distribution e_t should be close to the average emotion distribution e_a. The density function is defined as $f(e_t, e_a, s_t = 0) = Dirichlet(\alpha_0 e_a)$, where $s_t = 0$ indicates that the community is in an "emotion stable" state and α_0 is a positive parameter. To ensure that e_t closer to e_a will get higher probability, α_0 should satisfy the constraint $\alpha_0 \cdot \min e_a > 1$. An example of the probability density function of $f(e_t, e_a, s_t = 0)$ in a three dimension emotion space is shown in Figure 3(a).

If community c is in an "emotion burst" state in time interval t, the emotion distribution of c is : 1) significantly different from its average emotion distribution, or 2) extremely uneven distributed. For example, assume $e_a = [0.25, 0.25, 0.25, 0.25]$, if community is in burst in time interval t, its emotion distribution is more likely to be $[0.1, 0.4, 0.1, 0.4]$ (significantly different from e_a) or $[0.1, 0.7, 0.1, 0.1]$ (extremely *anger*). The density function is then defined as $f(e_t, e_a, s_t = 1) = Dirichlet(\alpha_1 e_a)$. $s_t = 1$ indicates that the community is in an "emotion burst" state and α_1 should satisfy the constraint: $\alpha_1 \cdot \max e_a < 1$. An example of the probability density function of $f(e_t, e_a, s_t = 1)$ in a three dimension emotion space is shown in Figure 3(b).

Based on above discussion, the cost function for an emotion state sequence q can be defined as:

$$cost(q|e) = \sum_{t=1}^{T} -\ln(f_t(e_t, e_a, q_t)) + b\ln(\frac{1-\pi}{\pi}) \quad (4)$$

where π is the probability the community will change the emotion state in two consecutive time intervals t and $t+1$. b denotes the number of emotion state changes in the whole time intervals $[1, T]$.

Equation 4 is exactly the objective function we need to optimize. This optimization problem can be efficiently solved by using a dynamic programming procedure, as summarized in Algorithm 1. Algorithm 1 mainly consists of two phases: a forward phase (line 5 - line 10) which calculates the minimal cost for emotion distribution subsequence and a backward phase to construct the optimal emotion state sequence (line 11 - line 14).

For convenience, we use $f_t(s)$ to denote $f_t(e_t, e_a, s)$. $\pi_{s's} = \pi$ if $s' \neq s$, otherwise $\pi_{s's} = (1-\pi)$. $c_t(s)$ denotes the minimal cost of generating the emotion distribution sub-sequence $\{e_1, ..., e_t\}$ with $q_t = s$. $q'_t(s)$ stores the state of time interval $t-1$ for the most likely state subsequence so far with $q_t = s$.

Algorithm 1 Emotion Burst Extraction

Input: The emotion distribution sequence $e = (e_1, e_2, ..., e_T)$, the state transition cost π and the parameters α_0 and α_1

1: **for** each $s \in \{0, 1\}$ **do**
2: Initialize $c_1(s) = -\ln f_1(s)$
3: Initialize $q'_1(s) = 0$
4: **end for**
5: **for** each $t = 2, \cdots, T$ **do**
6: **for** each $s \in \{0, 1\}$ **do**
7: $c_t(s) = \min_{s'}(c_{t-1}(s') - \ln f_t(s) - \ln \pi_{s's})$
8: $q'_t(s) = \arg\min_s(c_{t-1}(s') - \ln \pi_{s's})$
9: **end for**
10: **end for**
11: $q^*(T) = \arg\min_s c_T(s)$
12: **for** each $t = T-1, \cdots, 2$ **do**
13: $q^*(t) = q'_{t+1}(q^*(t+1))$
14: **end for**

Output: The optimal emotion state sequence q^*

Table 2: Basic statistics of data sets

Community	#User	#Link	#Microblog	#Event
legal cases	4937	97639	269871	31
football	9928	105483	416631	75
economy	2657	65403	179584	46
singer	3759	79458	478265	53

5 Experimental Setup

5.1 Data Set

We use a large microblog data set crawled from Sina Weibo, which is the most popular microblog service in China. This data set contains 212,859,466 Chinese microblog messages posted by 916,126 users in a period of about 8 months (1/12/2010∼ 20/7/2011). We also crawled the *following* network, which resulted in 15,681,296 *following* relationships for the 916,126 users.

We choose four communities: "legal cases", "football", "economy" and "singer". To obtain the members for each topical community, we manually selected several keywords and input them as queries to the user search page[4]. After filtering out the users whose microblog messages are not collected in our corpora, we extract the sub-network of the users from the whole *following* network.

We use a simple but efficient method to extract microblog messages for each topical community: 1) If a microblog message is posted by the community members and also contains keywords of the community, it belongs to the community; 2) If a microblog message is posted by community member u and it is *reposting*, *commenting on* or *replying to* another microblog posted by community member v, then it belongs to the community. The basic statistics of our data sets are shown in Table 2.

5.2 Ground Truth Generation

Algorithm 1 generates a list of emotion bursts for each community. Since our goal is to identify community-related events, we need to evaluate how well the generated emotion bursts correspond to the ground truth real-world events. To generate the ground truth for evaluation, we utilize the news website Sina News[5]. Two PhD students are invited to manually generate a list of real events for each community by referring to the annual topic summary page[6] of Sina News. The annotation agreement is higher than 90%. Each event is also associated with its occurred date. The number of events for each community is shown in the last column of Table 2. For each community, the event list is then used to evaluate the performance of different event detection models.

5.3 Evaluation Metric

We use precision, recall and F-measure as evaluation metric. For each community, we compare the event list $\mathcal{E}^c$ and the generated burst list $\mathcal{B}^c$ to compute the above metric. Specifically, the precision, recall and F-measure for a community c are defined as follows:

$$P = \frac{\sum_{j=1}^{|\mathcal{E}^c|}\sum_{k=1}^{|\mathcal{B}^c|} I(\mathcal{E}_j^c \in \mathcal{B}_k^c)}{|\mathcal{B}^c|} \quad (5)$$

$$R = \frac{\sum_{j=1}^{|\mathcal{E}^c|}\sum_{k=1}^{|\mathcal{B}^c|} I(\mathcal{E}_j^c \in \mathcal{B}_k^c)}{|\mathcal{E}^c|} \quad (6)$$

$$F = \frac{2 \times P \times R}{P + R} \quad (7)$$

where $\mathcal{E}_j^c$ is the occurring time of the j-th event in community c, $\mathcal{B}_k^c$ is the k-th identified burst for community c. $I(.)$ is the indicator function (which equals 1 if its argument is true and 0 otherwise).

The final precision, recall and F-measure are averaged over different communities.

5.4 Compared Methods

We now introduce four methods used for comparison as follows:

EmoPeakFind: The method proposed in (Akcora et al., 2010), which aims at discovering breakpoints from public emotion. They use the following simple rule to find breakpoints from emotion sequences:

$$Sim(e_{t-1}, e_t) < Sim(e_{t-2}, e_{t-1}) \quad (8)$$

$$Sim(e_{t-1}, e_t) < Sim(e_t, e_{t+1}) \quad (9)$$

where Sim is a similarity function. We use the cosine similarity function in our evaluation.

TopKEmoVar: The method used in (Zhao et al., 2012a). They first derive a sequence of relative variation V^n for each single emotion sequence e^n. Then they define a sequence of emotion variation as $(\sum_{n=1}^{4} |V_t^n|)$. This sequence is sorted in descending order and the top-k t is selected as burst

[4]http://s.weibo.com/user/&tag=keyword

[5]http://news.sina.com.cn

[6]http://news.sina.com.cn/zt/

Table 3: Event detection performance of different methods

Method	Precision	Recall	F-measure
EmoPeakFind	0.313	0.625	0.417
TopKEmoVari	0.423	0.423	0.423
KLB	0.575	0.702	0.632
MBurst	0.534	0.497	0.515
Our Method	**0.654**	**0.715**	**0.683**

state points, where k is set to be the size of the event list for each community.

KLB: The method proposed in (Kleinberg, 2003). Note that *KLB* can only deal with a single sequence $e^n = (e_1^n, ..., e_t^n, ...e_T^n)$ for emotion type n. We first apply *KLB* to find the optimal state sequences for each emotion type. Then we perform an OR function to merge the N state sequences to a global emotion state sequence.

MBurst: The method proposed in (Zhao et al., 2012b) for multi-stream burst detection. *MBurst* is evaluated on three activity streams in (Zhao et al., 2012b). Here we apply *MBurst* to the N emotion streams. Then we perform an OR function to merge the N state sequences to a global emotion state sequence.

6 Experimental Results

6.1 Performance Comparison Results

In this experiment, we compare our method with different baseline methods as introduced in Section 5.4. We use Equation (3) to aggregate community emotion for all the compared methods. The parameter α_0 and α_1 are empirically set to be $\frac{5}{\min_n e_a^n}$ and $\frac{0.5}{\max_n e_a^n}$, respectively. The experimental results are shown in Table 3.

Table 3 shows that *EmoPeakFind* and *TopKEmoVari* are less effective than other methods. The simple rule used in *EmoPeakFind* produced many noisy bursts, leading to low precision. *TopKEmoVari* only considers the relative variation of two consecutive time intervals. The choose of k is also nontrivial since it is hard to know the number of events before the events are identified. Note that *EmoPeakFind* and *TopKEmoVari* are both rule-based methods, while *KLB*, *MBurst* and *Our Method* are state machine based methods. This demonstrates that for community-related emotion burst detection, it is more appropriate to use a state machine based model.

It looks surprising that *MBurst* performs worse than *KLB*, since *MBurst* is specifically designed for multiple streams. However, *MBurst* is based on the assumption that the states of multiple streams in the same time interval tend to be the same (i.e., there is positive correlation between two different streams). This assumption no longer holds in the context of different emotion streams. For example, if a negative event occurs in the community, while sad emotion is likely to be in a burst state, $happy$ emotion is not likely to be in a burst state.

We can see from Table 3 that our method outperforms the four baselines. The main reason is that our burst detection method is based on the sequence of community emotion distribution. Modeling community emotion as a distribution is more suitable than modeling several different emotion types.

We further show some example events detected by our model in Table 4. Since the event keywords are manually translated from Chinese, one keyword may include more than one English words. We can see that community emotion can help to detect emotionally significant events for different communities. For example, the "legal cases" community is in a strong *anger* emotional state on December 25, 2010, which indicates an important event "Qian Yunhui's case".

Table 5: Performance of different weighting schemes in terms of F-measure

Community	Weighting schema	
	equal	*HITS-based*
legal cases	0.517	0.590
football	0.674	0.765
economy	0.642	0.712
singer	0.589	0.665
avg	0.605	0.683

6.2 Effect of Emotion Aggregation Functions

In this experiment, we show the importance of correctly aggregation community level emotion for community-related event detection. We compare the two aggregation approaches introduced in section 4.2. The first approach assigns equal weight to each community users, while the second approach assigns weights to users based on their authorities in the community.

The performance in terms of F-measure is shown in Table 5. It is obvious that, for all commu-

Table 4: Examples of events and the corresponding community emotions for two communities. The four emotion types (*happy*, *anger*, *sad* and *fear*) are mapped to green, red, blue and yellow color, respectively.

Community	Date	Emotion	Event keywords
legal cases	25/12/2010		Qian Yunhui, village head, Yueqing, run over, Zhejiang
	22/4/2011		Yao Jiaxin, death penalty, first instance, Xi'an, intentional killing
	9/5/2011		Xia Junfeng, final judgment, death penalty, pedlar, Shenyang
football	2/12/2010		Qatar, 2022, world cup bid, win, FIFA
	14/2/2011		Ronaldo, retire, legend, football, Brazil
	29/5/2011		Barcelona, 2011, Champions League final, win, UEFA

nities, HITS-based weighting approach performs better than equal weighting approach. Thus we can conclude that user authority is important when aggregating community emotion.

We further perform an empirically analysis of the events that successfully identified by HITS-based approach but failed by equal weighting approach. By manually analyzing the microblog messages of the corresponding time intervals, we found that most of the errors of equal weighting approach were caused by emotion spam. Users of low authority post many microblog messages with extreme emotion to claim the attention of the community. Since there is no significant community-related events at that time interval, we do not observe emotional changes of the high authority uses. In the equal weighting method, the existence of emotion spam lead to wrong result of community emotion. Since the weights of users who post emotion spam are small in the HITS-based approach, they have little effect on the community emotion. This is the main reason why HITS-based weighting method is more effective than equal weighting method.

6.3 Parameter Sensitivity

In this experiment, we test the sensitivity of our model by different choices of the parameters α_0 and α_1. $\alpha_0 \min_n e_a^n$ is set to be $[2, 3, 4, 5, 6, 7, 8]$ and $\alpha_1 \max_n e_a^n$ is set to be $[0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8]$. The event detection results of different parameter settings are shown in Figure 4.

It can be seen from Figure 4 that: 1) The performance is relatively good in a particular range of the parameters. When $\frac{2}{\min_n e_a^n} \leq \alpha_0 \leq \frac{7}{\min_n e_a^n}$ and $\frac{0.3}{\max_n e_a^n} \leq \alpha_1 \leq \frac{0.7}{\max_n e_a^n}$, the F-measure is consistently larger than 0.55. 2) In general, the performance is more sensitive to α_1 than to α_0. Note that α_1 controls the generation process of the emotion distribution when the community is in the "emotion burst" state, thus it is relatively more important to tune α_1 than α_0. The experimental results demonstrates that when $\frac{0.3}{\max_n e_a^n} \leq \alpha_1 \leq \frac{0.7}{\max_n e_a^n}$, the performance can be relatively good.

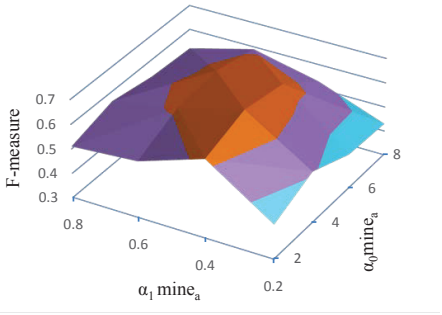


Figure 4: F-measure of our method for event detection in different parameter settings

7 Conclusion

Microblog has become a popular and convenient platform for people to share information about social events in real time. In this paper, we focus on the problem of community-related event detection by community emotions. We propose a novel method to compute community-level emotion by considering the user authority in the community network. Then we present an effective emotion burst detection algorithm for the community emotion distribution sequence.

We evaluate our approach on real microblog data sets. Experimental results demonstrate that it is important to take into account the user authority when aggregating community emotion for

community-related event detection. Our emotion burst detection algorithm also achieves better performance than several baseline methods.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments and suggestions. This research is supported by the National High Technology Research and Development Program of China (Grant No. 2012AA011002), Natural Science Foundation of China (Grant No. 61300003). This research is partially supported by General Research Fund of Hong Kong (417112), Shenzhen Fundamental Research Program (JCYJ20130401172046450) and Shenzhen International Cooperation Funding GJHZ20120613110641217. This work is partially supported by Huawei Noah's Ark Lab, Hong Kong. The contact author of this paper, according to the meaning given to this role by Peking University, is Wei Chen.

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, pages 30–38. Association for Computational Linguistics.

Cuneyt Gurcan Akcora, Murat Ali Bayir, Murat Demirbas, and Hakan Ferhatosmanoglu. 2010. Identifying breakpoints in public opinion. In *Proceedings of the First Workshop on Social Media Analytics*, pages 62–66. ACM.

Sigal G Barsäde and Donald E Gibson. 1998. Group emotion: A view from top and bottom. *Research on Managing Groups and Teams*, 1:81–102.

Hila Becker, Mor Naaman, and Luis Gravano. 2010. Learning similarity metrics for event identification in social media. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 291–300. ACM.

Adam Bermingham and Alan F. Smeaton. 2010. Classifying sentiment in microblogs: is brevity an advantage? In *CIKM*, pages 1833–1836.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *COLING (Posters)*, pages 241–249.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12.

Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. 2013. Unsupervised sentiment analysis with emotional signals. In *WWW*, pages 607–618.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *ACL*, pages 151–160.

Jon M Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632.

Jon Kleinberg. 2003. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397.

Zhenhui Li, Ding Zhou, Yun-Fang Juan, and Jiawei Han. 2010. Keyword extraction for social snippets. In *Proceedings of the 19th international conference on World wide web*, pages 1143–1144. ACM.

Kun-Lin Liu, Wu-Jun Li, and Minyi Guo. 2012. Emoticon smoothed language models for twitter sentiment analysis. In *AAAI*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Michael Mathioudakis and Nick Koudas. 2010. Twittermonitor: trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 1155–1158. ACM.

Thin Nguyen, Dinh Q. Phung, Brett Adams, and Svetha Venkatesh. 2013. Event extraction using behaviors of sentiment signals and burst structure in social media. *Knowl. Inf. Syst.*, 37(2):279–304.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW*, pages 851–860.

Eliot R Smith, Charles R Seger, and Diane M Mackie. 2007. Can emotions be truly group level? evidence regarding four conceptual criteria. *Journal of personality and social psychology*, 93(3):431.

Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. 2011. Twitter polarity classification with label propagation over lexical links and the follower graph. In *EMNLP*, pages 53–63.

Jianshu Weng and Bu-Sung Lee. 2011. Event detection in twitter. In *ICWSM*.

Jichang Zhao, Li Dong, Junjie Wu, and Ke Xu. 2012a. Moodlens: an emoticon-based sentiment analysis system for chinese tweets. In *KDD*, pages 1528–1531.

Wayne Xin Zhao, Baihan Shu, Jing Jiang, Yang Song, Hongfei Yan, and Xiaoming Li. 2012b. Identifying event-related bursts via social media activities. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1466–1477. Association for Computational Linguistics.

Detecting Disagreement in Conversations using Pseudo-Monologic Rhetorical Structure

Kelsey Allen **Giuseppe Carenini** **Raymond T. Ng**
Department of Computer Science, University of British Columbia
Vancouver, BC, V6T 1Z4, Canada
{kelseyra, carenini, rng}@cs.ubc.ca

Abstract

Casual online forums such as Reddit, Slashdot and Digg, are continuing to increase in popularity as a means of communication. Detecting disagreement in this domain is a considerable challenge. Many topics are unique to the conversation on the forum, and the appearance of disagreement may be much more subtle than on political blogs or social media sites such as twitter. In this analysis we present a crowd-sourced annotated corpus for topic level disagreement detection in Slashdot, showing that disagreement detection in this domain is difficult even for humans. We then proceed to show that a new set of features determined from the rhetorical structure of the conversation significantly improves the performance on disagreement detection over a baseline consisting of unigram/bigram features, discourse markers, structural features and meta-post features.

1 Introduction

How does disagreement arise in conversation? Being able to detect agreement and disagreement has a range of applications. For an online educator, dissent over a newly introduced topic may alert the teacher to fundamental misconceptions about the material. For a business, understanding disputes over features of a product may be helpful in future design iterations. By better understanding how debate arises and propagates in a conversation, we may also gain insight into how authors' opinions on a topic can be influenced over time.

The long term goal of our research is to lay the foundations for understanding argumentative structure in conversations, which could be applied to NLP tasks such as summarization, information retrieval, and text visualization. Argumentative structure theory has been thoroughly studied in both psychology and rhetoric, with negation and discourse markers, as well as hedging and dispreferred responses, being known to be indicative of argument (Horn, 1989; Brown and Levinson, 1987). As a starting point, in this paper we focus on the detection of disagreement in casual conversations between users. This requires a generalized approach that can accurately identify disagreement in topics ranging from something as mundane as whether GPS stands for galactic positioning system or global positioning system, to more ideological debates about distrust in science.

Motivated by the widespread consensus in both computational and theoretical linguistics on the utility of discourse markers for signalling pragmatic functions such as disagreement and personal opinions (Webber and Prasad, 2008; Abbott et al., 2011; J. E. Fox-Tree, 2010), we introduce a new set of features based on the Discourse Tree (DT) of a conversational text. Discourse Trees were formalized by Mann and Thompson (1988) as part of their Rhetorical Structure Theory (RST) to represent the structure of discourse. Although this theory is for monologic discourse, we propose to treat conversational dialogue as a collection of linked monologues, and subsequently build a *relation graph* describing both rhetorical connections within user posts, as well as between different users. Features obtained from this graph offer significant improvements on disagreement detection over a baseline consisting of meta-post features, lexical features, discourse markers and conversational features. Not only do these features improve disagreement detection, but the discovered importance of relations known to be theoretically relevant to disagreement detection, such as COMPARISON (Horn, 1989), suggest that this approach may be capturing the essential aspects of the conversational argumentative structure.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1169–1180,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

As a second contribution of this work, we provide a new dataset consisting of 95 topics annotated for disagreement. Unlike the publicly available *ARGUE* corpus based on the online debate forum 4forums.com (Abbott et al., 2011), our corpus is based on Slashdot, which is a general purpose forum not targeted to debates. Therefore, we expect that detecting disagreement may be a more difficult task in our new corpus, as certain topics (like discussing GPS systems) may be targeted towards objective information sharing without any participants expressing opinions or stances. Because of this, our corpus represents an excellent testbed to examine methods for more subtle disagreement detection, as well as the major differences between news-style and argument-style dialogue.

2 Related Work

In the past decade, there have been a number of computational approaches developed for the task of disagreement and controversy detection, particularly in synchronous conversations such as meetings (Somasundaran et al., 2007; Raaijmakers et al., 2008) and in monologic corpora such as news collections (Awadallah et al., 2012) and reviews (Popescu et al., 2005; Mukherjee and Liu, 2012).

In the domain of synchronous conversations, prosodic features such as duration, speech rate and pause have been used for spoken dialogue (Wang et al., 2011; Galley et al., 2004). Galley et al. (2004) found that local features, such as lexical and structural features, as well as global contextual features, were particularly useful for identifying agreement/disagreement in the ICSI meeting corpus. Germesin and Wilson (2009) also showed accuracies of 98% in detecting agreement in the AMI corpus using lexical, subjectivity and dialogue act features. However, they note that their system could not classify disagreement accurately due to the small number of training examples in this category. Somasundaran et al. additionally show that dialogue act features complement lexical features in the AMI meeting corpus (Somasundaran et al., 2007). These observations are taken into account with our feature choices, and we use contextual, discourse and lexical features in our analysis.

In the monologic domain, Conrad et al. (2012) recently found rhetorical relations to be useful for argument labelling and detection in articles on the topic of healthcare. Additionally, discourse markers and sentiment features have been found to assist with disagreement detection in collections of news documents on a particular topic, as well as reviews (Choi et al., 2010; Awadallah et al., 2012; Popescu et al., 2005).

In the asynchronous domain, there has been recent work in disagreement detection, especially as it pertains to stance identification. Content based features, including sentiment, duration, and discourse markers have been used for this task (Yin et al., 2012; Somasundaran and Wiebe, 2009; Somasundaran and Wiebe, 2010). The structure of a conversation has also been used, although these approaches have focused on simple rules for disagreement identification (Murakami and Raymond, 2010), or have assumed that adjacent posts always disagree (Agrawal et al., 2003). More recent work has focused on identifying users' attitudes towards each other (Hassan et al., 2010), influential users and posts (Nguyen et al., 2014), as well as identifying subgroups of users who share viewpoints (Abu-Jbara et al., 2010). In Slashdot, the h-index of a discussion has been used to rank articles according to controversiality, although no quantitative evaluation of this approach has been given, and, unlike in our analysis, they did not consider any other features (Gomez et al., 2008). Content based features such as polarity and cosine similarity have also been used to study influence, controversy and opinion changes on microblogging sites such as Twitter (Lin et al., 2013; Popescu and Pennacchiotti, 2010).

The simplified task of detecting disagreement between just two users (either a question/response pair (Abbott et al., 2011) or two adjacent paragraphs (Misra and Walker, 2013)) has also been recently approached on the *ARGUE* corpus. Abbott et al. (2011) use discourse markers, generalized dependency features, punctuation and structural features, while Misra and Walker (2013) focus on n-grams indicative of denial, hedging and agreement, as well as cue words and punctuation. Most similar to our work is that by Mishne and Glance (2006). They performed a general analysis of weblog comments, using punctuation, quotes, lexicon counts, subjectivity, polarity and referrals to detect disputative and non-disputative comments. Referrals and questions, as well as polarity measures in the first section of the post, were found to be most useful. However, their analysis did not

Type	Num	P/A	S/P	W/P	Num Authors	W/S	TBP	TT	TP	Length
Disagreement - C	19.00	1.02	3.21	65.86	15.84	20.47	4.60	50.90	16.21	49.11
Disagreement - NC	27.00	1.00	3.08	59.80	14.07	19.33	3.89	42.29	14.11	42.26
No disagreement - NC	28.00	1.03	2.85	57.25	10.29	19.94	6.83	50.12	10.50	28.00
No disagreement - C	21.00	1.00	3.69	69.66	6.29	20.22	6.14	18.22	6.29	19.81

Table 1: Characteristics of the four categories determined from the crowd-sourced annotation. All values except for the number of topics in the category are given as the average score per topic across all topics in that category. **Key: C and NC:** Confident ($score \geq 0.75$) and Not confident ($score < 0.75$), **Num:** Number of topics in category, **P/A:** Posts per author, **S/P:** Sentences per post, **W/P:** Words per post, **Num Authors:** Number of authors, **W/S:** Words per sentence, **TBP:** Time between posts (minutes), **TT:** Total time in minutes, **TP:** Total posts, and **Length:** Length of topic in sentences

take into account many features that have been subsequently shown to be relevant, such as discourse markers and conversational structure, and was hampered by a severe imbalance in the test set (with very few disputative comments).

Our method takes advantage of insights from many of these previous studies, focusing on discussion thread structure as well as text based features to form our basic feature set. It is unlike Mishne and Glance's work in that we incorporate several new features, have a balanced testing and training set, and only use comments from one type of online blog. Furthermore, it is a very different task from those so far performed on the *ARGUE* corpus, as we consider topics discussed by more than two users. We aim to compare our features to those found to be previously useful in these related tasks, and expect similar feature sets to be useful for the task of disagreement detection in this new corpus.

3 Corpus

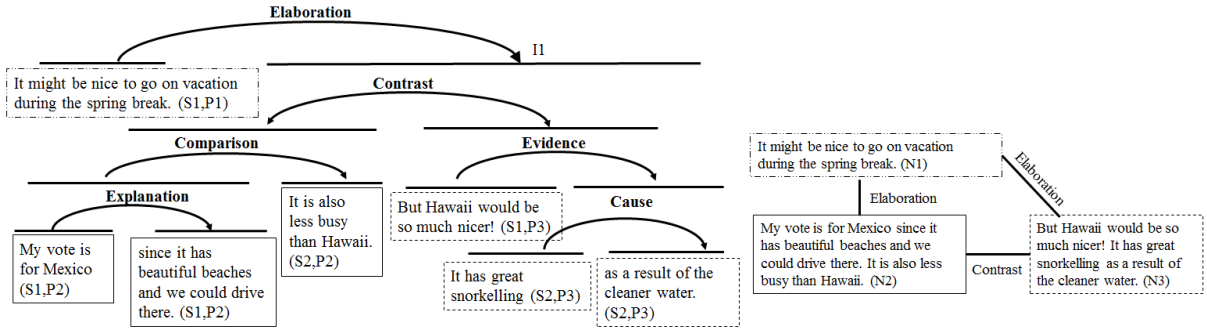
The corpus stems from the online forum Slashdot.[1] Slashdot is a casual internet forum, including sections for users to ask questions, post articles, and review books and games. For the task of disagreement detection, we focus our analysis on the section of the site where users can post articles, and then comment either on the article or respond to other users' posts. This results in a tree-like dialogue structure for which the posted article is the root, and branches correspond to threads of comments. Each comment has a timestamp at the minute resolution as well as author information (although it is possible to post on the forum anonymously). Additionally, other users can give different posts scores (in the range -1 to 5) as well as categorizing posts under "funny", "interesting", "informative", "insightful", "flamebait", "off topic", or "troll". This user moderation, as well as the formalized reply-to structure between comments, makes Slashdot attractive over other internet forums as it allows for high-quality and structured conversations.

In a previous study, Joty et al. (2013) selected 20 articles and their associated comments to be annotated for topic segmentation boundaries and labels by an expert Slashdot contributor. They define a topic as a subset of the utterances in a conversation, while a topic label describes what the given topic is about (e.g., Physics in videogames). Of the 98 annotated topics from their dataset, we filtered out those with only one contributing user, for a total of 95 topics. Next, we developed a Human Intelligence Task (HIT) using the crowdsourcing platform Crowdflower.[2] The objective of this task was to both develop a corpus for testing our disagreement detection system, as well as to investigate how easily human annotators can detect disagreement in casual online forums. For training, users were shown 3 sample topics, labelling them as containing disagreement or not. In each round, annotators were shown 5 topics, with a set of radio buttons for participants to choose "Yes", "No", or "Not sure" in response to asking whether or not the users in the conversation disagree on the topic. In order to limit the number of spam responses, users were shown test questions, which consisted of topics where there was obvious disagreement, as well as topics where there was obviously no disagreement (either agreement, or more news-style information sharing). We required that users correctly identify 4 of these test topics before they were allowed to continue with the task. Users were also shown test questions throughout the task, which, if answered incorrectly, would reduce the amount of money they received for the task, and ultimately disqualify them.

For each topic, five different judgements were obtained. We consider the trust of each partici-

[1] www.slashdot.org

[2] www.Crowdflower.com



(a) Discourse tree **(b)** Relation graph

Figure 1: Discourse tree (left) with extracted relation graph (right) for a sample conversation involving three users with three different posts P1, P2 and P3. N1, N2 and N3 are the corresponding nodes in the relation graph.

pant as the fraction of test questions which they answered correctly. Then, each topic is assigned a score according to a weighted average of the responses, with the weight being the trust of each participant:

$$score = A \sum_{users} (\frac{test_{correct}}{test_{total}})_{user_i} \times (0, 0.5, 1) \quad (1)$$

where 0, 0.5 and 1 represent the answers "No", "Not sure" and "Yes" to the question of disagreement existence, and A is a normalization factor. If the score is less than 0.5, its confidence would be $1 - score$ towards "No disagreement", whereas greater than 0.5 would be a confidence of $score$ towards "Disagreement". The average confidence score across all topics was 0.73. Our corpus consists of 49 topics without disagreement and 46 topics with disagreement. Interestingly, 22 topics had confidence scores below 55%, which suggests that subtle disagreement detection is a subjective and difficult task. Further statistics for the developed corpus are given in Table 1.

4 Features for Disagreement Detection

The features we use in our experiments combine information from conversational structure, rhetorical relations, sentiment features, n-gram models, Slashdot meta-features, structural features, and lexicon features.

4.1 Rhetorical Relation Graphs

Discourse markers have been found to aid in argument and disagreement detection, and for tasks such as stance identification (Abbott et al., 2011; Misra and Walker, 2013; Somasundaran and Wiebe, 2009). We aim to improve over discourse markers by capturing the underlying discourse structure of the conversation in terms of discourse relations.

In Rhetorical Structure Theory, Discourse trees are a hierarchical representation of document structure for monologues (Mann and Thompson, 1988). At the lowest level, Elementary Discourse Units (EDUs) are connected by discourse relations (such as ELABORATION and COMPARISON), which in turn form larger discourse units that are also linked by these relations. Computational systems (discourse parsers) have been recently developed to automatically generate a discourse tree for a given monologue (Joty et al., 2013). Although theoretically the rhetorical relations expected in dialogues are different from those in monologues (Stent and Allen, 2000), no sophisticated computational tools exist yet for detecting these relations reliably. The core idea of this work is that some useful (although noisy) information about the discourse structure of a conversation can be obtained by applying state-of-the-art document level discourse parsing to parts of the conversation.

More specifically, posts on a particular topic are concatenated according to their temporal order. This pseudo-monologic document is then fed to a publicly available document level discourse parser (Joty et al., 2013). A discourse tree such as that seen in Figure 1a is output by the parser. Then, we extract the novel *relation graph* (Figure 1b) from the discourse tree. In this graph, each node (N1, N2, N3) corresponds to a post (P1, P2, P3) and links aim to capture the argumentative structure. There are three cases when a link is added between two nodes in the relation graph. Firstly, links existing between two posts directly, such as the COMPARISON relation between P2 and P3, are added between the corresponding nodes in the re-

lation graph (N2 and N3). Secondly, links existing between fractions of posts in the discourse tree are added to the relation graph (e.g. if (S2,P2) was connected to (S1,P3) directly, N2 and N3 would have an additional link with that label). Finally, when posts are connected through internal nodes (such as P1 and I1 in Figure 1a), a labelled link is added for each post in the internal node to the relation graph (N1->N2 and N1->N3 in Figure 1b).

This relation graph allows for the extraction of many features that may reflect argumentative structure, such as the *number of connections*, the frequency of each rhetorical relation in the graph per post (*diff per post*), and the frequency as a percentage of all rhetorical relations (*diff percentage*). For example, COMPARISON relations are known to indicate disagreement (Horn, 1989), so we expect higher frequencies of this relation if the conversation contains argument. Features from the discourse tree such as the *average depth* of each rhetorical relation are also added to reflect the cohesiveness of conversation. Moreover, features combining the graph and tree representations, such as the ratio of the frequency of a rhetorical relation occurring between different posts to the average depth ($\frac{\text{CONTRAST between different posts}}{\text{Average depth of CONTRAST}}$), called *avg ratio* are implemented. These reflect the hypothesis that relations connecting larger chunks of text (or whole posts) may be more important than those connecting sentences or only partial posts.

Finally, the sub-trees corresponding to individual posts are used to extract the average frequency of rhetorical relations within a post (*same per post*) and the average frequency of a rhetorical relation with respect to other rhetorical relations in the post (*same percentage*). A measure of how often a rhetorical relation connects different users compared to how often it connects discourse units in the same post (*same to diff*), is also added. These capture the intuition that certain rhetorical relations such as CAUSE, EVIDENCE and EXPLANATION are expected to appear more within a post if users are trying to support their perspective in an argument. In total, there are 18 (rhetorical relations)×7 (*avg ratio, avg depth, same per post, same percentage, diff percentage, diff per post, same to diff*) + 1 (number of connections) = 127 features.

4.2 Discourse Markers

Motivated by previous work, we include a frequency count of 17 discourse markers which were found to be the most common across the *ARGUE* corpus (Abbott et al., 2011). Furthermore, we hypothesize that individual discourse markers might have low frequency counts in the text. Therefore, we also include an aggregated count of all 17 discourse markers in each fifth of the posts in a topic (e.g. the count of all 17 discourse markers in the first fifth of every post in the topic). Altogether, there are 5 aggregated discourse marker features in addition to the 17 frequency count features.

4.3 Sentiment Features

Sentiment polarity features have been shown to be useful in argument detection (Mishne and Glance, 2006). For this work, we use four sentiment scoring categories: the *variance*, *average score*, *number of negative sentences*, and *controversiality score* (Carenini and Cheung, 2008) of sentences in a post. These are determined using SoCAL (Taboada et al., 2011), which gives each sentence a polarity score and has been shown to work well on user-generated content.

Overall, we have two main classes of sentiment features. The first type splits all the posts in a topic into 4 sections corresponding to the sentences in each quarter of the post. The sentiment scores described above are then applied to each section of the posts (e.g. one feature is the number of negative sentences in the first quarter of each post). As a separate feature, we also include the scores on just the first sentence, as Mishne and Glance (2006) previously found this to be beneficial. This gives a total of 4×5 = 20 features. We refer to this set as "sentiment".

Motivated by the rhetorical features, our second main class of sentiment features aims to identify "more important" posts for argument detection by applying the four categories of sentiment scores to only those posts connected by each of our 18 rhetorical relations. This is done for both posts with an inner rhetorical connection (identified by the sub-tree for that post), as well as for posts connected by a rhetorical relation in the relation graph. This results in a total of (4 sentiment categories)×(2 (different + same post connections))×(18 rhetorical relations) = 144 features. This set is referred to as "RhetSent".

4.4 Fragment Quotation Graphs

As previously noted in (Murakami and Raymond, 2010; Gomez et al., 2008), the structure of discussion threads can aid in disagreement detection. In online, threaded conversations, the standard approach to extracting conversational structure is through reply-to relations usually present in online forums. However, if users strongly disagree on a topic (or sub-topic), they may choose to quote a specific paragraph (defined as a fragment) of a previous post in their reply. Being able to determine which specific fragments are linked by relations may then be useful for more targeted content-based features, helping to reduce noise. In order to address this, we use the Fragment Quotation Graph (FQG), an approach previously developed by Carenini et al. (2007) for dialogue act modelling and topic segmentation (Joty et al., 2011; Joty et al., 2013).

For our analysis, the FQG is found over the entire Slashdot article. We then select the subgraph corresponding to those fragments in the target topic. From the fragment quotation sub-graph, we are then able to extract features for disagreement detection such as the *number of connections*, *total number of fragments*, and the *average path length* between nodes which we hypothesize to be useful. We additionally extract the *h-index* (Gomez et al., 2008) and *average branching ratio per fragment* of the topic from the simpler reply-to conversational structure. In total, there are 8 FQG features.

4.5 N-gram models

As noted previously (Somasundaran and Wiebe, 2010; Thomas et al., 2006), it is often difficult to outperform a unigram/bigram model in the task of disagreement and argument detection. In this analysis, because of the very small number of samples, we do not consider dependency or part-of-speech features, but do make a comparison with a filtered unigram/bigram model. In the filtering, we remove stop words and any words that occur in fewer than three topics. This helps to prevent topic specific words from being selected, and limits the number of possible matches slightly. Additionally, we use a lexicon of bias-words (Recasens et al., 2013) to extract a bias-word frequency score over all posts in the topic as a separate feature.

4.6 Structural Features

Length features have been well documented in the literature to provide useful information about whether or not arguments exist, especially in online conversations that may be more informative than subjective (Biyani et al., 2014; Yin et al., 2012). In this work, length features include the *length of the post in sentences*, the *average number of words per sentence*, the *average number of sentences per post*, the *number of contributing authors*, the *rate of posting*, and the *total amount of time* of the conversation. This results in a total of 9 features.

4.7 Punctuation

Like many other features already described, frequency counts of '?','!',"","", and '.' are found for each fifth of the post (the first fifth, second fifth, etc.). These counts are then aggregated across all posts for a total of $5 \times 5 = 25$ features.

4.8 Referrals

Referrals have been found to help with the detection of disagreement (Mishne and Glance, 2006), especially with respect to other authors. Since there are no direct referrals to previous authors in this corpus, references to variations of "you", "they", "us", "I", and "he/she" in each fifth of the post are included instead, for a total of $5 \times 5 = 25$ features.

4.9 Meta-Post Features

Slashdot allows users to rank others' posts with the equivalent of a "like" button, changing the "score" of the post (to a maximum of 5). They are also encouraged to tag posts as either "Interesting", "Informative", "Insightful", "Flamebait", "Troll", "Off-topic" or "Funny". Frequency counts of these tags as a percentage of the total number of comments are included as features, as well as the overall fraction of posts which were tagged with any category. The *average score* across the topic, as well as the *number of comments with a score of 4 or 5*, are also added. These are expected to be informative features, since controversial topics may encourage more up and down-voting on specific posts, and generally more user involvement. This results in 9 meta-post features.

Feature Set	Random Forest					SVM				
	P	R	F1	A	ROC-AUC	P	R	F1	A	ROC-AUC
N-grams	0.71	0.57	0.63	0.67	0.69	0.52	0.60	0.56	0.53	0.53
Basic	0.69	0.67	0.68	0.69	0.73	0.62	0.62	0.62	0.63	0.67
Basic+N-grams	0.73	0.66	0.69	0.70	0.73	0.57	0.65	0.60	0.59	0.61
Basic+FQG	0.69	0.66	0.67	0.69	0.71	0.64	0.63	0.63	0.65	0.70
Basic+Sentiment	0.68	0.65	0.66	0.68	0.73	0.61	0.59	0.60	0.62	0.67
Basic+RhetStruct	0.71	0.70	0.70	0.71	0.73	0.73	0.70	**0.71**	**0.73**	**0.78**
Basic+RhetStruct+FQG	0.69	0.69	0.69	0.70	0.73	0.74	0.74	**0.74**	**0.75**	**0.80**
Basic+RhetAll	0.72	0.73	**0.73**	**0.73**	**0.75**	0.76	0.76	**0.76**	**0.77**	**0.79**
RhetStructOnly	0.69	0.72	0.71	0.71	0.72	0.76	0.72	**0.74**	**0.75**	**0.79**
RhetAllOnly	0.69	0.74	0.71	0.71	0.73	0.75	0.72	**0.73**	**0.75**	**0.78**
All	0.71	0.72	**0.71**	**0.72**	0.74	0.74	0.77	**0.75**	**0.76**	**0.77**

Table 2: Basic: Meta-post, all structural, bias words, discourse markers, referrals, punctuation **RhetAll:** Structural and sentiment based rhetorical features **All:** Basic, all rhetorical, sentiment and FQG features. The N-gram models include unigrams and bi-grams. All feature sets in the bottom part of the table include rhetorical reatures.

5 Experiments

Experiments were all performed using the Weka machine learning toolkit. Two different types of experiments were conducted - one using all annotated topics in a binary classification of containing disagreement or not, and one using only those topics with confidence scores greater than 0.75 (corresponding to the more certain cases). All results were obtained by performing 10 fold cross-validation on a balanced test set. Additionally, in-fold cross-validation was performed to determine the optimal number of features to use for each feature set. Since this is done in-fold, a paired t-test is still a valid comparison of different feature sets to determine significant differences in F-score and accuracy.

5.1 Classifiers

Two classifiers were used for this task: Random Forest and SVM. Random Forest was selected because of its ability to avoid over-fitting data despite large numbers of features for relatively few samples. For all runs, 100 trees were generated in the Random Forest, with the number of features to use being determined by in-fold optimization on the F-score. For the SVM classifier, we use the normalized poly-vector kernel with a maximum exponent of 2 (the lowest possible), and a C parameter of 1.0 (Weka's default value). This was chosen to avoid over-fitting our data. We additionally use a supervised in-fold feature selection algorithm, Chi-Squared, to limit over-fitting in the SVM. The number of features to be used is also optimized using in-fold cross-validation on the F-score. Both the SVM classifier and the Random Forest classifier were tested on the same training/testing fold pairs, with a total of 10 iterations.

6 Results

The results of the experiments are shown in Tables 2 and 3. In order to compare to previous analyses, unigram and bigram features are shown, as well as a combination of the basic features with the n-grams. When performing the experiments, we noticed that the n-gram features were hurting the performance of the classifiers when included with most of our other feature sets (or not changing results significantly), and therefore those results are not shown here. As seen in the table,

Feature Set	Random Forest					SVM				
	P	R	F1	A	ROC-AUC	P	R	F1	A	ROC-AUC
N-grams	0.70	0.70	0.70	0.71	0.77	0.63	0.70	0.66	0.63	0.66
Basic	0.74	0.69	0.72	0.72	0.77	0.73	0.70	0.72	0.71	0.78
Basic+FQG	0.72	0.67	0.69	0.69	0.76	0.73	0.63	0.68	0.69	0.76
Basic+Sentiment	0.71	0.65	0.68	0.68	0.76	0.73	0.67	0.70	0.70	0.76
Basic+RhetStruct	0.79	0.75	**0.77**	**0.77**	**0.78**	0.79	0.67	**0.72**	**0.74**	**0.79**
Basic+RhetStruct+FQG	0.76	0.71	0.73	0.73	0.77	0.74	0.64	0.69	0.70	0.78
Basic+RhetAll	0.77	0.75	**0.76**	**0.76**	**0.78**	0.72	0.69	0.71	0.70	0.76
RhetStructOnly	0.75	0.71	0.73	0.73	0.75	0.76	0.63	0.69	0.70	0.76
RhetAllOnly	0.73	0.76	0.74	0.73	0.76	0.67	0.62	0.65	0.65	0.67
All	0.73	0.69	0.71	0.70	0.76	0.71	0.70	0.70	0.69	0.74

Table 3: Precision, recall, F1, accuracy and ROC-AUC scores for the simpler task of identifying the cases deemed "high confidence" in the crowd-sourcing task.

the best performing feature sets are those that include rhetorical features under the SVM+χ^2 classifier. In fact, these feature sets perform significantly better than a unigram/bigram baseline according to a paired t-test between the best classifiers for each set ($p < 0.0001$). The inclusion of rhetorical structure also significantly outperforms the "basic" and "basic+N-grams" feature baselines (which includes discourse markers, referrals, punctuation, bias word counts and structural features) with respect to both the F-score and accuracy ($p < 0.02$ for all feature sets with rhetorical features). Overall, the feature sets "Basic+RhetAll" and "All" under the SVM+χ^2 classifier perform best. This performance is also better than previously reported results for the *ARGUE* Corpus (Abbott et al., 2011), even though the basic and unigram/bigram features perform similarly to that reported in previous analyses.

As an additional check, we also conduct experiments on the "high confidence" data (those topics with a confidence score greater than 0.75). These results are shown in Table 3. Clearly the basic features perform better on this subset of the samples, although the addition of rhetorical structure still provides significant improvement ($p < 0.001$). Overall, this suggests that the rhetorical, sentiment and FQG features help more when the disagreement is more subtle.

Structural	*Length of topic in sentences, total number of authors, quotes in first sentence, quotes in second sentence, questions in first sentence, questions in second sentence, referrals to you and they in first half of post*
Meta	*Number of comments with labels, number of comments labelled 'Flamebait', number of comments with scores of 4 or 5*
FQG	*Number of connections, Number of fragments, Maximum number of links from one node*
RhetStruct	COMPARISON *(same to diff, diff per post, diff percentage, avg ratio, same per post, same percentage),* EXPLANATION *(avg ratio, diff per post),* JOINT *(diff percentage),* ELABORATION *(average depth)*
Discourse Markers	*Aggregated first sentence, Aggregated middle, 'and', 'oh', 'but' frequency counts*
N-grams	*'don't ?', 'plus', 'private', 'anti', 'hey', 'present', 'making', 'developers'*
RhetSent	ELABORATION *(variance in same post),* ATTRIBUTION *(variance in same post),* CONTRAST *(range in same post)*
Sentiment	*Range of sentiment scores in first sentence of all posts, range of sentiment scores over all posts*

Table 4: Features found to be commonly selected over different iterations of the classifiers

7 Analysis and Discussion

In order to examine the quality of our features, we report on the rhetorical features selected, and show that these are reasonable and in many cases, theoretically motivated. Furthermore, we check whether the commonly selected features in each of our feature categories are similar to those found to be useful over the *ARGUE* corpus, as well as within other argument detection tasks in online forums.

The rhetorical features that are consistently selected are very well motivated in the context of argument detection. From the rhetorical structural features, we find COMPARISON relation features to be most commonly selected across all rhetorical feature sets. Other highly ranked features include the proportion of JOINT relations linking different authors, EXPLANATION relations between different authors, and the average depth of ELABORATION relations.

The COMPARISON relations are expected to indicate disagreement as motivated by theoretical considerations (Horn, 1989). The importance of other rhetorical relation features can also be explained by examining conversations in which they appear. In particular, EXPLANATION relations often link authors who share viewpoints in a debate, especially when one author is trying to support the claim of another. The JOINT relations are also very well motivated. In the extreme case, conversations with a very high number of JOINT relations between different users are usually news based. The high proportion of these relations indicates that many users have added information to the conversation about a specific item, such as adding new suggested videogame features to an ongoing list. Fewer JOINT relations seem to indicate disagreement, especially when found in conjunction with COMPARISON relations between different users. This appears to generally indicate that users are taking sides in a debate, and commenting specifically on evidence which supports their viewpoint.

The average depth of ELABORATION relations reveals how deep the perceived connections are between users in a conversation over time. Deeper ELABORATION connections seem to indicate that the conversation is more cohesive. Alone, this does not signify disagreement/agreement but does seem to signify argument-style over news-style dialogues. This is particularly helpful for differentiating between articles with many COMPARISON relations, as COMPARISON may be present in both news-style dialogues (e.g. comparing citation styles) as well as argument style dialogues (e.g. arguing over which of two operating systems is superior).

For the combined sentiment and rhetorical relations, range and variance in ELABORATION, CONTRAST and ATTRIBUTION within the same post are found to be the most informative features. Additionally, neither ATTRIBUTION nor CONTRAST are useful features when only their structural information is considered. In the case of ATTRIBUTION, we hypothesize that the added sentiment score within the post differentiates between a neutral attribution (which would not signify disagreement) and a negative or positive attribution (which may signify disagreement). For CONTRAST, the added sentiment helps to distinguish between responses such as "*We will be trying the Microsoft software. We won't, however, be able to test the Apple equivalent.*" and "*We will be trying the Microsoft software. We won't, however, be trying the inferior Apple equivalent.*" where the second example more likely signals, or even provokes, disagreement.

Outside of the rhetorical features, the discourse markers which are found to be the most useful in our experiments agree with those found in the *ARGUE* corpus (Abbott et al., 2011). Namely, 'oh', 'but', 'because' and 'and' are discovered to be the most informative features. We also find the aggregated discourse marker frequency count in the first part of each post to be useful.

Previous analysis on Slashdot as a social network (Gomez et al., 2008) suggests that the h-index of the conversation is relevant for detecting controversy in a posted article. We include the h-index as part of the Fragment Quotation Graph feature set, but surprisingly do not find this to be a useful feature. This may be due to our corpus involving relatively shallow conversational trees - the maximum h-index across all topics is three.

Comparing to Mishne and Glance's work, we also find quotations, questions and sentiment range near the beginning of a post to be very informative features. These are often selected across all feature sets which include the "basic" set.

The topics most often misclassified across all feature sets are those with relatively few sentences. In these cases, the rhetorical structure is not very well defined, and there is much less content available for detecting quotes, punctuation and referrals. Additionally, the feature sets which only use rhetorical and sentiment features consistently misclassify the same set of conversations (those that have lower quality discourse trees with few connections). When combined with the "basic" feature set, these errors are mitigated, and the topics which the "basic" features miss are picked up by the rhetorical features. This leads to the best overall accuracy and F-score.

7.1 Discourse Parser

A major source of error in detecting disagreement arises because of inaccuracies in our discourse parser. In particular, document-level discourse parsing is a challenging task, with relatively few parsers available at the time of this analysis (Joty et al., 2013; Hernault et al., 2010). We chose to use the discourse parser developed by Joty et al. which both identifies elementary discourse units in a text, and then builds a document-level discourse tree using Conditional Random Fields. Because their approach uses an optimal parsing algorithm as opposed to a greedy parsing algorithm, they are able to achieve much higher accuracies in relation and structure identification than other available parsers. Here, results from their parser on the standard RST-DT dataset are presented since there is no currently available dialogic corpora to compare to.

	RST-DT			Instructional
Metrics	Joty	HILDA	Human	Joty
Span	83.84	74.68	88.70	81.88
Nuclearity	68.90	58.99	77.72	63.13
Relation	55.87	44.32	65.75	43.60

Table 5: Joty et al. document-level parser accuracy of the parser used in this paper. The parser was originally tested on two corpora: RST-DT and Instructional. HILDA was the state-of-the-art parser at that time. Span and Nuclearity metrics assess the quality of the structure of the resulting tree, while the Relation metric assesses the quality of the relation labels.

Examining the relation labels confusion matrix

	T-C	T-O	T-CM	M-M	CMP	EV	SU	CND	EN	CA	TE	EX	BA	CO	JO	S-U	AT	EL
T-C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
T-O	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T-CM	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	0	0	7
M-M	0	0	0	10	0	0	0	0	0	0	0	1	1	0	0	0	1	3
CMP	0	0	0	1	4	0	0	1	0	1	0	3	3	0	1	1	0	2
EV	0	0	0	0	0	0	0	0	0	0	0	2	0	0	2	0	2	11
SU	0	0	0	0	0	0	8	0	0	0	0	0	0	0	1	0	0	12
CND	0	0	0	0	0	0	0	22	0	0	0	0	1	3	0	0	3	2
EN	0	0	0	0	0	0	0	1	24	1	0	0	0	0	0	0	1	7
CA	0	0	0	0	0	0	0	0	2	3	0	4	2	2	7	0	3	11
TE	0	0	0	1	0	0	0	1	2	0	7	1	9	1	9	0	3	4
EX	0	0	0	1	0	0	0	0	1	5	0	12	0	1	3	0	3	12
BA	0	0	0	1	0	0	0	1	0	1	4	1	19	2	6	1	5	12
CO	0	0	0	1	2	0	0	2	0	1	3	2	2	33	7	0	0	9
JO	0	0	0	0	0	0	1	2	0	1	1	1	1	2	57	1	0	13
S-U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	85	1	0
AT	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	3	272	9
EL	0	1	0	0	0	0	0	0	14	6	1	8	1	0	8	2	2	359

Figure 2: Confusion matrix for relation labels on RST-DT. The X-axis represents predicted relations, while the Y-axis corresponds to true values. The relations are Topic-Change (T-C), Topic-Comment (T-CM), Textual Organization (T-O), Manner-Means (M-M), Comparison (CMP), Evaluation (EV), Summary (SU), Condition (CND), Enablement (EN), Cause (CA), Temporal (TE), Explanation (EX), Background (BA), Contrast (CO), Joint (JO), Same-Unit (S-U), Attribution (AT) and Elaboration (EL).

for the discourse parser in Figure 2, some of the chosen rhetorical features make even more sense. In particular, the confusion of ELABORATION and EXPLANATION may account for the perceived importance of ELABORATION relations in the analysis. Likewise, CAUSE (which may be present when users attribute positive or negative qualities to an entity, signalling disagreement) is often confused with JOINT and ELABORATION which were often picked as important features by our classifiers.

8 Conclusions and Future Work

In this paper, we have described a new set of features for detecting disagreement in online blog forums. By treating a written conversation as a series of linked monologues, we can apply a document level discourse parser to extract a discourse tree for the conversation. We then aggregate this information in a *relation graph*, which allows us to capture post-level rhetorical relations between users. Combining this approach with sentiment features shows significantly improved performance in both accuracy and F-score over a baseline consisting of structural and lexical features as well as referral counts, punctuation, and discourse markers. In building our new crowd-sourced corpus from Slashdot, we have also shown the challenges of detecting subtle disagreement in a dataset that contains a significant number of news-style discussions.

In future work, we will improve sentiment features by considering methods to detect opinion-topic pairs in conversation, similar to Somasundaran and Wiebe (2009). Additionally, we will incorporate generalized dependency and POS features (Abbott et al., 2011), which were not used in this analysis due to the very small number of training samples in our dataset. The fragment quotation graph features did not perform as well as we expected, and in future work we would like to investigate this further. Furthermore, we will also explore how to create a discourse tree from the thread structure of a conversation (instead of from its temporal structure), and verify whether this improves the accuracy of the relation graphs, especially when the temporal structure is not representative of the reply-to relationships.

Finally, we plan to apply our novel feature set to other corpora (e.g., *ARGUE*) in order to study the utility of these features across genres and with respect to the accuracy of the discourse parser. This may provide insights into where discourse parsers can be most effectively used, as well as how to modify parsers to better capture rhetorical relations between participants in conversation.

References

Rob Abbott, Marilyn Walker, Pranav Anand, Jean E. Fox Tree, Robeson Bowmani and Joseph King. 2011. How can you say such things?!?: Recognizing Disagreement in Informal Political Argument. In *Proceedings of LSM*, pages 2-11.

Amjad Abu-Jbara, Mona Diab, Pradeep Dasigi, and Dragomir Radev. 2012. Subgroup detection in ideological discussions. In *Proceedings of ACL*, pages 399-409.

Rakesh Agrawal, Sridhar Rajagopalan, Ramakrishnan Srikant, and Yirong Xu. 2003. Mining newsgroups using networks arising from social behavior. In *Proceedings of WWW*, pages 529-535.

Rawia Awadallah, Maya Ramanath, Gerhard Weikum. 2012. Harmony and Dissonance: Organizing the People's Voices on Political Controversies. In *Proceedings of WSDM*, pages 523-532.

Prakhar Biyani, Sumit Bhatia, Cornelia Caragea, Prasenjit Mitra. 2014. Using non-lexical features for identifying factual and opinionative threads in online forums. In *Knowledge-Based Systems*, in press.

Penelope Brown and Stephen Levinson. 1987. *Politeness: Some universals in language usage.* Cambridge University Press.

Giuseppe Carenini and Jackie Cheung. 2008. Extractive vs. NLG-based Abstractive Summarization of Evaluative Text: The Effect of Corpus Controversiality. In *Proceedings of INLG*, pages 33-41.

Giuseppe Carenini, Raymond Ng, Xiaodong Zhou. 2007. Summarizing Email Conversations with Clue Words. In *Proceedings of WWW*, pages 91-100.

Yoonjung Choi, Yuchul Jung, and Sung-Hyon Myaeng. 2010. Identifying controversial issues and their subtopics in news articles. In *Proceedings of PAISI*, pages 140-153.

Alexander Conrad, Janyce Wiebe and Rebecca Hwa. 2012. Recognizing Arguing Subjectivity and Argument Tags. In *ACL Workshop on Extra-Propositional Aspects of Meaning*, pages 80-88.

Jean E. Fox Tree. 2010. Discourse markers across speakers and settings. *Language and Linguistics Compass*, 3(1):1-113.

Michel Galley, Kathleen McKeown, Julia Hirschberg, and Elizabeth Shriberg. 2004. Identifying agreement and disagreement in conversational speech: Use of bayesian networks to model pragmatic dependencies. In *Proceedings of ACL*, pages 669-es.

Sebastian Germesin and Theresa Wilson. 2009. Agreement Detection in Multiparty Conversation. In *Proceedings of International Conference on Multimodal Interfaces* pages 7-14.

Vicenc Gomez, Andreas Kaltenbrunner and Vicente Lopez. 2008. Statistical Analysis of the Social Network and Discussion Threads in Slashdot. In *Proceedings of WWW*, pages 645-654.

Ahmed Hassan, Vahed Qazvinian, and Dragomir Radev. 2010. What's with the attitude?: identifying sentences with attitude in online discussions. In *Proceedings of EMNLP*, pages 1245-1255.

Hugo Hernault, Helmut Prendinger, David A. duVerle and Mitsuru Ishizuka. 2010. HILDA: A Discourse Parser Using Support Vector Machine Classification. *Dialogue and Discourse*, 1(3):1-33.

Laurence R. Horn. 1989. *A natural history of negation.* Chicago University Press.

Shafiq Joty, Giuseppe Carenini, and Chin-Yew Lin. 2011. Unsupervised modeling of dialog acts in asynchronous conversations. In *Proceedings of IJCAI*, pages 1807-1813.

Shafiq Joty, Giuseppe Carenini, Raymond Ng and Yashar Mehdad. 2013. Combining Intra- and Multi-sentential Rhetorical Parsing for Document-level Discourse Analysis. In *Proceedings of ACL*.

Shafiq Joty, Giuseppe Carenini and Raymond Ng. 2013. Topic Segmentation and Labeling in Asynchronous Conversations. *Journal of AI Research*, 47:521-573.

Ching-Sheng Lin, Samira Shaikh, Jennifer Stromer-Galley, Jennifer Crowley, Tomek Strzalkowski, Veena Ravishankar. 2013. Topical Positioning: A New Method for Predicting Opinion Changes in Conversation. In *Proceedings of LASM*, pages 41-48.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243-281.

Gilad Mishne and Natalie Glance. 2006. Leave a reply: An analysis of weblog comments. In *Proceedings of WWW*.

Amita Misra and Marilyn Walker. 2013. Topic Independent Identification of Agreement and Disagreement in Social Media Dialogue. In *Proceedings of SIGDIAL*, pages 41-50.

Arjun Mukherjee and Bing Liu. 2012. Modeling review comments. In *Proceedings of ACL*, pages 320-329.

Akiko Murakami and Rudy Raymond. 2010. Support or Oppose? Classifying Positions in Online Debates from Reply Activities and Opinion Expressions. In *Proceedings of the International Conference on Computational Linguistics*, pages 869-875.

Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, Deborah Cai, Jennifer Midberry, Yuanxin Wang. 2014. Modeling topic control to detect influence in conversations using nonparametric topic models. *Machine Learning* 95:381-421.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting Product Features and Opinions from Reviews. In *Proceedings of HLT/EMNLP*, pages 339-346.

Ana-Maria Popescu and Marco Pennacchiotti. 2010. Detecting controversial events from twitter. In *Proceedings of CIKM*, pages 1873-1876.

Stephan Raaijmakers, Khiet Truong, Theresa Wilson. 2008. Multimodal subjectivity analysis of multiparty conversation. In *Proceedings of EMNLP*, pages 466-474.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic Models for Analyzing and Detecting Biased Language. In *Proceedings of ACL*, pages 16501659.

Swapna Somasundaran, Josef Ruppenhofer, Janyce Wiebe. 2007. Detecting Arguing and Sentiment in Meetings. In *Proceedings of SIGDIAL Workshop on Discourse and Dialogue*.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of ACL*, pages 226-234.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of NAACL, Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116124.

Amanda Stent and James Allen. 2000. Annotating Argumentation Acts in Spoken Dialog. *Technical Report*.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Journal of Computational Linguistics*, 37(2):267-307.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Proceedings of EMNLP*, pages 327-335.

Wen Wang, Sibel Yaman, Kristen Precoda, Colleen Richey, and Geoffrey Raymond. 2011. Detection of agreement and disagreement in broadcast conversations. In *Proceedings of ACL*, pages 374-378.

Bonnie Webber and Rashmi Prasad. 2008. Sentence-initial discourse connectives, discourse structure and semantics. In *Proceedings of the Workshop on Formal and Experimental Approaches to Discourse Particles and Modal Adverbs*.

Jie Yin, Paul Thomas, Nalin Narang, and Cecile Paris. 2012. Unifying local and global agreement and disagreement classification in online debates. In *Proceedings of Computational Approaches to Subjectivity and Sentiment Analysis*, pages 61-69.

+/-EffectWordNet: Sense-level Lexicon Acquisition for Opinion Inference

Yoonjung Choi and **Janyce Wiebe**
Department of Computer Science
University of Pittsburgh
yjchoi, wiebe@cs.pitt.edu

Abstract

Recently, work in NLP was initiated on a type of opinion inference that arises when opinions are expressed toward events which have positive or negative effects on entities (*+/-effect events*). This paper addresses methods for creating a lexicon of such events, to support such work on opinion inference. Due to significant sense ambiguity, our goal is to develop a sense-level rather than word-level lexicon. To maximize the effectiveness of different types of information, we combine a graph-based method using WordNet[1] relations and a standard classifier using gloss information. A hybrid between the two gives the best results. Further, we provide evidence that the model is an effective way to guide manual annotation to find +/-effect senses that are not in the seed set.

1 Introduction

Opinion mining (or sentiment analysis) identifies positive or negative opinions in many kinds of texts such as reviews, blogs, and news articles. It has been exploited in many application areas such as review mining, election analysis, and information extraction. While most previous research focusses on explicit opinion expressions, recent work addresses a type of opinion inference that arises when opinions are expressed toward events which have positive or negative effects on entities (Deng et al., 2013; Deng and Wiebe, 2014). We call such events *+/-effect* events.[2] Deng and Wiebe (2014) show how sentiments toward one entity may be propagated to other entities via opinion inference rules. They give the following example:

> (1) *The bill would curb skyrocketing health care costs.*

The writer expresses an explicit **negative** sentiment (by *skyrocketing*) toward the **object** (*health care costs*). The event, *curb*, has a **negative effect** on *costs*, since they are reduced. We can reason that the writer is **positive** toward the **event** because it has a negative effect on *costs*, toward which the writer is negative. From there, we can reason that the writer is **positive** toward *the bill*, since it is the agent of the positive event. Deng and Wiebe (2014) show that such inferences may be exploited to significantly improve explicit sentiment analysis systems.

However, to achieve its results, the system developed by Deng and Wiebe (2014) requires that all instances of +/-effect events in the corpus be manually provided as input. For the system to be fully automatic, it needs to be able to recognize +/-effect events automatically. This paper addresses methods for creating lexicons of such events, to support such work on opinion inference. We have discovered that there is significant sense ambiguity, meaning that words often have mixtures of senses among the classes *+effect*, *-effect*, and *Null*. Thus, we develop a sense-level rather than word-level lexicon.

One of our goals is to investigate whether the +/-effect property tends to be shared among semantically-related senses, and another is to use a method that applies to all word senses, not just to the senses of words in a given word-level lexicon. Thus, we build a graph-based model in which each node is a WordNet sense, and edges represent semantic WordNet relations between senses. In addition, we hypothesized that glosses also contain useful information. Thus, we develop

[1] WordNet 3.0, http://wordnet.princeton.edu/

[2] While the term *goodFor/badFor* is used in previous papers (Deng et al., 2013; Deng and Wiebe, 2014; Deng et al., 2014), we have since decided that +/-effect is a better term.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1181–1191,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

a supervised gloss classifier and define a hybrid model which gives the best overall performance. Finally, because all WordNet verb senses are incorporated into the model, we investigate the ability of the method to identify unlabeled senses that are likely to be +/-effect senses. We find that by iteratively labeling the top-weighted unlabeled senses and rerunning the model, it may be used as an effective method for guiding annotation efforts.

2 Background

There are many varieties of +/-effect events, including *creation/destruction* (changes in states involving existence), *gain/loss* (changes in states involving possession), and *benefit/injury* (Anand and Reschke, 2010; Deng et al., 2013). The *creation*, *gain*, and *benefit* classes are +effect events. For example, *baking a cake* has a positive effect on the cake because it is created;[3] *increasing the tax rate* has a positive effect on the tax rate; and *comforting the child* has a positive effect on the child. The antonymous classes of each are -effect events: *destroying the building* has a negative effect on the building; *demand decreasing* has a negative effect on demand; and *killing Bill* has a negative effect on Bill.[4]

While sentiment (Esuli and Sebastiani, 2006; Wilson et al., 2005; Su and Markert, 2009) and connotation lexicons (Feng et al., 2011; Kang et al., 2014) are related, sentiment, connotation, and +/-effects are not the same; a single event may have different sentiment and +/-effect polarities, for example. Consider the following example:

> **perpetrate**:
> S: (v) perpetrate, commit, pull (perform an act, usually with a negative connotation) "perpetrate a crime"; "pull a bank robbery"

This sense of *perpetuate* has a **negative** connotation, and is an objective term in `SentiWordNet`. However, it has a **positive effect** on the object, *a crime*, since performing a crime brings it into existence.

As we mentioned, the +/-effect ambiguity cannot be avoided in a word-level lexicon. In the +/-effect corpus of Deng et al. (2013),[5] +/-effect events and their agents and objects are annotated at the word level. In that corpus, 1,411 +/-effect instances are annotated; 196 different +effect words and 286 different -effect words appear in these instances. Among them, 10 words appear in both +effect and -effect instances, accounting for 9.07% of all annotated instances. They show that +/-effect events (and the inferences that motivate this work) appear frequently in sentences with explicit sentiment. Further, **all** instances of +/-effect words that are **not** identified as +/-effect events are false hits from the perspective of a recognition system.

The following is an example of a word with senses of different classes:

> **purge**:
> S: (v) purge (oust politically) "Deng Xiao Ping was purged several times throughout his lifetime" **-effect**
> S: (v) purge (clear of a charge) **+effect**
> S: (v) purify, purge, sanctify (make pure or free from sin or guilt) "he left the monastery purified" **+effect**
> S: (v) purge (rid of impurities) "purge the water"; "purge your mind" **+effect**

This is part of the WordNet output for the word *purge*. In the first sense, the polarity is -effect since it has a negative effect on the object, *Deng Xizo Ping*. However, the other cases have positive effect on the object. Moreover, although a word may not have both +effect and -effect senses, it may have mixtures of ((+effect or -effect) and Null). A purely word-based approach is blind to these cases.

3 Related Work

Lexicons are widely used in sentiment analysis and opinion mining. Several works such as Hatzivassiloglou and McKeown (1997), Turney and Littman (2003), Kim and Hovy (2004), Strapparava and Valitutti (2004), and Peng and Park (2011) have tackled automatic lexicon expansion or acquistion. However, in most such work, the lexicons are word-level rather than sense-level.

[3] Deng et al. (2013) point out that +/-effect objects are not equivalent to benefactive/malefactive semantic roles. An example they give is *She baked a cake for me*: *a cake* is the object of the +effect event *baked* as just noted, while *me* is the filler of its benefactive semantic role (Ziga and Kittil, 2010).

[4] Their annotation manual, which gives additional cases, is available with the annotated data at http://mpqa.cs.pitt.edu/.

[5] Called the *goodFor/badFor* corpus in that paper.

For the related (but different) tasks of developing subjectivity, sentiment and connotation lexicons, some do take a sense-level approach. Esuli and Sebastiani (2006) construct `SentiWordNet`. They assume that terms with the same polarity tend to have similar glosses. So, they first expand a manually selected seed set of senses using WordNet lexical relations such as *also-see* and *direct antonymy* and train two classifiers, one for positive and another for negative. As features, a vectorial representation of glosses is adopted. These classifiers were applied to all WordNet senses to measure positive, negative, and objective scores. In extending their work (Esuli and Sebastiani, 2007), the PageRank algorithm is applied to rank senses in terms of how strongly they are positive or negative. In the graph, each sense is one node, and two nodes are connected when they contain the same words in their WordNet glosses. Moreover, a random-walk step is adopted to refine the scores in their recent work (Baccianella et al., 2010). In contrast, our approach uses WordNet relations and graph propagation in addition to gloss classification.

Gyamfi et al. (2009) construct a classifier to label the subjectivity of word senses. The hierarchical structure and domain information in WordNet are exploited to define features in terms of similarity (using the LCS metric in Resnik (1995)) of target senses and a seed set of senses. Also, the similarity of glosses in WordNet is considered. Even though they investigated the hierarchical structure by LCS values, WordNet relations are not exploited directly.

Su and Markert (2009) adopt a semi-supervised mincut method to recognize the subjectivity of word senses. To construct a graph, each node corresponds to one WordNet sense and is connected to two classification nodes (one for subjectivity and another for objectivity) via a weighted edge that is assigned by a classifier. For this classifier, WordNet glosses, relations, and monosemous features are considered. Also, several WordNet relations (e.g., *antonymy*, *similiar-to*, *direct hypernym*, etc.) are used to connect two nodes. Although they make use of both WordNet glosses and relations, and gloss information is utilized for a classifier, this classifier is generated only for weighting edges between sense nodes and classification nodes, not for classifying all senses.

Kang et al. (2014) present a unified model that assigns connotation polarities to both words and senses. They formulate the induction process as collective inference over pairwise-Markov Random Fields, and apply loopy belief propagation for inference. Their approach relies on selectional preferences of *connotative predicates*; the polarity of a connotation predicate suggests the polarity of its arguments. We have not discovered an analogous type of predicate for the problem we address.

Goyal et al. (2010) generate a lexicon of patient polarity verbs (PPVs) that impart positive or negative states on their patients. They harvest PPVs from a Web corpus by co-occurance with Kind and Evil agents and by bootstrapping over conjunctions of verbs. Riloff et al. (2013) learn positive sentiment phrases and negative situation phrases from a corpus of tweets with hashtag "sarcasm". However, both of these methods are word-level rather than sense-level.

Ours is the first NLP research into developing a sense-level lexicon for events that have negative or positive effects on entities.

4 +/-Effect Word-Level Seed Lexicon and Sense Annotations

To create the corpus used in this work, we developed a word-level seed lexicon, and then manually annotated all the senses of the words in that lexicon.

FrameNet[6] is based on a theory of meaning called Frame Semantics. In FrameNet, a Lexical Unit (LU) is a pairing of a word with a meaning, i.e., it corresponds to a sense in WordNet. Each LU of a polysemous word belongs to a different semantic frame, which is a description of a type of event, relation, or entity and, where appropriate, its participants. For instance, in the **Creating** frame, the definition is that a **Cause** leads to the formation of a **Created_entity**. It has a positive effect on the object, **Created_entity**. This frame contains about 10 LUs such as *assemble*, *create*, *yield*, and so on. FrameNet consists of about 1,000 semantic frames and about 10,000 LUs.

FrameNet is a useful resource to select +/-effect words since each semantic frame covers multiple LUs. We believe that using FrameNet to find +/-effect words is easier than finding +/-effect words without any information since words may

[6]FrameNet, https://framenet.icsi.berkeley.edu/fndrupal/

be filtered by semantic frames. To select +/-effect words, an annotator (who is not a co-author) first identified promising frames as +/-effect and extracted all LUs from them. Then, he went through them and picked out the LUs which he judged to be +effect or -effect. In total, 736 +effect LUs and 601 -effect LUs were selected from 463 semantic frames.

While Deng et al. (2013) and Deng and Wiebe (2014) specifically focus on events affecting objects (i.e., themes), we do not want to limit the lexicon to only that case. Sometimes, events have positive or negative effects on agents or other entities as well. Thus, in this paper, we consider a sense to be +effect (-effect) if it has +effect (-effect) on an entity, which may be the agent, the theme, or some other entity.

In a previous paper (Choi et al., 2014), we conducted a study of the sense-level +/-effect property. For the evaluation, two annotators (who are co-authors of that paper) independently annotated senses of selected words, where some are from pure +effect (-effect) words (i.e., all senses of the words are classified into the same class) and some are from mixed words (i.e., the words have both +effect and -effect senses). In the agreement study, we calculated percent agreement and κ (Artstein and Poesio, 2008), and achieved 0.84 percent agreement and 0.75 κ value.

For a seed set and an evaluation set in this paper, we need annotated sense-level +/-effect data. Mappings between FrameNet and WordNet are not perfect. Thus, we opted to manually annotate the senses of the words in the word-level lexicon. We first extracted all words from 736 +effect LUs and 601 -effect LUs; this extracts 606 +effect words and 537 -effect words (the number of words is smaller than the number of LUs because one word can have more than one LU). Among them, 14 words (e.g., *crush, order,* etc.) are in both the +effect word set and the -effect word set. That is, these words have both +effect and -effect meanings. Recall that this annotator was focusing on frames, not on words - he did not look at all the senses of all the words. As we will see just below, when all the senses of all the words are annotated, a much higher percentage of the words have both +effect and -effect senses. We will also see that many of the senses are revealed to be Null, showing that +effect vs. Null and -effect vs. Null ambiguities are quite prevalent.

A different annotator (a co-author) then went through all senses of all the words from the previous step and manually annotated each sense as to whether it is +effect, -effect, or Null. Note that this annotator participated in an agreement study with positive results in Choi et al. (2014).

For the experiments in this paper, we divided this annotated data into two equal-sized sets. One is a fixed test set that is used to evaluate both the graph model and the gloss classifier. The other set is used as a seed set by the graph model, and as a training set by the gloss classifer. Table 1 shows the distribution of the data. In total, there are 258 +effect senses, 487 -effect senses, and 880 Null senses. To avoid too large a bias toward the Null class,[7] we randomly chose half (i.e., the Null set contains 440 senses). Half of each set is used as seed and training data, and the other half is used for evaluation.

	+effect	**-effect**	**Null**
# annotated data	258	487	880
# Seed/TrainSet	129	243	220
# TestSet	129	244	220

Table 1: Distribution of annotated data.

5 Graph-based Semi-Supervised Learning for WordNet Relations

WordNet (Miller et al., 1990) is organized by semantic relations such as *hypernymy*, *troponymy*, *grouping*, and so on. These semantic relations can be used to build a network. Since the most frequently encoded relation is the super-subordinate relation, most verb senses are arranged into hierarchies; verb senses towards the bottom of the graph express increasingly specific manner. Thus, by following this hierarchical information, we hypothesized that +/-effect polarity tends to propagate. We use a graph-based semi-supervised learning (GSSL) method to carry out the label propagation.

5.1 Graph Formulation

We formulate a graph for semi-supervised learning as follows. Let $G = \{X, E, W\}$ be the undirected graph in which X is the set of nodes, E is the set

[7] As mentioned in the introduction, we want our method to be able to identify unlabeled senses that are likely to be +/-effect senses (see Section 8); we resize the Null class to support this goal.

of edges, and W represents the edge weights (i.e., the weight of edge E_{ij} is W_{ij}). The weight matrix is a non-negative matrix.

Each data point in X = $\{x_1, ... ,x_n\}$ is one sense. The labeled data of X is represented as X_L = $\{x_1, ... ,x_l\}$ and the unlabeled data is represented as X_U = $\{x_{l+1}, ... ,x_n\}$). The labeled data X_L is associated with labels Y_L = $\{y_1, ... ,y_l\}$, where $y_i \in \{1, ..., c\}$ (c is the number of classes). As is typical in such settings, $l \ll n$: n is 13,767, i.e., the number of verb senses in WordNet. Seed/TrainSet in Table 1 is the labeled data.

To connect two nodes, WordNet relations are utilized. We first connect nodes by the hierarchical relations. Since *hypernym* relations represent more general senses and *troponym* relations represent more specific verb senses, we hypothesized that hypernyms or troponyms of a verb sense tends to have its same polarity. *Verb groups* relations that represent verb senses having a similar meaning are also promising. Even though verb-group coverage is not large, its relations are reliable since they are manually grouped. The *entailment* relation is defined as the verb Y is entailed by X if you must be doing Y by doing X. Since pairs connected by this relation are co-extensive, we can assume that both are the same type of event. The *synonym* relation is not used because it is already defined in senses (i.e., each node in the graph is a synset), and the *antonym* relation is also not applied since the weight matrix should be non-negative. The weight value of all edges is 1.0.

5.2 Label Propagation

Given a constructed graph, the label inference (or prediction) task is to propagate the seed labels to the unlabeled nodes. One of the classic GSSL label propagation methods is the local and global consistency (LGC) method suggested by Zhou et al. (2004). The LGC method is a graph transduction algorithm which is sufficiently smooth with respect to the intrinsic structure revealed by known labeled and unlabeled data. The cost function typically involves a tradeoff between the smoothness of the predicted labels over the entire graph and the accuracy of the predicted labels in fitting the given labeled nodes X_L. LGC fits in a univariate regularization framework, where the output matrix is treated as the only variable in optimization, and the optimal solutions can be easily obtained by solving a linear system. Thus, we adopt the LGC method in this paper. Although there are some robust GSSL methods for handling noisy labels, we do not need to handle noisy labels because our input is the annotated data.

Let F be a $n \times c$ matrix to save the output values of label propagation. So, we can label each instance x_i as a label $y_i = \text{argmax}_{j \le c} F_{ij}$ after the label propagation. The initial discrete label matrix Y, which is also $n \times c$, is defined as $Y_{ij} = 1$ if x_i is labeled as $y_i = j$ in Y_L, and $Y_{ij} = 0$ otherwise. The vertex degree matrix $D = \text{diag}([D_{11}, ..., Dnn])$ is defined by $D_{ii} = \sum_{j=1}^{n} W_{ij}$.

LGC defines the cost function Q which integrates two penalty components, global smoothness and local fitting (μ is the regularization parameter):

$$Q = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} \| \frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}} \|^2 + \mu \sum_{i=1}^{n} \|F_i - Y_i\|^2$$

The first part of the cost function is the *smoothness constraint*: a good classifying function should not change too much between nearby points. That is, if x_i and x_j are connected with an edge, the difference between them should be small. The second is the *fitting constraint*: a good classifying function should not change too much from the initial label assignment. The final label prediction matrix F can be obtained by minimizing the cost function Q.

5.3 Experimental Results

Note that, in the rest of this paper, all tables except the last one give results on the same fixed test set (TestSet in Table 1).

We can apply the graph model in two ways.

- **UniGraph**: All three classes (+effect, -effect, and Null) are represented in one graph.
- **BiGraph**: Two separate graphs are first constructed and then combined. One graph is for classifying +effect and Other (i.e., -effect or Null). This graph is called *+eGraph*. The other graph, called *-eGraph*, is for classifying -effect and Other (i.e., +effect or Null).

		UniGraph	BiGraph	BiGraph*
baseline-accuracy		0.411		
accuracy		**0.630**	**0.623**	**0.658**
+effect	P	0.621	0.610	0.642
	R	0.655	0.647	0.680
	F	0.637	0.628	**0.660**
-effect	P	0.644	0.662	0.779
	R	0.720	0.677	0.612
	F	0.680	0.670	**0.686**
Null	P	0.615	0.583	0.583
	R	0.516	0.550	0.695
	F	0.561	0.561	**0.634**

Table 2: Results of UniGraph, BiGraph, and BiGraph*.

These are combined into one model as follows. Nodes that are labeled as +effect by +eGraph and Other by -eGraph are regarded as +effect, and nodes that are labeled as -effect by -eGraph and Other by +eGraph are regarded as -effect. If nodes are labeled as +effect by +eGraph and -effect by -eGraph, they are deemed to be Null. Nodes that are labeled Other by both graphs are also considered as Null.

We had two motivations for experimenting with the BiGraph model: (1) SVM, the supervised learning method used for gloss classification, tends to have better performance on binary classification tasks, and (2) the two graphs of the combined model can "negotiate" with each other via constraints.

In Table 2, we calculate precision (P), recall (R), and f-measure (F) for all three classes. The baseline shown in the top row is the accuracy of a majority class classifier. The first two columns of Table 2 show the results of UniGraph and BiGraph when they are built using the *hypernym*, *troponym*, and *verb group* relations. UniGraph outperforms BiGraph in this experiment.

To improve the results by performing something possible with BiGraph (but not UniGraph), constraints are added when determining the class. As we explained, the label of instance x_i is determined by F_i in the graph. When the label of x_i is decided to be j, we can say that its confidence value is F_{ij}. There are two constraints as follows.

		H+T	+V	+E
+effect	P	0.653	0.642	0.651
	R	0.660	0.680	0.683
	F	0.656	0.660	**0.667**
-effect	P	0.784	0.779	0.786
	R	0.547	0.612	0.604
	F	0.644	**0.686**	0.683
Null	P	0.557	0.583	0.564
	R	0.735	0.695	0.691
	F	**0.634**	**0.634**	0.621

Table 3: Effect of each relation

- If a sense is labeled as +effect (-effect), but the confidence value is less than a threshold, we count it as Null.
- If a sense is labeled as both +effect and -effect by BiGraph, we choose the label with the higher confidence value only if the higher one is larger than a threshold and the lower one is less than a threshold.

The thresholds are determined on Seed/TrainSet by running BiGraph several times with different thresholds, and choosing the one that gives the best performance **on Seed/TrainSet**. (The chosen value is 0.025 for +effect and 0.03 for -effect).

As can be seen in Table 2, BiGraph with constraints (called *BiGraph**) outperforms not only BiGraph without any constraints but also UniGraph. Especially, for BiGraph*, the recall of the Null class is considerably increased, showing that constraints not only help overall, but are particularly important for detecting Null cases.

Table 3 gives ablation results, showing the contribution of each WordNet relation in BiGraph*. With only hierarchical information (i.e., *hypernym* (H) and *troponym* (T) relations), it already shows good performance for all classes. However, they cannot cover some senses. Among the 13,767 verb senses in WordNet, 1,707 (12.4%) cannot be labeled because there are not sufficient hierarchical links to propagate polarity information. When adding the *verb group* (+V) relation, it shows improvement in both +effect and -effect. Especially, the recall for +effect and -effect is significantly increased. In addition, the coverage of the 13,767 verb senses increases to 95.1%. For *entailment* (+E), whereas adding it shows a slight improvement in +effect (and increases coverage by 1.1 percentage points), the

performance is decreased a little bit in the -effect and Null classes. Since the average f-measure for all classes is the highest with *hypernym* (H), *troponym* (T), and *verb group* (V) relations (not *entailment*), we only consider these three relations when constructing the graph.

6 Supervised Learning applied to WordNet Glosses

In WordNet, each sense contains a gloss consisting of a definition and optional example sentences. Since a gloss consists of several words and there are no direct links between glosses, we believe that a word vector representation is appropriate to utilize gloss information as in Esuli and Sebastiani (2006). For that, we adopt an SVM classifier.

6.1 Features

Two different feature types are used.

Word Features (WF): The bag-of-words model is applied. We do not ignore stop words for several reasons. Since most definitions and examples are not long, each gloss contains a small number of words. Also, among them, the total vocabulary of WordNet glosses is not large. Moreover, some prepositions such as *against* are sometimes useful to determine the polarity (+effect or -effect).

Sentiment Features (SF): Some glosses of +effect (-effect) senses contain positive (negative) words. For instance, the definition of {*hurt#4, injure#4*} is "cause damage or affect negatively." It contains a negative word, *negatively*. Since a given event may positively (negatively) affect entities, some definitions or examples already contain positive (negative) words to express this. Thus, as features, we check how many positive (negative) words a given gloss contains. To detect sentiment words, the subjectivity lexicon provided by Wilson et al. (2005)[8] is utilized.

6.2 Gloss Classifier

We have three classes, +effect, -effect, and Null. Since SVM shows better performance on binary classification tasks, we generate two binary classifiers, one (*+eClassifier*) to determine whether a given sense is +effect or Other, and another (*-eClassifier*) to classify whether a given sense is -effect or Other. Then, they are combined as in BiGraph.

[8] Available at http://mpqa.cs.pitt.edu/

6.3 Experimental Results

Seed/TrainSet in Table 1 is used to train the two classifiers, and TestSet is utilized for the evaluation. So, the training set for +eClassifier consists of 129 +effect instances and 463 Other instances, and the training set for -eClassifier contains 243 -effect instances and 349 Other instances. As a baseline, we adopt a majority class classifier.

Table 4 shows the results on TestSet. Performance is better for the -effect than for the +effect class, perhaps because the -effect class has more instances.

When sentiment features (SF) are added, all metric values increase, providing evidence that sentiment features are helpful to determine +/-effect classes.

		WF	WF+SF
baseline accuracy		0.411	
accuracy		**0.509**	**0.539**
+effect	P	0.541	0.588
	R	0.354	0.393
	F	0.428	**0.472**
-effect	P	0.616	0.672
	R	0.500	0.511
	F	0.552	**0.580**
Null	P	0.432	0.451
	R	0.612	0.657
	F	0.507	**0.535**

Table 4: Results of the gloss classifier.

7 Hybrid Method

To use more combined knowledge, the gloss classifier and BiGraph* can be combined. That is, for WordNet gloss information, the gloss classifier is utilized, and for WordNet relations, BiGraph* is used. With the Hybrid method, we can see not only the effect of propagation by WordNet relations but also the usefulness of gloss information and sentiment features. Also, while BiGraph* cannot cover all senses in WordNet, the Hybrid method can.

The outputs of the gloss classifier and BiGraph* are combined as follows. The label of the gloss classifier is one of +effect, -effect, Null, or Both (when a given sense is classified as both +effect by +eClassifier and -effect by -eClassifier). Possible labels of BiGraph* are +effect, -effect, Null, Both, or None (when a given sense is not

labeled by BiGraph*). There are five rules:

- If both labels are +effect (-effect), it is +effect (-effect).
- If one of them is Both and the other is +effect (-effect), it is +effect (-effect).
- If the label of BiGraph* is None, believe the label of the gloss classifier
- If both labels are Both, it is Null
- Otherwise, it is Null

The results for Hybrid are given in the first row of the lower half of Table 5; the results for BiGraph* are in the first row of the upper half, for comparison. Generally, the Hybrid method shows better performance than the gloss classifier and BiGraph*. In the Hybrid method, since more +/-effect senses are detected than by BiGraph*, while precision is decreased, recall is increased by more. However, by the same token, the overall performance for the Null class is decreased. Actually, that is expected since the Null class is determined by the Other class in the gloss classifier and BiGraph*. Through this experiment, we see that the Hybrid method is better for classifying +/-effect senses.

7.1 Model Comparison

To provide evidence for our assumption that different models are needed for different information to maximize effectiveness, we compare the hybrid method with the supervised learning and the graph-based learning (GSSL) methods, each utilizing both WordNet relations and gloss information.

Supervised Learning (*onlySL*): The gloss classifier is trained with word features and sentiment features for WordNet Gloss. To exploit WordNet relations in supervised learning, especially the hierarchical information, we use least common subsumer (LCS) values as in Gyamfi et al. (2009), which, recall, performs supervised learning of subjective/objective senses. The values are calculated as follows. For a target sense t and a seed set S, the maximum LCS value between a target sense and a member of the seed set is found as:

$$Score(t, S) = max_{s \in S} LCS(t, s)$$

With this LCS feature and the features described in Section 6, we run SVM on the same training and test data. For LCS values, the similarity using the information content proposed by Resnik (1995) is measured. WordNet Similarity[9] package provides pre-computed pairwise similarity values for that.

Table 6 shows results of onlySL. Compared to Table 4, while +effect and Null classes show a slight improvement, the performance is degraded for -effect. This means that the added feature is rather harmful to -effect. Even though the hierarchical feature is very helpful to expand +/-effect, it is not helpful for onlySL since SVM cannot capture propagation according to the hierarchy.

Graph-based Learning (*onlyGraph*): In Section 5, the graph is constructed by using WordNet relations. To apply WordNet gloss information in onlyGraph, we calculate a cosine similarity between glosses. If the similarity value is higher than a threshold, two nodes are connected with this similarity value. The threshold is determined by training and testing on Seed/TrainSet (the chosen value is 0.3).

Comparing Tables 2 and 6, BiGraph* generally outperforms onlyGraph (the exception is precision of +effect). By gloss similarity, many nodes are connected to each other. However, since uncertain connections can cause incorrect propagation in the graph, this negatively affects the performance.

Through this experiment, we see that since each type of information has a different character, we need different models to maximize the effectiveness of each type. Thus, the hybrid method with different models can have better performance.

		Hybrid	**onlySL**	**onlyGraph**
+effect	P	0.610	0.584	0.701
	R	0.735	0.400	0.364
	F	**0.667**	0.475	0.480
-effect	P	0.717	0.778	0.651
	R	0.669	0.316	0.562
	F	**0.692**	0.449	0.603
Null	P	0.556	0.440	0.473
	R	0.520	0.813	0.679
	F	0.538	**0.571**	0.557

Table 6: Comparison to onlySL and onlyGraph.

[9] WordNet Similarity, http://wn-similarity.sourceforge.net/

		+effect			-effect			Null		
		P	R	F	P	R	F	P	R	F
BiGraph*	**Initial**	0.642	0.680	**0.660**	0.779	0.612	**0.686**	0.583	0.695	**0.634**
	1st	0.636	0.684	0.663	0.770	0.632	0.694	0.591	0.672	0.629
	2nd	0.642	0.701	0.670	0.748	0.656	0.699	0.605	0.655	0.629
	3rd	0.636	0.708	0.670	0.779	0.652	0.710	0.599	0.669	0.632
	4th	0.681	0.674	**0.678**	0.756	0.674	**0.712**	0.589	0.669	0.626
Hybrid	**Initial**	0.610	0.735	**0.667**	0.717	0.669	**0.692**	0.556	0.520	**0.538**
	1st	0.614	0.713	0.672	0.728	0.681	0.704	0.562	0.523	0.542
	2nd	0.613	0.743	0.672	0.716	0.697	0.706	0.559	0.497	0.526
	3rd	0.616	0.739	0.672	0.717	0.706	0.712	0.559	0.494	0.525
	4th	0.688	0.681	**0.684**	0.712	0.764	**0.732**	0.565	0.527	0.545

Table 5: Results of an iterative approach.

8 Guided Annotation

Recall that Seed/TrainSet and TestSet, the data used so far, are all the senses of the words in a word-level +/-effect lexicon. This section presents evidence that our method can guide annotation efforts to find other words that have +/-effect senses. A bonus is that the method pinpoints particular +/-effect senses of those words.

All unlabeled data are senses of words that are not included in the original lexicon. Since presumably the majority of verbs do not have any +/-effect senses, a sense randomly selected from WordNet is very likely to be Null. We explore an iterative approach to guided annotation, using BiGraph* and Hybrid as the method for assigning labels.

The system is initially created as described above using Seed/TrainSet as the initial seed set. Each iteration has four steps: 1) rank all unlabeled data (i.e., the data other than TestSet and the current seed set) based on the F_{ij} confidence values (see Section 5.3); 2) choose the top 5% and manually annotate them (the same annotator as above did this); 3) add them to the seed set; 4) rerun the system using the expanded seed set. We performed four iterations in this paper.

The upper and lower parts of Table 5 show the intial results and the results after each iteration for BiGraph* and Hybrid. Recall that these are results on the fixed set, TestSet. Overall for both models, f-measure increases for both the +effect and -effect classes as more seeds are added, mainly due to improvements in recall. The evaluation on the fixed set is also useful in the annotation process because it trades off +/-effect vs. Null annotations. If the new manual annotations were biased, in that they incorrectly label Null senses as +/-effect, then the f-measure results would instead degrade on the fixed TestSet, since the system is created each time using the increased seed set.

We now consider the accuracy of the system on the newly labeled annotated data in Step 2. Note that our method is similar to Active Learning (Tong and Koller, 2001), in that both automatically identify which unlabeled instances the human should annotate next. However, in active learning, the goal is to find instances that are difficult for a supervised learning system. In our case, the goal is to find needles in the haystack of WordNet senses. In Step 3, we add the newly labeled senses to the seed set, enabling the model to find unlabeled senses close to the new seeds when the system is rerun for the next iteration.

We assess the system's accuracy on the newly labeled data by comparing the system's labels with the human's new labels. Accuracy for +effect and -effect is calculated such as:

$$Accuracy_{+effect} = \frac{\text{\# annotated +effect}}{\text{\# top 5\% +effect data}}$$

$$Accuracy_{-effect} = \frac{\text{\# annotated -effect}}{\text{\# top 5\% -effect data}}$$

That is, the accuracy means that out of the top 5% of the +effect (-effect) data as scored by the system, what percentage are correct as judged by a human annotator. Table 7 shows the accuracy for each iteration in the top part and the number of senses labeled in the bottom part. As can be seen, the accuracies range between 60% and 78%; these

values are much higher than what would be expected if labeling senses of words randomly chosen from WordNet.[10] The annotator spent, on average, approximately an hour to label 100 senses. For finding new words with +/-effect usages, it would be much more cost-effective if a significant percentage of the data chosen for annotation are senses of words that in fact have +/-effect senses.

	1st	2nd	3rd	4th
+effect	65.63%	62.50%	63.79%	59.83%
-effect	73.55%	73.97%	77.78%	70.30%
+effect	128	122	116	117
-effect	155	146	153	145
total	283	268	269	262

Table 7: Accuracy and frequency of the top 5% for each iteration

9 Conclusion and Future Work

In this paper, we investigated methods for creating a sense-level +/-effect lexicon. To maximize the effectiveness of each type of information, we combined a graph-based method using WordNet relations and a standard classifier using gloss information. A hybrid between the two gives the best results. Further, we provide evidence that the model is an effective way to guide manual annotation to find +/-effect words that are not in the seed word-level lexicon. This is important, as the likelihood that a random WordNet sense (and thus word) is +effect or -effect is not large.

So as not to limit the inferences that may be drawn, our annotations include events that are +effect or -effect either the agent or object. In future work, we plan to exploit corpus-based methods using patterns as in Goyal et al. (2010) combined with semantic role labeling to refine the lexicon to distinguish which is the affected entity. Further, to actually exploit the acquired lexicon to process corpus data, an appropriate coarse-grained sense disambiguation process must be added, as Akkaya et al. (2009) and Akkaya et al. (2011) did for subjective/objective classification.

We hope the general methodology will be effective for other semantic properties. In opinion mining and sentiment analysis this is particularly needed, because different meanings of *positive* and *negative* are appropriate for different applications. This is a way to create lexicons that are customized with respect to one's own definitions.

It would be promising to combine our method with other methods to enable it to find +effect and -effect senses that are outside the coverage of WordNet. However, a WordNet-based lexicon gives a substantial base to build from.

[10]For reference, in 5th iteration, the +effect accuracy is 60.18% and the -effect accuracy is 69.93%, and in 6th iteration, the +effect accuracy is 59.81% and the -effect accuracy is 69.12%.

Acknowledgments

This work was supported in part by DARPA-BAA-12-47 DEFT grant #12475008 and National Science Foundation grant #IIS-0916046. We would like to thank the reviewers for their helpful suggestions and comments.

References

Cem Akkaya, Janyce Wiebe, and Rada Mihalcea. 2009. Subjectivity word sense disambiguation. In *Proceedings of EMNLP 2009*, pages 190–199.

Cem Akkaya, Janyce Wiebe, Alexander Conrad, and Rada Mihalcea. 2011. Improving the impact of subjectivity word sense disambiguation on contextual opinion analysis. In *Proceedings of CoNLL 2011*, pages 87–96.

Pranna Anand and Kevin Reschke. 2010. Verb classes as evaluativity functor classes. In *Interdisciplinary Workshop on Verbs. The Identification and Representation of Verb Features*.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Comput. Linguist.*, 34(4):555–596.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of LREC*, pages 2200–2204.

Yoonjung Choi, Lingjia Deng, and Janyce Wiebe. 2014. Lexical acquisition for opinion inference: A sense-level lexicon of benefactive and malefactive events. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*, pages 107–112. Association for Computational Linguistics.

Lingjia Deng and Janyce Wiebe. 2014. Sentiment propagation via implicature constraints. In *Proceedings of EACL*.

Lingjia Deng, Yoonjung Choi, and Janyce Wiebe. 2013. Benefactive/malefactive event and writer attitude annotation. In *Proceedings of 51st ACL*, pages 120–125.

Lingjia Deng, Janyce Wiebe, and Yoonjung Choi. 2014. Joint inference and disambiguation of implicit sentiments via implicature constraints. In *Proceedings of COLING*, page 7988.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of 5th LREC*, pages 417–422.

Andrea Esuli and Fabrizio Sebastiani. 2007. Pageranking wordnet synsets: An application to opinion mining. In *Proceedings of ACL*, pages 424–431.

Song Feng, Ritwik Bose, and Yejin Choi. 2011. Learning general connotation of words using graph-based algorithms. In *Proceedings of EMNLP*, pages 1092–1103.

Amit Goyal, Ellen Riloff, and Hal DaumeIII. 2010. Automatically producing plot unit representations for narrative text. In *Proceedings of EMNLP*, pages 77–86.

Yaw Gyamfi, Janyce Wiebe, Rada Mihalcea, and Cem Akkaya. 2009. Integrating knowledge for subjectivity sense labeling. In *Proceedings of NAACL HLT 2009*, pages 10–18.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of ACL*, pages 174–181.

Jun Seok Kang, Song Feng, Leman Akoglu, and Yejin Choi. 2014. Connotationwordnet: Learning connotation over the word+sense network. In *Proceedings of the 52nd ACL*, page 15441554.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of 20th COLING*, pages 1367–1373.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 13(4):235–312.

Wei Peng and Dae Hoon Park. 2011. Generate adjective sentiment dictionary for social media sentiment analysis using constrained nonnegative matrix factorization. In *Proceedings of ICWSM*.

Philip Resnik. 1995. Using information content to evaluate semantic similarity. In *Proceedings of 14th IJCAI*, pages 448–453.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of EMNLP*, pages 704–714.

Carlo Strapparava and Alessandro Valitutti. 2004. Wordnet-affect: An affective extension of wordnet. In *Proceedings of 4th LREC*, pages 1083–1086.

Fangzhong Su and Katja Markert. 2009. Subjectivity recognition on word senses via semi-supervised mincuts. In *Proceedings of NAACL HLT 2009*, pages 1–9.

Simon Tong and Daphne Koller. 2001. Support vector machin active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66.

Peter Turney and Michael Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 21(4):315–346.

Theresa Wilson, Janyce Wiebe, , and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT-EMNLP*, pages 347–354.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Scholkopf. 2004. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16:321–329.

Fernando Ziga and Seppo Kittil. 2010. *Benefactives and malefactives, Typological perspectives and case studies*. John Benjamins Publishing.

A Sentiment-aligned Topic Model for Product Aspect Rating Prediction

Hao Wang
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
hwa63@sfu.ca

Martin Ester
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
ester@sfu.ca

Abstract

Aspect-based opinion mining has attracted lots of attention today. In this paper, we address the problem of product aspect rating prediction, where we would like to extract the product aspects, and predict aspect ratings simultaneously. Topic models have been widely adapted to jointly model aspects and sentiments, but existing models may not do the prediction task well due to their weakness in sentiment extraction. The sentiment topics usually do not have clear correspondence to commonly used ratings, and the model may fail to extract certain kinds of sentiments due to skewed data. To tackle this problem, we propose a sentiment-aligned topic model(SATM), where we incorporate two types of external knowledge: product-level overall rating distribution and word-level sentiment lexicon. Experiments on real dataset demonstrate that SATM is effective on product aspect rating prediction, and it achieves better performance compared to the existing approaches.

1 Introduction

Online reviews have become an important source of information for consumers. People tend to read reviews to help them compare products, and make informed decisions. As the volume of product reviews continues to grow, it is often impossible to read all of them, which calls for efficient methods for opinion mining. Nowadays, for each product, many websites aggregate the overall rating of reviews, and display its distribution. However, this cannot provide detailed information. For example, two products may have similar overall rating distributions, while people talk about different unsatisfactory aspects. This problem has inspired a new line of research on aspect-level opinion mining(Hu and Liu, 2004).

An aspect refers to a rateable feature, such as staff and location in hotel reviews, or size and battery for digital camera reviews. In this paper, we deal with the problem of *product aspect rating prediction*. The input is a collection of products, and each product is associated with a set of reviews. The goal is to extract the corpus-level aspects, and predict the aspect ratings for each product. This kind of fine-grained sentiment analysis will help users efficiently digest the reviews, and gain more insight into the product quality.

The product aspect rating prediction problem usually involves two subtasks: aspect extraction and sentiment identification(Titov and McDonald, 2008b). Given some text, we would like to know what aspects it talks about, and what kind of sentiments are expressed. For example, given a sentence "the room is filthy", we would like to know that it talks about the aspect "room". Also, "filthy" is a sentiment word, and it expresses strongly negative sentiment towards the aspect "room".

Topic models(Blei et al., 2003; Hofmann, 1999) have been popular in aspect-based opinion mining(Liu, 2012). Existing works have used topic models to extract only aspects(Titov and McDonald, 2008a; Brody and Elhadad, 2010; Chen et al., 2013), or jointly model aspects and sentiments (Mei et al., 2007; Lin and He, 2009; Li et al., 2010; Jo and Oh, 2011; Moghaddam and Ester, 2011; Lakkaraju et al., 2011; Sauper et al., 2011; Mukherjee and Liu, 2012; Lazaridou et al., 2013; Moghaddam and Ester, 2013; Kim et al., 2013). In the joint modelling approaches, a sentiment topic is usually modelled as a sentiment label-word distribution, analogous to the topic-word distribution in standard topic models. However, the difference is that the sentiment topics need to be ordered. If the model is to be applied for aspect rating prediction, the sentiment topics should have clear cor-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1192–1202,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

respondence to the ratings. Suppose there are 5 sentiment topics with sentiment labels from 1 to 5. The sentiment topic with label i is expected to correspond to the rating i on the 1-5 rating scale. For example, the sentiment topic with label 5 should have high probability over positive sentiment words, so it expresses highly positive sentiment, which matches our natural interpretation for the rating 5. In this case, sentiment labels and ratings are *aligned*. However, in a standard topic model, the learned sentiment topics may not have clear correspondence with different ratings. Also, if the positive reviews are dominant in the data, the topic model may fail to capture the negative sentiments with any sentiment topic, so no sentiment labels are matched with low ratings. If the sentiment labels are not correctly aligned to the ratings, we cannot use these sentiment labels to predict aspect ratings. Consequently, the aspect rating prediction accuracy is compromised, and the method is less practical. We call this the *sentiment label alignment* problem. To tackle this problem, models in the literature usually use some seed words for each sentiment topic to define Dirichlet priors with asymmetric concentration parameter vectors (Sauper et al., 2011; Kim et al., 2013), or use seed words to initialize word assignment to sentiment topic(Lin and He, 2009), or both(Li et al., 2010; Jo and Oh, 2011). However, these seed words are usually arbitrarily selected, and how to define asymmetric priors is not clear, especially when we would like to capture more than two (positive and negative) kinds of sentiments.

In this paper, we propose a sentiment-aligned topic model(SATM) for product aspect rating prediction, which focuses the sentiment label alignment problem. We use two kinds of external knowledge: the product overall rating distribution, and a sentiment lexicon. For each product, the overall rating distribution is available on most online review websites. It provides the big picture of the product-level sentiments. In SATM, for each product and each aspect, we define a multinomial distribution over sentiment labels, with prior parameterized by the overall rating distribution. Sentiment lexicon is constructed by linguistic experts, and every word in the lexicon is associated with a sentiment polarity score(Taboada et al., 2011). We treat the polarity score as an extra word feature in a semi-supervised framework. By incorporating both product-level and word-level knowledge into the model, the sentiment labels can be aligned with ratings, and the extracted sentiment topics can capture different kinds of sentiments, ranging from highly positive to highly negative. Experiments on a TripAdvisor dataset demonstrate that our method can effectively deal with the sentiment label alignment problem, and outperforms state-of-the-art methods in terms of product aspect rating prediction accuracy.

2 Related work

Several methods have been proposed for product aspect rating prediction, and many of them are based on topic models.

In (Lu et al., 2009), the authors studied the problem of generating an aspect rating summary for short comments. The text was first preprocessed into phrases of the format <headterm, sentiment word>, and the headterms are clustered by Structured PLSA to find K major aspects. Then, phrase ratings are predicted by either Local Prediction or Global Prediction, and they are aggregated to get aspect ratings. The method in (Brody and Elhadad, 2010) also first uses topic models to find aspects. Then, for each aspect, it extracts all the relevant adjectives, and builds a conjunction graph. A label propagation algorithm(Zhu and Ghahramani, 2002) is used on the graph to learn the sentiment polarity score of adjective words. Although this approach is not proposed for aspect rating prediction, it can be used for this task if the polarity scores of adjective words are aggregated for each aspect. All the methods above perform aspect extraction and sentiment identification separately, while our approach takes a joint modelling approach so that different subtasks can potentially reinforce with each other. To demonstrate this, we use these methods as baselines in our experiments.

Wang et al. worked on the *Latent Aspect Rating Analysis* problem(Wang et al., 2010; Wang et al., 2011), the task of inferring aspect ratings for each review and the relative weights reviewers have placed on each aspect. In (Wang et al., 2010), aspect keywords are provided as user input, and a two-stage method, called Latent Rating Regression(LRR), is proposed. The first stage uses a bootstrapping algorithm to obtain more related words for each aspect, and segments the document content. In the second stage, the overall rating is "generated" as weighted combination of the latent aspect ratings, and LRR is used to infer both the

weights and aspect ratings. Their follow-up work (Wang et al., 2011) does not need keyword specification from users, and replaces the bootstrapping method with a topic model. However, both methods implicitly require that each review talks about all aspects, which is not always true due to the data sparsity in online reviews.

In (Moghaddam and Ester, 2011), ILDA was proposed for product aspect rating prediction. Later, it was extended to FLDA (Moghaddam and Ester, 2013) to address the cold start problem, when there are few reviews associated with a product. Similar to (Lu et al., 2009), in ILDA and FLDA, a preprocessing step parses the text into phrases of the format <headterm, sentiment word>, and a review is modelled as a bag of phrases. We also adopt this assumption in our model. The method in (Sauper et al., 2011; Sauper and Barzilay, 2013) does not use phrases, but instead uses "snippets", and an snippet is a short sentence or phrase. However, the sentiment label alignment problem is not well addressed in these models, which limits their practicality. ILDA and FLDA did not deal with this problem. The model in (Sauper et al., 2011; Sauper and Barzilay, 2013) follows the most common approach of using seed words to define asymmetric priors. It supports only two kinds of sentiment topics: positive and negative, while how to define asymmetric priors for more sentiment topics becomes unclear. More importantly, the prior approach may not work well in practice(see Experiment Section). Lakkaraju et al. try to tackle the sentiment label alignment problem by assuming that the overall rating is generated as response variable(Lakkaraju et al., 2011), with the sentiment topic proportions as features. However, how the sentiment labels are related to ratings is still unknown until learned, and we may not get the desired alignment. Lazaridou et al. attempt to connect sentiment labels with ratings by Kronecker symbol, but this method only applies to three sentiment polarities: -1(negative), 0(neutral), $+1$(positive), and it does not explore the word-level lexicon, which is also an important source of knowledge.

Another line of research on product aspect rating prediction or summarization does not use topic models, but relies mainly on word frequency and grammatical relations(Hu and Liu, 2004; Popescu and Etzioni, 2005; Blair-goldensohn et al., 2008), or specialized review selection(Long et al., 2014). In this case, the extracted aspect words need to be clustered manually. For example, *picture* and *photo* may refer to the same aspect in digital camera reviews. By comparison, topic modelling approaches extract aspect words and cluster them simultaneously.

Our method incorporates the product overall rating distributions and sentiment lexicons into the model, so it is also related to topic models which use observed features or domain knowledge(Mimno and McCallum, 2008; Andrzejewski et al., 2009; Andrzejewski et al., 2011). Mimno et al. introduces two general frameworks to integrate observed features into the generative process: downstream and upstream topic models(Mimno and McCallum, 2008). In the context of aspect-based opinion mining, MaxEnt-LDA(Zhao et al., 2010) integrates a discriminative maximum entropy component to help separate aspect words and sentiment words. The SAS model (Mukherjee and Liu, 2012) uses seed words to provide guidance for aspect discovery, and MC-LDA (Chen et al., 2013) uses must-links and cannot-links to extract coherent aspects. However, MaxEnt-LDA, SAS and MC-LDA cannot be used for aspect rating prediction, since they fail to identify the sentiment polarity of sentiment words.

3 Method

3.1 Preliminaries

We first introduce several key concepts used in our model.

Products: Let $P = \{P_1, P_2, \dots\}$ be a set of products. Each product P_i is associated with a set of reviews $D_i = \{d_1, d_2, \dots d_{N_i}\}$, and also an overall rating distribution Y_i. Y_i is a multinomial distribution on R ratings. It is available on most online review websites, and usually $R = 5$.

Aspects: An aspect is a rateable feature of a product, and each aspect is modelled as a distribution over aspect words. The number of aspects is predefined as K.

Sentiment topics: A sentiment topic is modelled as a distribution over sentiment words, and each sentiment topic is associated with a sentiment label. To make it consistent with commonly used rating scale, we assume there are R sentiment labels, corresponding to the R ratings. The challenge is that sentiment labels with higher values are expected to be associated with sentiment topics which express more positive sentiments, so that

we can match sentiment labels with ratings.

Phrases: An opinion phrase $f =< h, m >$ is a pair of aspect word h and sentiment word m, such as $< room, filthy >$(Lu et al., 2009; Moghaddam and Ester, 2011). For each product P_i, we first parse the related reviews D_i into phrases F_i, and each product can be modelled as a bag of phrases.

Sentiment lexicons : A sentiment lexicon L is a list of sentiment words, and each word $m \in L$ is associated with a sentiment polarity score s_m. s_m can take T values. Note that the lexicon L usually only covers a small subset of sentiment words in the whole vocabulary.

Sentiment association: The sentiment label takes R values, and there are T different values for the polarity score in the sentiment lexicon. However, the relation between sentiment labels and polarity scores are unknown. If we have training instances where a sentiment word m is associated with both a sentiment label r_m and polarity score s_m, we can build a classifier, where the explanatory variable for the classifier is a sentiment label, and outcome is the polarity score. In this case, $H(s_m|r_m)$ can be interpreted as the probability of observing a polarity score s_m, given its sentiment label r_m. We refer to this probability H as sentiment association. This is a key component in our model. It naturally bridges the gap between sentiment labels and polarity scores, and captures the uncertainty in their relations. Note that H can be trained independent of the topic model part. For each training instance, suppose the sentiment word is $m \in L$, we need to know its sentiment label r_m and polarity score s_m. s_m can be retrieved directly from the sentiment lexicon, and r_m can be either manually or automatically annotated. For example, suppose the word m appears in review d, we can assign the overall rating of d as its sentiment label. In this case, each word $m \in L$ can be associated with multiple training instances that have the same value for s_m but different sentiment labels r_m. We adopt this approach to automatically annotate sentiment labels, and details are described in the Experiments section.

3.2 Problem definition

The product aspect rating prediction problem can be defined as follows. The input is a set of products P. Each product P_i has a bag of phrases F_i, and an overall rating distribution Y_i over R ratings. The output is the K corpus-level aspects, and for each product, we predict its ratings on the K aspects, also in the $[1, R]$ rating scale. We assume products in P are in the same category so they share the same aspects.

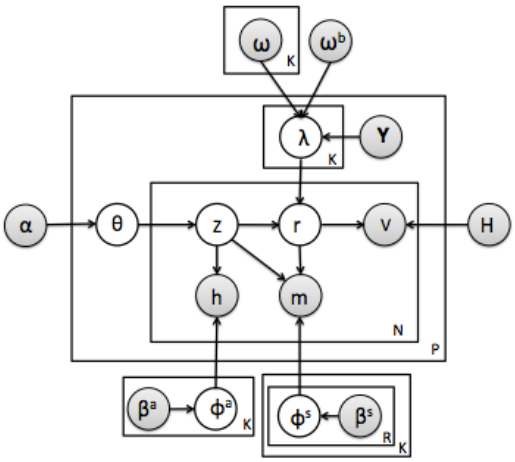


Figure 1: Graphical model of SATM

3.3 The SATM model

We introduce the Sentiment-aligned Topic Model(SATM) in this section, and its graphical representation is shown in Figure 1. Note that the sentiment association H is observed, because it is trained independently of the topic model part.

At the word level, each observed phrase $< h, m >$ is associated with two latent variables: aspect z and sentiment label r. Aspect z models what aspect this phrase talks about, and r determines the sentiment of m. If m is in the sentiment lexicon, we assume r is also responsible for generating a word feature v_m, based on the sentiment association H, which is equal to its polarity score s_m in the lexicon. In this case, the observed data becomes $(< h, m >, v_m)$, and the latent sentiment label r is responsible for generating both word m, and word feature v_m. For example, for the phrase <room, filthy>, we observe a word feature $v = -5$, since the sentiment polarity score for the word "filthy" is -5. Given H, sentiment labels 1 or 2 are more likely to generate a word feature -5. Also, people tend to use "filthy" to express low ratings, like 1 or 2, so the sentiment labels and ratings can be aligned.

At the product level, for each product p and each aspect k, we define a multinomial distribution $\lambda_{p,k}$ over R sentiment labels. Since Y_p already gives us the big picture about the overall senti-

ment expressed on this product, we assume $\lambda_{p,k}$ is drawn from a dirichlet distribution $Dir(\boldsymbol{\pi_{p,k}})$ with asymmetric concentration parameters, where $\boldsymbol{\pi_{p,k}} = f(\boldsymbol{Y_p}, \boldsymbol{\omega_k}, \omega^b)$. We can use a linear parametrization, and set

$$f(\boldsymbol{Y_p}, \boldsymbol{\omega_k}, \omega^b) = \omega_k^1 \boldsymbol{Y_p} + \omega_k^0 + \omega^b \quad (1)$$

ω_k^1 captures the influence of the product overall rating distribution, and can favour certain sentiment labels in the prior. ω_k^0 and ω^b are the aspect-specific and corpus-level bias, respectively. Through this linear parametrization, we build a direct matching between sentiment label i and rating i. For example, for a product p, if its overall rating distribution Y_p has high probability over rating 4, for aspect k, we assume its product-aspect-sentiment label distribution also has high probability on sentiment label 4 in the prior. The actual aspect rating is affected by both the text which talks about aspect k, and also the prior.

To sum up, we assume the generative process as follows:

- For each aspect $k = 1, 2, \ldots K$,
  - draw an aspect-word distribution $\phi_k^a \sim Dir(\boldsymbol{\beta}^a)$
  - For each sentiment label $r = 1, 2, \ldots R$, draw an aspect-sentiment label-word distribution $\phi_{k,r}^s \sim Dir(\boldsymbol{\beta}^s)$
- For each product $p \in P$,
  - draw a product-aspect distribution $\theta_p \sim Dir(\boldsymbol{\alpha})$
  - for each aspect k, draw a product-aspect-sentiment label distribution $\lambda_{p,k} \sim Dir(\boldsymbol{\pi_{p,k}})$ where $\boldsymbol{\pi_{p,k}} = f(\boldsymbol{Y_p}, \boldsymbol{\omega_k}, \omega^b)$
- For each phrase $f =< h, m >$ of product p,
  1. Draw an aspect z from θ_p
  2. Draw a sentiment label r from $\lambda_{p,z}$
  3. Draw an aspect word h from ϕ_z^a
  4. Draw a sentiment word m from $\phi_{z,r}^s$. If $m \in L$, generate a word feature v_m based on H.

By integrating out $\boldsymbol{\theta}$, $\boldsymbol{\phi}$ and $\boldsymbol{\lambda}$, the joint probability can be defined as:

$$P(\boldsymbol{z}, \boldsymbol{r}, \boldsymbol{h}, \boldsymbol{m}, \boldsymbol{v} | \boldsymbol{\alpha}, \boldsymbol{\beta^a}, \boldsymbol{\beta^s}, \boldsymbol{\pi}, H) = P(\boldsymbol{z}|\boldsymbol{\alpha}) P(\boldsymbol{r}|\boldsymbol{z}, \boldsymbol{\pi}) P(\boldsymbol{h}|\boldsymbol{z}, \boldsymbol{\beta^a}) P(\boldsymbol{m}|\boldsymbol{z}, \boldsymbol{r}, \boldsymbol{\beta^s}) P(\boldsymbol{v}|\boldsymbol{r}, H) \quad (2)$$

3.4 Inference

We use Gibbs Sampling(Griffiths and Steyvers, 2004) to estimate the posterior distribution given the observed data.

We jointly sample the aspect z and sentiment label r for the ith phrase $< h, m >$ of product p, given the assignments of other phrases:

$$P(z_i = k, r_i = l | \boldsymbol{z}_{-i}, \boldsymbol{r}_{-i}, \boldsymbol{h}, \boldsymbol{m}, \boldsymbol{v}) \propto (n_{p,k} + \alpha) \frac{n_{k,h}^a + \beta^a}{\sum_{h'} (n_{k,h'}^a + \beta^a)} \frac{n_{p,k,l} + \pi_{p,k,l}}{\sum_{l'} (n_{p,k,l'} + \pi_{p,k,l'})} \frac{n_{k,l,m}^s + \beta^s}{\sum_{m'} (n_{k,l,m'}^s + \beta^s)} g(m, l) \quad (3)$$

where $g(m, l) = H(v_m | l)$ if $m \in L$. In this case, when we sample the sentiment label r for this phrase, the probability of generating word feature v_m from r is also considered. For example, the word "excellent" has a word feature value $v_m = 5$. Based on H, the probability of generating a word feature 5 is higher for sentiment labels with larger values. If $m \notin L$, there is no $g(m, l)$ term, since no word feature is associated with this phrase. In Equation 3, $n_{p,k}$ is the number of times aspect k is assigned to phrases of product p, and $n_{k,h}^a$ is the number of times aspect word h is assigned to aspect k. $n_{p,k,l}$ is the number of times sentiment label l is assigned to aspect k for product p, and $n_{k,l,m}^s$ is the number of times sentiment word m is assigned to aspect k and sentiment label l. All these counts exclude assignments for the current phrase $< h, m >$.

Based on the samples, we can estimate $\lambda_{p,k,r}$ as:

$$\lambda_{p,k,r} = \frac{n_{p,k,r} + \pi_{p,k,r}}{\sum_{r'} (n_{p,k,r'} + \pi_{p,k,r'})} \quad (4)$$

Since sentiment labels and atings are aligned, the aspect rating t_{pk} of product p on aspect k can be simply calculated as the expectation of $\boldsymbol{\lambda_{p,k}}$:

$$t_{pk} = \sum_r \lambda_{p,k,r} \cdot r \quad (5)$$

4 Experiments

In this section, we describe the experiments and analyze the results.

4.1 Dataset

We use the TripAdvisor dataset[1](Wang et al., 2010) for evaluation, since in this dataset, reviews are not only associated with overall ratings, but also with ground truth aspect ratings on 7 aspects: *value, room, location, cleanliness, check in/front desk, service, business service*. All the ratings in the dataset are in the range from 1 star to 5 stars. We first remove reviews with any missing aspect ratings or very short reviews(less than three sentences). Then we adopt the dependency parser technique to identify opinion phrases, and collect phrases with adjective sentiment words. The dependency parser can deal with conjunctions, negations and bigram aspect words, and it results in the best performance according to (Moghaddam and Ester, 2012). Some sample phrases are shown in Table 1. All words are converted into lower case, and we remove phrases containing words that appear no more than 10 times or stop words. Since we are only interested in product-level aspect rating prediction, for each product, we aggregate all the review overall ratings to get the overall rating distribution. The statistics of the dataset is shown in Table 2. The average rating is the rating averaged over all reviews and all products. As we can see, positive reviews are dominant in the data, which raises the challenge of discovering negative sentiment topics.

Sentences	Phrases
The room, facing the courtyard, was large and comfortable.	<room, large>, <room, comfortable>
The room was not really clean.	<room, no_clean>
Internet access was available.	<Internet access, available>

Table 1: Sample extracted phrases

#Products	#Reviews	Avg rating	#Phrases
1850	61306	4.03	740982

Table 2: Statistics of the dataset

4.2 Metrics

We use three evaluation metrics for comparison.

RMSE: Root-mean-square error is used to measure the difference between the predicted aspect ratings and ground truth aspect ratings. It is defined as:

$$RMSE = \sqrt{\frac{\sum_p \sum_k (t_{pk} - \hat{t}_{pk})^2}{|P| \times K}} \quad (6)$$

where t_{pk} is the predicted aspect rating for product p on aspect k, and $\hat{t}_{pk}$ is the ground truth.

Precision@N: For each aspect k, we rank the hotels based on their predicted aspect ratings, and get the top N results. A hotel is considered *relevant* if its ground truth aspect rating is in the top 10% of the ground truth aspect ratings of all hotels. Precision@N is defined as the percentage of the top N results that are relevant:

$$Precision@N = \frac{|\{\text{relevant hotels}\} \cap \{\text{top N ranked hotels}\}|}{N} \quad (7)$$

We use $N = 10$, and the result is averaged over K aspects.

$\boldsymbol{\rho_{hotel}}$: Pearson correlation across hotels(Wang et al., 2010) is defined as:

$$\boldsymbol{\rho_{hotel}} = \frac{\sum_k \rho(\boldsymbol{t_k}, \hat{\boldsymbol{t}}_{\boldsymbol{k}})}{K} \quad (8)$$

where $\boldsymbol{t_k}$ is the predicted aspect rating vector for all hotels on aspect k, and $\hat{\boldsymbol{t}}_{\boldsymbol{k}}$ is the corresponding ground truth vector. $\rho(\boldsymbol{t_k}, \hat{\boldsymbol{t}}_{\boldsymbol{k}})$ is the Pearson correlation between these two vectors. It measures how the predicted ratings of aspect k can preserve the order in the ground truth(Wang et al., 2010). If we can predict an aspect-specific ranking similar to the ground truth, we can use the predicted aspect ratings to answer questions like "Is hotel a better than hotel b on aspect k?"

4.3 Baselines

The first three baselines are **Local Prediction**, **Global Prediction** and **Graph Propagation**. They all separate aspect extraction and sentiment identification. For each phrase $f =< h, m >$ from review d of product p, we first find the aspect assignment of this phrase. Then, we use three methods to get the phrase rating. Local Prediction(Lu et al., 2009) simply uses the overall rating of d as its phrase rating. Global Prediction(Lu et al., 2009) trains a multi-class classifier to classify the sentiment word m into a rating category $r \in 1, 2 \ldots R$,

[1] http://sifaka.cs.uiuc.edu/~wang296/Data/index.html

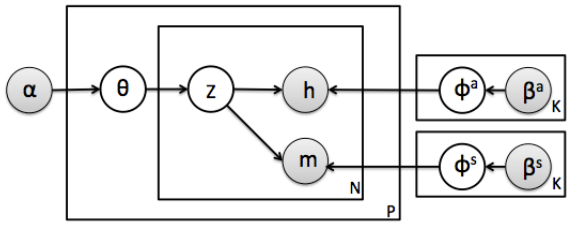


Figure 2: Method for aspect extraction in Local Prediction, Global Prediction and Graph Propagation

then assigns r as the phrase rating. Graph Propagation(Brody and Elhadad, 2010) builds a conjunction graph for sentiment words, and uses a Label Propagation algorithm on the graph to learn the sentiment polarity score for each sentiment word. The score of m is set as phrase rating. Finally, we aggregate all the phrases of each aspect to predict the aspect ratings. To apply these methods in our experiments, in the aspect extraction step, we adapt our model to extract only aspects, as shown in Figure 2. In this simplified model, no sentiment labels is involved, and the latent aspect explains both the aspect word and sentiment word.

ILDA(Moghaddam and Ester, 2011) was proposed for aspect rating prediction, but it fails to deal with the sentiment label alignment problem, so it cannot be directly used for this task. We adopt the common approach of providing seed words to set priors for each sentiment topic.

LRR(Wang et al., 2010) was proposed to predict aspect ratings for each review, but it can also be used to predict product aspect ratings by aggregating all the reviews of a product into a single "h-review"(Wang et al., 2011). First, we can run a topic model to learn aspects, and annotate each sentence with an aspect. Then LRR is applied on the annotated sentences to predict aspect ratings. This approach provided the best result, according to (Wang et al., 2011). In the first step we use the sentence-LDA(Jo and Oh, 2011) to annotate sentences, which is slightly different from the original method, but still provides a good analogy.

We also test two simplified version of the SATM model. First, we remove the part which involves sentiment lexicons, so we only use the product overall rating distribution. We call this method **SATM-O**. Second, we use only sentiment lexicons, ignoring the influence of overall rating distribution. We call it **SATM-L**. These two baselines can help us identify how the sentiment lexicon and overall rating distribution can improve the results, if used separately.

Our last baseline simply uses the overall rating of a hotel as its aspect ratings. For each hotel, its overall rating is defined as the average overall rating of its reviews. This method is referred to as **Overall**.

4.4 Experimental Setup

For all topic modelling based approaches, the number of aspects is set to 7. Since we can evaluate aspect rating prediction only on the predefined aspects, we need to ensure the discovered aspects match the predefined aspects. To do this, we adopt the common approach of providing a few seed words for each aspect as priors, as in (Wang et al., 2010). The seed words are listed in Table 3. There may be better methods to use seed words for aspect discovery (Jagarlamudi et al., 2012; Mukherjee and Liu, 2012), and it would be interesting to combine their methods with ours. However, this is beyond the scope of this paper, and we list it as future work.

Aspects	Seed words
Value	value, price, worth
Room	room, rooms
Location	location
Cleanliness	room, dirty, smelled, clean
Check in/front desk	staff
Service	service, breakfast, food
Business service	internet, wifi

Table 3: Seed words for aspect discovery

We use 5 sentiment labels in SATM, SATM-L and SATM-O, as this is the number of distinct ratings. The lexicon L used in our experiment is part of (Taboada et al., 2011) where words are associated with polarity scores in the range $[-5,-1] \cup [1,5]$. We observe that words with polarity score 1 and -1 express too weak sentiments, so we discard them in our experiment. To get training instances for sentiment association H, we treat each appearance of word $m \in L$ in the data as one training instance. The polarity score s_m is directly retrieved from L, and the sentiment label r_m is the overall rating of review d where m appears. This approach avoids the need for manual annotation of sentiment labels, and the annotation result captures the characteristics of the dataset. How-

ever, all training instances in a review will have the same sentiment label, which means that we assume all sentiment words in a review express the same sentiment, no matter what aspects they talk about. This is not true, thus will introduce noise to the training. To reduce noise, for words with positive polarity score, we ignore their appearance in reviews with rating 1 and 2, since we assume positive sentiment words rarely express negative sentiments, even if they appear in negative reviews. Therefore, $H(s_m|r_m) = 0$ for $r_m = 1, 2$ and s_m in the range $[2, 5]$. A similar method is used to deal with words with negative polarity score.

For Global Prediction, in (Lu et al., 2009), the prior for the multi-class classifier is uniform, while in our experiment, for product p, we used product overall rating distribution on r as the prior for rating category r, which achieves better results than uniform prior.

The Graph Propagation method requires a small set of sentiment words as seeds, from which the algorithm can learn sentiment score for other words. The method in (Brody and Elhadad, 2010) constructs these seed words based on morphology in an unsupervised way, and can only support two kinds of sentiment: positive and negative. In our experiment, since the sentiment lexicon is available, the sentiment seed words are from the lexicon, and we update the polarity score for those not in the lexicon.

For ILDA, since we need to provide seed words as priors for sentiment topics, we have two options, and we use both for experiment. First, we can employ the common approach of using two sentiment labels(R=2, positive and negative). Then, words with positive polarity scores in lexicon L are used as priors for the positive sentiment topic, and similarly words with negative polarity scores for negative sentiment topic. An alternative approach is to use 5 sentiment labels(R=5). It provides finer grained sentiment extraction, but raises the question of how to choose seed words for each sentiment topic. To do this, we use the full sentiment lexicon in (Taboada et al., 2011), where sentiment words have polarity score in the range of $[-5, -1] \cup [1, 5]$. We divide the lexicon, and use words with polarity score 4 and 5 as prior for the sentiment topic with label 5. Then, words with polarity score 2 and 3 are used for the sentiment topic with label 4, and so on.

For all topic modelling based approaches, we set the number of iterations for Gibbs Sampling to 3000, and take samples from the markov chain every 50 iterations after a burn-in period of 1000 iterations. In SATM and SATM-O, for all aspects k, we need to choose the parameters $\boldsymbol{\omega_k}$ and also w^b. We use a small portion of dataset with ground truth to choose the best value, and we set $\omega_k^1 = 20$, $w_k^0 = 0.01, w^b = 0$. Automatically learning these parameters are feasible. One possible option is to use stochastic EM sampling scheme, as in (Mimno and McCallum, 2008). For the LRR implementation[2], we use the default parameters included in the package, and train the model with seed words provided by the author(Wang et al., 2010).

4.5 Results

The experimental results are listed in Table 4. For RMSE, the smaller the better, while for the other two measures, the larger the better. Graph Propagation, ILDA and SATM-L do not use the overall ratings(except for training sentiment association H), so we group them together. Similarly we group Local Prediction, Global Prediction, SATM-O and SATM. The Overall method is a special baseline that does not do any aspect based prediction. For the LRR method, after the first step of sentence annotation, we notice that sentence-LDA fails to annotate the "h-review" of some hotel with all 7 aspects, mainly because these hotels are associated with less reviews. In this case, the LRR model will fail in the second step, so we do not include LRR in Table 4. Instead, we compared our method with LRR on a subset of products that comment on all aspects based on the sentence annotation. There are 1533 hotels in this subset, and the result is shown in Table 5. Note that our experimental results for LRR are far worse than those reported in the original paper(Wang et al., 2011). We believe this maybe due to different parameter settings, or due to the choice of different reviews.

We observe that SATM achieves the best RMSE value, i.e., it produces the most accurate aspect rating prediction. The Overall method does better in ranking all the hotels(ρ_{hotel}), but SATM is better at ranking top hotels($P@10$). When we compare the results of SATM with SATM-L and SATM-O, we find that the good performance of SATM is mainly due to the use of the overall rating distribution. On one hand, this is reasonable, since in-

[2] http://sifaka.cs.uiuc.edu/~wang296/Codes/LARA.zip

Sentiment label	Top sentiment words
1	old, dirty, worn, older, dark, stained, broken, dated, outdated, bad
2	small, tiny, little, noisy, single, double, uncomfortable, smaller, larger, narrow
3	large, double, big, mini, hard, main, huge, twin, single, jacuzzi
4	nice, comfortable, modern, clean, new, good, great, flat, big, comfy
5	large, huge, great, beautiful, big, lovely, separate, spacious, wonderful, excellent

Table 6: Top sentiment words for aspect "room" with different sentiment labels

Methods	RMSE	P@10	ρ_{hotel}
ILDA,R=2	1.202	0.30	0.193
ILDA,R=5	1.096	0.257	0.222
Graph Propagation	0.718	0.271	0.442
SATM-L	0.774	0.443	0.483
Local Prediction	0.572	0.486	0.761
Global Prediction	0.625	0.30	0.778
SATM-O	0.429	0.80	0.841
SATM	**0.384**	**0.814**	0.854
Overall	0.415	0.80	**0.863**

Table 4: Experimental results except LRR

Methods	RMSE	P@10	ρ_{hotel}
LRR	1.018	0.3	0.404
SATM	**0.373**	**0.829**	**0.849**

Table 5: Experimental comparison with LRR

tuitively aspect ratings usually do not diverge too far from the overall rating, especially for hotels with higher overall ratings. As we can see from the result of Overall, the overall rating has good correlation with aspect ratings, and using overall rating only is already a strong predictor for aspect ratings. Also, in most cases, methods using overall ratings(Overall and the four methods in the middle of Table 4) are better than others(first four methods). On the other hand, we should not rely only on the overall rating distribution. By incorporating the sentiment lexicon, for RMSE, SATM achieves 10% improvement over SATM-O and 7% improvement than Overall. Also, the overall rating may not always be a good aspect rating predictor, depending on the dataset.

To take a closer look at cases where the overall rating is not a good aspect rating predictor, we evaluate the RMSE on different subsets of hotels. We divide the hotels into different overall rating ranges: [1.2), [2,3), [3,4) and [4,5]. The results are shown in Table 7. Going from the [4,5] group to [1,2) group, the overall rating becomes less and less reliable to predict aspect ratings, and the gain of SATM increases compared to SATM-O and Overall. For a hotel with higher overall rating(good hotel), its aspect ratings are closer to the overall rating. This matches our intuition that good hotels are expected to be good on most aspects, if not on all aspects. For a hotel with average and lower overall rating, the average difference between aspect ratings and overall rating is larger. In this case, the overall rating can not tell us the whole story, which calls for aspect based prediction. Our method achieves the best RMSE gain on this group of hotels.

Methods	[1,2)	[2,3)	[3,4)	[4-5]
Local Prediction	0.789	0.772	0.621	0.456
Global Prediction	1.013	0.884	0.584	0.567
SATM-O	0.703	0.564	0.446	0.359
SATM	**0.606**	**0.494**	**0.394**	0.332
Overall	0.735	0.612	0.431	**0.320**

Table 7: RMSE on hotels with different overall rating ranges

4.6 Qualitative analysis

To provide a qualitative analysis, we can list the top words for the aspect-sentiment label-word distributions. In Table 6, we list them for the aspect "room", with 5 different sentiment labels. We observe that, as the sentiment label value increases, the sentiment topics express more and more positive sentiments. This means the sentiment labels and ratings are indeed aligned, so that we can use these sentiment labels to predict ratings.

5 Conclusion and future work

In this paper, we proposed a sentiment aligned topic model(SATM) for product aspect rating prediction. By incorporating the overall rating distribution and a sentiment lexicon, our SATM model can align sentiment labels with ratings. Experiments on a TripAdvisor dataset demonstrate the effectiveness of SATM on aspect rating prediction.

In SATM, for each product and each aspect, the multinomial distribution over sentiment labels has

prior parameterized by product overall rating distribution. We assume linear dependency, but it will be interesting to explore other dependencies. Another direction is to learn the parameters $\boldsymbol{\omega_k}$ automatically, so that $\boldsymbol{\omega_k}$ can be different for different k, capturing the influence of the overall rating on different aspects.

Finally, we assume each phrase is associated with one latent aspect. However, aspects may be correlated. For example, the phrase <room, filthy> gives us information about the aspect *room* and also the aspect $cleanliness$. To deal with this problem, we can relax the assumption that one phrase talks about one aspect, or we can model correlation among aspects.

Acknowledgments

This research is supported by NSERC Discovery Grant. The authors thank Dr. Maite Taboada for providing the sentiment lexicon.

References

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 25–32, New York, NY, USA. ACM.

David Andrzejewski, Xiaojin Zhu, Mark Craven, and Benjamin Recht. 2011. A framework for incorporating general domain knowledge into latent dirichlet allocation using first-order logic. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1171–1177. AAAI Press.

Sasha Blair-goldensohn, Tyler Neylon, Kerry Hannan, George A. Reis, Ryan Mcdonald, and Jeff Reynar. 2008. Building a sentiment summarizer for local service reviews. In *WWW Workshop on NLP in the Information Explosion Era*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 804–812, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malú Castellanos, and Riddhiman Ghosh. 2013. Exploiting domain knowledge in aspect extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1655–1667.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, April.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA. ACM.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Jagadeesh Jagarlamudi, Hal Daumé, III, and Raghavendra Udupa. 2012. Incorporating lexical priors into topic models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 204–213, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yohan Jo and Alice H. Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 815–824, New York, NY, USA. ACM.

Suin Kim, Jianwen Zhang, Zheng Chen, Alice H. Oh, and Shixia Liu. 2013. A hierarchical aspect-sentiment model for online reviews. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*.

Himabindu Lakkaraju, Chiranjib Bhattacharyya, Indrajit Bhattacharya, and Srujana Merugu. 2011. Exploiting coherence for the simultaneous discovery of latent facets and associated sentiments. In *Proceedings of the Eleventh SIAM International Conference on Data Mining, SDM 2011, April 28-30, 2011, Mesa, Arizona, USA*, pages 498–509.

Angeliki Lazaridou, Ivan Titov, and Caroline Sporleder. 2013. A bayesian model for joint unsupervised induction of sentiment, aspect and discourse representations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 1630–1639.

Fangtao Li, Minlie Huang, and Xiaoyan Zhu. 2010. Sentiment analysis with global topics and local dependency. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 375–384, New York, NY, USA. ACM.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Chong Long, Jie Zhang, Minlie Huang, Xiaoyan Zhu, Ming Li, and Bin Ma. 2014. Estimating feature ratings through an effective review selection approach. *Knowl. Inf. Syst.*, 38(2):419–446.

Yue Lu, ChengXiang Zhai, and Neel Sundaresan. 2009. Rated aspect summarization of short comments. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 131–140, New York, NY, USA. ACM.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: Modeling facets and opinions in weblogs. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 171–180, New York, NY, USA. ACM.

David M. Mimno and Andrew McCallum. 2008. Topic models conditioned on arbitrary features with dirichlet-multinomial regression. In *the Conference on Uncertainty in Artificial Intelligence*, pages 411–418.

Samaneh Moghaddam and Martin Ester. 2011. Ilda: Interdependent lda model for learning latent aspects and their ratings from online product reviews. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 665–674, New York, NY, USA. ACM.

Samaneh Moghaddam and Martin Ester. 2012. On the design of lda models for aspect-based opinion mining. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 803–812, New York, NY, USA. ACM.

Samaneh Moghaddam and Martin Ester. 2013. The flda model for aspect-based opinion mining: Addressing the cold start problem. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 909–918.

Arjun Mukherjee and Bing Liu. 2012. Aspect extraction through semi-supervised modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 339–348, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 339–346, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christina Sauper and Regina Barzilay. 2013. Automatic aggregation by joint modeling of aspects and values. *J. Artif. Int. Res.*, 46(1):89–127, January.

Christina Sauper, Aria Haghighi, and Regina Barzilay. 2011. Content models with attitude. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 350–358, Stroudsburg, PA, USA. Association for Computational Linguistics.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Comput. Linguist.*, 37(2):267–307, June.

Ivan Titov and Ryan McDonald. 2008a. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 111–120, New York, NY, USA. ACM.

Ivan Titov and Ryan T. McDonald. 2008b. A joint model of text and aspect ratings for sentiment summarization. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 308–316.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: A rating regression approach. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 783–792, New York, NY, USA. ACM.

Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 618–626, New York, NY, USA. ACM.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 56–65, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical report, Technical Report CMU-CALD-02-107, Carnegie Mellon University.

Learning Emotion Indicators from Tweets: Hashtags, Hashtag Patterns, and Phrases

Ashequl Qadir
School of Computing
University of Utah
Salt Lake City, UT 84112, USA
asheq@cs.utah.edu

Ellen Riloff
School of Computing
University of Utah
Salt Lake City, UT 84112, USA
riloff@cs.utah.edu

Abstract

We present a weakly supervised approach for learning hashtags, hashtag patterns, and phrases associated with five emotions: AFFECTION, ANGER/RAGE, FEAR/ANXIETY, JOY, and SADNESS/DISAPPOINTMENT. Starting with seed hashtags to label an initial set of tweets, we train emotion classifiers and use them to learn new emotion hashtags and hashtag patterns. This process then repeats in a bootstrapping framework. Emotion phrases are also extracted from the learned hashtags and used to create phrase-based emotion classifiers. We show that the learned set of emotion indicators yields a substantial improvement in F-scores, ranging from +%5 to +%18 over baseline classifiers.

1 Introduction

Identifying emotions in social media text can be beneficial for many applications, for example to help companies understand how people feel about their products, to assist governments in recognizing growing anger or fear associated with an event, or to help media outlets understand people's emotional response toward controversial issues or international affairs. On the Twitter micro-blogging platform, people often use hashtags to express an emotional state (e.g., *#happyasalways*, *#angryattheworld*). While some hashtags consist of a single word (e.g., *#angry*), many hashtags include multiple words and creative spellings (e.g., *#cantwait4tmrw, #Youredabest*), which can not be easily recognized using sentiment or emotion lexicons.

Our research learns three types of emotion indicators for tweets: hashtags, hashtag patterns, and phrases for one of five emotions: AFFECTION, ANGER/RAGE, FEAR/ANXIETY, JOY, or SADNESS/DISAPPOINTMENT. We present a bootstrapping framework for learning emotion hashtags and extend the framework to also learn more general hashtag patterns. We then harvest emotion phrases from the hashtags and hashtag patterns for contextual emotion classification.

First, we make the observation that emotion hashtags often share a common prefix. For example, *#angryattheworld* and *#angryatlife* both have the prefix *"angry at"*, which suggests the emotion ANGER. Consequently, we generalize beyond specific hashtags to create *hashtag patterns* that will match all hashtags with the same prefix, such as the pattern *#angryat** which will match both *#angryattheworld* and *#angryatlife*.

A key challenge is that a seemingly strong emotion word or phrase can have a different meaning depending upon the following words. For example, *#angry** may seem like an obvious pattern to identify ANGER tweets. But *#angrybirds* is a popular hashtag that refers to a game, not the writer's emotion. Similarly, "love you" usually expresses AFFECTION when it is followed by a person (e.g., *#loveyoumom*). But it may express JOY in other contexts (e.g., *#loveyoulife*). We use probability estimates to determine which hashtag patterns are reliable indicators for an emotion.

Our second observation is that hashtags can also be used to harvest emotion phrases. For example, if we learn that the hashtag *#lovelife* is associated with JOY, then we can extract the phrase "love life" from the hashtag and use it to recognize emotion in the body of tweets. However, unlike hashtags, which are self-contained, the words surrounding a phrase in a tweet must also be considered. For example, negation can toggle polarity (*"don't love life"* may suggest SADNESS, not JOY) and the aspectual context may indicate that no emotion is being expressed (e.g., *"I would love life if ..."*). Consequently, we train classifiers to determine if a tweet contains an emotion based on both an emotion phrase and its context.

2 Related Work

In addition to sentiment analysis, which has been widely studied (e.g., (Barbosa and Feng, 2010; Brody and Diakopoulos, 2011; Kouloumpis et al., 2011; Mitchell et al., 2013)), recognizing emotions in social media text has also become a popular research topic in recent years. Researchers have studied feature sets and linguistic styles (Roberts et al., 2012), emotion influencing behaviors (Kim et al., 2012), sentence contexts (Yang et al., 2007b), hierarchical emotion classification (Ghazi et al., 2010; Esmin et al., 2012) and emotion lexicon creation (Yang et al., 2007a; Mohammad, 2012a; Staiano and Guerini, 2014). Researchers have also started to utilize the hashtags of tweets, but primarily to collect labeled data (e.g., for sarcasm (Davi-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1203–1209,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

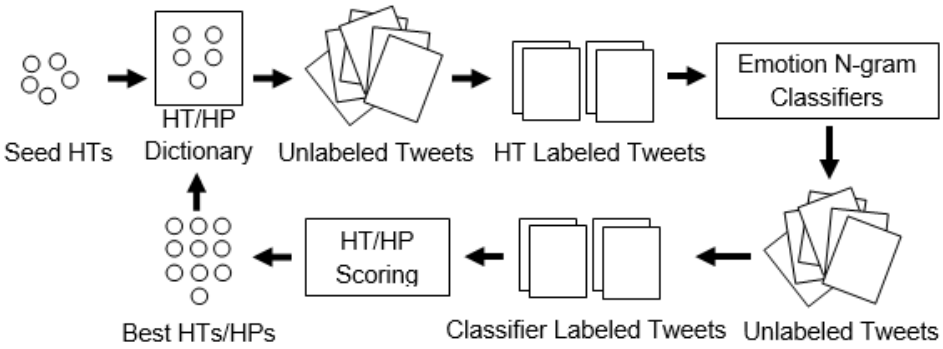


Figure 1: Bootstrapped Learning. (HT = hashtag; HP = hashtag pattern)

dov et al., 2010; Riloff et al., 2013) and for sentiment/emotion data (Wang et al., 2012; Mohammad et al., 2013; Choudhury et al., 2012; Purver and Battersby, 2012; Mohammad, 2012a)).

Wang et al. (2011) investigated several graph based algorithms to collectively classify hashtag sentiments, but their work is focused on positive versus negative polarity classification. Our research extends the preliminary work on bootstrapped learning of emotion hashtags (Qadir and Riloff, 2013) to additionally learn patterns corresponding to hashtag prefix expressions and to extract emotion phrases from the hashtags, which are used to train phrase-based emotion classifiers.

3 Learning Emotion Hashtags, Hashtag Patterns and Phrases

For our research, we collapsed Parrot's emotion taxonomy (Parrott, 2001)[1] into 5 emotion classes that frequently occur in tweets and minimally overlap with each other: AFFECTION, ANGER/RAGE, FEAR/ANXIETY, JOY, and SADNESS/DISAPPOINTMENT. We also used a NONE OF THE ABOVE class for tweets that do not express any emotion or express an emotion different from our five classes. For each of these categories, we identified 5 common hashtags that are strongly associated with the emotion and used them as seeds. Table 1 shows the seed hashtags.

Compared to the Ekman emotion classes (Ekman, 1992), one of the emotion taxonomies frequently used in NLP research (Strapparava and Mihalcea, 2007; Mohammad, 2012b), JOY, ANGER, SADNESS and FEAR are comparable to 4 of our 5 emotion classes. We do not study Ekman's SURPRISE and DISGUST classes, but include AFFECTION.

3.1 Learning Hashtags

Figure 1 presents the framework of the bootstrapping algorithm for hashtag learning. The process begins by collecting tweets that contain the seed hashtags and labeling them with the corresponding emotion. For this purpose, we collected 323,000 tweets in total that contain at least one of our seed hashtags. We also exploit a large pool of *unlabeled tweets* to use during bootstrapping, consisting of 2.3 million tweets with at least one hashtag per tweet (because we want to learn hashtags), collected using Twitter's streaming API. We did not include retweets or tweets with URLs, to reduce duplication and focus on tweets with original content. The *unlabeled tweets* dataset had 1.29 average hashtags-per-tweet and 3.95 average tweets-per-hashtag. We preprocessed the tweets with CMU's tokenizer (Owoputi et al., 2013) and normalized with respect to case.

Emotion Classes	Seed Hashtags
AFFECTION	*#loveyou*, *#sweetheart*, *#bff* *#romantic*, *#soulmate*
ANGER & RAGE	*#angry*, *#mad*, *#hateyou* *#pissedoff*, *#furious*
FEAR & ANXIETY	*#afraid*, *#petrified*, *#scared* *#anxious*, *#worried*
JOY	*#happy*, *#excited*, *#yay* *#blessed*, *#thrilled*
SADNESS & DISAPPOINTMENT	*#sad*, *#depressed* *#disappointed*, *#unhappy* *#foreveralone*

Table 1: Emotion Classes and Seed Hashtags

The labeled tweets are then used to train a set of emotion classifiers. We trained one logistic regression classifier for each emotion class using the LIBLINEAR package (Fan et al., 2008). We chose logistic regression because it produces probabilities with its predictions, which are used to assign scores to hashtags. As features, we used unigrams and bigrams with frequency $>$ 1. We removed the seed hashtags from the tweets so the classifiers could not use them as features.

For each emotion class $e \in E$, the tweets containing a seed hashtag for e were used as positive training instances. The negative training instances consisted of the tweets containing seed hashtags for the competing emotions as well as 100,000 randomly selected tweets

[1]There were other emotions in Parrott's taxonomy such as SURPRISE, NEGLECT, etc. that we did not use for this research.

Affection	Anger & Rage	Fear & Anxiety	Joy	Sadness & Disappointment
#yourthebest	#godie	#hatespiders	#tripleblessed	#leftout
#myotherhalf	#donttalktome	#haunted	#tgfad	#foreverugly
#bestfriendforever	#pieceofshit	#shittingmyself	#greatmood	#singleprobs
#loveyoulots	#irritated	#worstfear	#thankful	#lonerlyfe
#flyhigh	#fuming	#scaresme	#atlast	#teamlonely
#comehomesoon	#hateliars	#nightmares	#feelinggood	#unloved
#wuvyou	#heated	#paranoid	#happygirl	#friendless
#alwaysandforever	#getoutofmylife	#hateneedles	#godisgreat	#heartbroken
#missyousomuch	#angrytweet	#frightened	#superhappy	#needalife
#loveyougirl	#dontbothermewhen	#freakedout	#ecstatic	#letdown

Table 2: Examples of Learned Hashtags

from our unlabeled tweets. Although some of the unlabeled tweets may correspond to emotion e, we expect that most will have no emotion or an emotion different from e, giving us a slightly noisy but large, diverse set of negative instances.

We then apply each emotion classifier to the unlabeled tweets. For each emotion e, we collect the tweets classified as e and extract the hashtags from those tweets to create a candidate pool H_e of hashtags for emotion e. To limit the number of candidates, we discard hashtags that occur < 10 times, have just one character, or have > 20 characters. Next, we score each candidate hashtag h by computing the average probability assigned by the logistic regression classifier for emotion e over all of the tweets containing hashtag h. For each emotion class, we select the 10 hashtags with the highest scores. From the *unlabeled tweets*, we then add all tweets with one of the learned hashtags to the training instances, and the bootstrapping process continues. Table 2 shows examples of the learned hashtags.

3.2 Learning Hashtag Patterns

We learn hashtag patterns in a similar but separate bootstrapping process. We first expand each hashtag into a sequence of words using an N-gram based word segmentation algorithm[2] supplied with corpus statistics from our tweet collection. For example, *#angryatlife* expands[3] to the phrase *"angry at life"*. We use a Prefix Tree (Trie) data structure to represent all possible prefixes of the expanded hashtag phrases, but the prefixes consist of words instead of characters.

Next, we traverse the tries and consider all possible prefix paths as candidate hashtag patterns. We only consider prefixes that have occurred with at least one following word. For example, *#angryashell*, *#angryasalways*, *#angrybird*, *#angryatlife*, *#angryatyou* would produce patterns: *#angry**, *#angryas**, *#angryat** as shown in Figure 2.

We score each pattern by applying the classifier for emotion e (trained in the same way as hashtag learning) to all tweets having hashtags that match the pattern. We compute the average probability produced by the classifier, and for each emotion class, we select the 10 hashtag patterns with the highest scores. From the *unlabeled tweets*, we then add all tweets with hashtags that match one of the learned hashtag patterns to the training instances, and the bootstrapping process continues. Table 3 shows examples of learned hashtag patterns and matched hashtags.

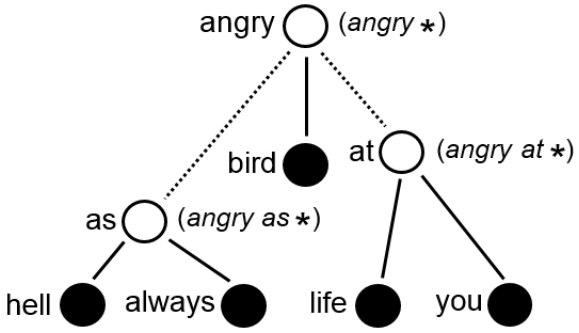


Figure 2: Trie of example hashtags with prefix *angry*. Dotted lines lead to non-terminal nodes where patterns are extracted.

3.3 Creating Phrase-based Classifiers

The third type of emotion indicator that we acquire are emotion phrases. At the end of the bootstrapping process, we apply the word segmentation algorithm to all of the learned hashtags and hashtag patterns to expand them into phrases (e.g., *#lovemylife* → *"love my life"*). Each phrase is assumed to express the same emotion as the original hashtag. However, as we will see in Section 4, just the presence of a phrase yields low precision, and surrounding context must also be taken into account.

Consequently, we train a logistic regression classifier for each emotion e, which classifies a tweet with respect to emotion e based on the presence of a learned phrase for e as well as a context window of size 6 around the phrase (set of 3 words on its left and set of 3

[2]http://norvig.com/ngrams/

[3]On a random sample of 100 hashtags, we found expansion accuracy to be 76% (+8% partially correct expansions).

Emotion	Hashtag Pattern	Examples of Matching Hashtags
AFFECTION	#bestie*	#bestiefolyfe, #bestienight, #bestielove
	#missedyou*	#missedyoutoomuch, #missedyouguys, #missedyoubabies
ANGER & RAGE	#godie*	#godieoldman, #godieyou, #godieinahole
	#pissedoff*	#pissedofffather, #pissedoffnow, #pissedoffmood
FEAR & ANXIETY	#tooscared*	#tooscaredtogoalone, #tooscaredformama, #tooscaredtomove
	#nightmares*	#nightmaresfordays, #nightmaresforlife, #nightmarestonight
JOY	#feelinggood*	#feelinggoodnow, #feelinggoodforme, #feelinggoodabout
	#goodmood*	#goodmooditsgameday, #goodmoodmode, #goodmoodnight
SADNESS & DISAPPOINTMENT	#bummed*	#bummedout, #bummedaf, #bummednow
	#singlelife*	#singlelifeblows, #singlelifeforme, #singlelifesucks

Table 3: Examples of Learned Hashtag Patterns and Matching Hashtags

words on its right). Tweets containing a learned phrase for e and a seed hashtag for e are the positive training instances. Tweets containing a learned phrase for e and a seed hashtag for a different emotion are used as the negative training instances. For example, when *"love my life"* is learned as an emotion phrase for JOY, the tweet, *"how can I love my life when everybody leaves me! #sad"* will have one feature each for the left words *"how"*, *"can"*, and *"I"*, one feature each for the right words *"when"*, *"everybody"* and *"leaves"*, and one feature for the phrase *"love my life"*. The tweet will then be considered a negative instance for JOY because *"#sad"* indicates a different emotion.

4 Experimental Results

To evaluate our learned emotion indicators, we manually selected 25 topic keywords/phrases[4] that we considered to be strongly associated with emotions, but not necessarily with any specific emotions of our study. We then searched in Twitter using Twitter Search API for any of these topic phrases and their corresponding hashtags. These 25 topic phrases are: *Prom, Exam, Graduation, Marriage, Divorce, Husband, Wife, Boyfriend, Girlfriend, Job, Hire, Laid Off, Retirement, Win, Lose, Accident, Failure, Success, Spider, Loud Noise, Chest Pain, Storm, Home Alone, No Sleep* and *Interview*. Since the purpose is to evaluate the quality and coverage of the emotion hashtags that we learn, we filtered out any tweet that did not have at least one hashtag.

Two annotators were given annotation guidelines and were instructed to label each tweet with up to two emotions. The instructions specified that the emotion must be felt by the writer. The annotators annotated 500 tweets with an inter-annotator agreement level of 0.79 Kappa (κ) (Carletta, 1996). The annotation disagreements in these 500 tweets were then adjudicated, and each annotator labeled an additional 2,500 tweets. Altogether this gave us an emotion annotated dataset of 5,500 tweets. We randomly separated out 1,000 tweets from this collection as a tuning set, and used the remaining 4,500 tweets as evaluation data. The distribution of emotions in the evaluation data was 6% for AFFECTION, 9% for ANGER/RAGE, 13% for FEAR/ANXIETY, 22% for JOY, and 12% for SADNESS/DISAPPOINTMENT. 42% of the tweets had none of the 5 emotions and 4% of the tweets had more than one emotions in the same tweet.

We created two baseline systems to assess the difficulty of the emotion classification task. First, we created SVM classifiers for each emotion using N-gram features and performed 10-fold cross-validation on the test data. We used LIBSVM (Chang and Lin, 2011) and set the $cost$ and $gamma$ parameters based on the tuning data. Second, we acquired the NRC Emotional Tweets Lexicon (Mohammad, 2012a), which contains emotion unigrams and bigrams for 8 emotions, 4 that are comparable to ours: ANGER, FEAR, JOY and SADNESS. We created a hashtag from each term in the lexicon by appending a # symbol on the front and removing whitespace. For each term, we chose the emotion with the highest score in the lexicon.

Table 4 shows our experimental results. The baseline classifiers (SVM_1 uses unigrams, SVM_{1+2} uses unigrams and bigrams) have low recall but 63-78% precision. The hashtags created from the NRC Lexicon have low precision. This could be due to possible entries (e.g., *"candy"* or *"idea"*), which without context are not much indicative of any specific emotion.

The second section of Table 4 shows the results when we label a tweet based on the presence of a hashtag or hashtag pattern. First, we use just the 5 seed hashtags to assess their coverage (as expected, high precision but low recall). Next, we add the hashtags learned during bootstrapping. For most emotions, the hashtags achieve performance similar to the supervised SVMs. The following row shows results for our learned hashtag patterns. Recall improves by +14% for AFFECTION, which illustrates the benefit of more general hashtag patterns, and at least maintains similar level of precision for other emotions. When the hashtags and hashtag patterns are combined (HTs+HPs), we see the best of both worlds with improved recall as high as +17% in AFFECTION and +10% in FEAR/ANXIETY

[4]This data collection process is similar to the emotion tweet dataset creation by Roberts et al. (2012)

Method	AFFECTION			ANGER & RAGE			FEAR & ANXIETY			JOY			SADNESS & DISAPPOINT.		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Baselines															
SVM_1	78	40	53	66	17	27	68	33	44	66	47	55	63	26	37
SVM_{1+2}	78	35	48	67	10	17	68	29	41	65	43	52	63	21	32
NRC Lexicon HTs	n/a			26	16	20	39	12	18	36	13	19	28	18	22
Learned Hashtags (HTs) and Hashtag Patterns (HPs)															
Seed HTs	94	06	11	75	01	03	100	06	11	93	04	08	81	02	05
All HTs	82	34	48	63	23	34	60	37	46	81	13	22	72	28	40
All HPs	76	48	59	60	22	32	57	42	48	84	09	16	73	16	26
All HTs+HPs	74	51	60	56	27	36	55	47	51	80	15	25	70	29	41
Learned Emotion Phrases															
Emotion Phrases	32	28	30	17	46	25	28	45	35	50	23	32	26	30	28
Phrase-based Classifier (PC)	54	07	12	48	05	09	63	17	27	69	12	20	50	06	11
SVM_1+PC	79	42	55	63	18	28	70	35	47	68	48	56	62	27	38
Hybrid Approach															
SVM_1+PC $\cup$ HTs+HPs	69	64	**66**	55	38	**45**	54	61	**57**	68	54	**60**	62	44	**51**

Table 4: Emotion Classification Results (P = Precision, R = Recall, F = F-score)

compared to All HTs, as well as improved F-scores across the board.

The third section of Table 4 presents the results for the emotion phrases. The first row (Emotion Phrases) shows that labeling a tweet based solely on the presence of a phrase is not very accurate. Next, we applied the trained models of the phrase-based classifiers (described in Section 3.3) to each tweet of the evaluation data. This provided us with probability of an emotion for each of the 5 emotions. The phrase-based classifiers (PC) yield higher precision, albeit with low recall. Finally, we use these probabilities as 5 additional features to SVM_1. The corresponding SVM_1+PC row shows a consistent 1-2 point F score gain over the original SVM_1 baseline.

The last section of Table 4 shows the best results with a hybrid system, which labels a tweet with emotion e if EITHER the enhanced SVM labels it as e OR the tweet contains a hashtag or hashtag pattern associated with e. This combined approach achieves substantially higher performance than any individual method across all 5 emotion classes, with improved F-scores ranging from +%5 to +%18 over the baseline classifiers, demonstrating that the different types of emotion indicators are complementary.

5 Conclusions

We have shown that three types of emotion indicators can be learned from tweets with weakly supervised bootstrapping: hashtags, hashtag patterns, and phrases. Our findings suggest that emotion hashtags are strong indicators for recognizing writer's emotion in tweets, and can be further generalized into hashtag patterns by learning prefix expressions corresponding to an emotion. Phrases learned from the hashtags and patterns are not always reliable by themselves, but training additional classifiers with the emotion phrases and their surrounding context provides added benefits to emotion classification in tweets. Our results showed that combining the learned emotion indicators with an N-gram classifier in a hybrid approach substantially improves performance across 5 emotion classes.

Acknowledgments

This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D12PC00285. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10.

Samuel Brody and Nicholas Diakopoulos. 2011. Cooooooooooooooollllllllllllll!!!!!!!!!!!!!!: using word lengthening to detect sentiment in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11.

Jean Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. *Comput. Linguist.*, 22:249–254, June.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May.

Munmun De Choudhury, Michael Gamon, and Scott Counts. 2012. Happy, nervous or surprised? classification of human affective states in social media. In *Proceedings of the Sixth International Conference on Weblogs and Social Media*.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL '10.

Paul Ekman. 1992. An argument for basic emotions. *Cognition and Emotion*, 6(3):169200.

Ahmed Ali Abdalla Esmin, Roberto L. De Oliveira Jr., and Stan Matwin. 2012. Hierarchical classification approach to emotion recognition in twitter. In *Proceedings of the 11th International Conference on Machine Learning and Applications, ICMLA, Boca Raton, FL, USA, December 12-15, 2012. Volume 2*, pages 381–385. IEEE.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June.

Diman Ghazi, Diana Inkpen, and Stan Szpakowicz. 2010. Hierarchical versus flat classification of emotions in text. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10.

Suin Kim, JinYeong Bak, and Alice Oh. 2012. Discovering emotion influence patterns in online social network conversations. *SIGWEB Newsl.*, (Autumn):3:1–3:6, September.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! In *Proceedings of the Fifth International Conference on Weblogs and Social Media*.

Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)*.

Saif Mohammad. 2012a. #emotional tweets. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics*.

Saif Mohammad. 2012b. Portable features for classifying emotional text. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Olutobi Owoputi, Brendan OConnor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL-2013)*.

W. Gerrod Parrott, editor. 2001. *Emotions in Social Psychology*. Psychology Press.

Matthew Purver and Stuart Battersby. 2012. Experimenting with distant supervision for emotion classification. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 482–491.

Ashequl Qadir and Ellen Riloff. 2013. Bootstrapped learning of emotion hashtags #hashtags4you. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, EMNLP '13.

Kirk Roberts, Michael A. Roach, Joseph Johnson, Josh Guthrie, and Sanda M. Harabagiu. 2012. Empatweet: Annotating and detecting emotions on twitter. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. ACL Anthology Identifier: L12-1059.

Jacopo Staiano and Marco Guerini. 2014. Depechemood: a lexicon for emotion analysis from crowd-annotated news. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Carlo Strapparava and Rada Mihalcea. 2007. SemEval-2007 Task 14: Affective Text. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*.

Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. 2011. Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11.

Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P. Sheth. 2012. Harnessing twitter "big data" for automatic emotion identification. In *Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE*

International Conference on Privacy, Security, Risk and Trust, SOCIALCOM-PASSAT '12.

Changhua Yang, Kevin Hsin-Yih Lin, and Hsin-Hsi Chen. 2007a. Building emotion lexicon from weblog corpora. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07.

Changhua Yang, Kevin Hsin-Yih Lin, and Hsin-Hsi Chen. 2007b. Emotion classification using web blog corpora. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, WI '07, pages 275–278.

Fine-Grained Contextual Predictions for Hard Sentiment Words

Sebastian Ebert and **Hinrich Schütze**
Center for Information and Language Processing
University of Munich, Germany
ebert@cis.lmu.de, inquiries@cislmu.org

Abstract

We put forward the hypothesis that high-accuracy sentiment analysis is only possible if word senses with different polarity are accurately recognized. We provide evidence for this hypothesis in a case study for the adjective "hard" and propose contextually enhanced sentiment lexicons that contain the information necessary for sentiment-relevant sense disambiguation. An experimental evaluation demonstrates that senses with different polarity can be distinguished well using a combination of standard and novel features.

1 Introduction

This paper deals with fine-grained sentiment analysis. We aim to make three contributions. First, based on a detailed linguistic analysis of contexts of the word "hard" (Section 3), we give evidence that highly accurate sentiment analysis is only possible if senses with different polarity are accurately recognized.

Second, based on this analysis, we propose to return to a lexicon-based approach to sentiment analysis that supports identifying sense distinctions relevant to sentiment. Currently available sentiment lexicons give the polarity for each word or each sense, but this is of limited utility if senses cannot be automatically identified in context. We extend the lexicon-based approach by introducing the concept of a *contextually enhanced sentiment lexicon* (CESL). The lexicon entry of a word w in CESL has three components: (i) the senses of w; (ii) a sentiment annotation of each sense; (iii) a data structure that, given a context in which w occurs, allows to identify the sense of w used in that context.

As we will see in Section 3, the CESL sense inventory – (i) above – should be optimized for sentiment analysis: closely related senses with the same sentiment should be merged whereas subtle semantic distinctions that give rise to different polarities should be distinguished.

The data structure in (iii) is a statistical classification model in the simplest case. We will give one other example for (iii) below: it can also be a set of centroids of context vector representations, with a mapping of these centroids to the senses.

If sentiment-relevant sense disambiguation is the first step in sentiment analysis, then powerful contextual features are necessary to support making fine-grained distinctions. Our third contribution is that we experiment with deep learning as a source of such features. We look at two types of deep learning features: word embeddings and neural network language model predictions (Section 4). We show that deep learning features significantly improve the accuracy of context-dependent polarity classification (Section 5).

2 Related work

Initial work on sentiment analysis was either based on sentiment lexicons that listed words as positive or negative sentiment indicators (e.g., Turney (2002), Yu and Hatzivassiloglou (2003)), on statistical classification approaches that represent documents as ngrams (e.g., Pang et al. (2002)) or on a combination of both (e.g., Riloff et al. (2003), Whitelaw et al. (2005)). The underlying assumption of lexicon-based sentiment analysis is that a word always has the same sentiment. This is clearly wrong because words can have senses with different polarity, e.g., "hard wood" (neutral) vs. "hard memory" (negative).

Ngram approaches are also limited because ngram representations are not a good basis for relevant generalizations. For example, the neutral adverbial sense 'intense' of "hard" ("laugh hard", "try hard") vs. the negative adjectival mean-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1210–1215,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

		Cobuild	syntax	meaning	example	patterns	sent.	# train	# test
1	FIRM	1	ADJ	firm, stiff	*hard floor*		neu	78	5
2	DIFFICULT	2, 4, 9, 10, 11	ADJ	difficult	*hard question*	*hard for*, *hard on*, *hard to V*	neg	2561	120
3	ADVERB	3a, 5, 6, 7	ADV	intensely	*work hard*		neu	425	19
4	INTENSE	3b	ADJ	intense	*hard look*	*be hard at it*	neu	24	7
5	HARD-MAN	8	ADJ	unkind	*hard man*		neg	15	0
6	HARD-TRUTH	12	attributive ADJ	definitely true	*hard truth*		neu	5	6
7	MUSIC		ADJ	hard-rock-type music	*hard beats*		neu	347	15
8	CONTRAST		ADJ	opposite of soft transition	*hard edge*		neu	3	1
9	NEGATIVE-P	13, 15	phrases				neg	36	2
10	NEUTRAL-P	14, 16	phrases				neu	375	27

Table 1: Sense inventory of "hard".

ing 'difficult' ("hard life", "hard memory") cannot be easily distinguished based on an ngram representation. Moreover, although ngram approaches could learn the polarity of these phrases they do not generalize to new phrases.

More recent compositional approaches to sentiment analysis can outperform lexicon and ngram-based methods (e.g., Socher et al. (2011), Socher et al. (2013)). However, these approaches conflate two different types of contextual effects: differences in sense or lexical meaning ("hard memory" vs. "hard wood") on the one hand and meaning composition like negation on the other hand. From the point of view of linguistic theory, these are different types of contextual effects that should not be conflated. Recognizing that "hard" occurs in the scope of negation is of no use if the basic polarity of the contextually evoked sense of "hard" (e.g., negative in "no hard memories" vs. neutral in "no hard wood") is not recognized.

Wilson et al. (2009) present an approach to classify contextual polarity building on a two-step process. First, they classify if a sentiment word is polar in a phrase and if so, second, they classify its polarity. Our approach can be seen as an extension of this approach; the main difference is that we will show in our analysis of "hard" that the polarity of phrases depends on the senses of the words that are used. This is evidence that high-accuracy polarity classification depends on sense disambiguation.

There has been previous work on assigning polarity values to senses of words taken from WordNet (e.g., Baccianella et al. (2010), Wiebe and Mihalcea (2006)). However, these approaches are not able to disambiguate the sense of a word given its context.

Previous work on representation learning for sentiment analysis includes (Maas and Ng, 2010) and (Maas et al., 2011). Their models learn word embeddings that capture semantic similarities and word sentiment at the same time. Their approach focuses on sentiment of entire sentences or documents and does not consider each sentiment word instance at a local level.

We present experiments with one supervised and one semisupervised approach to word sense disambiguation (WSD) in this paper. Other WSD approaches, e.g., thesaurus-based WSD (Yarowsky, 1992), could also be used for CESL.

3 Linguistic analysis of sentiment contexts of "hard"

We took a random sample of 5000 contexts of "hard" in the Amazon Product Review Data (Jindal and Liu, 2008). We use 200 as a test set and set aside 200 for future use. We analyzed the remaining 4600 contexts using a tool we designed for this study, which provides functionality for selecting and sorting contexts, including a keyword in context display. If a reliable pattern has been identified (e.g., the phrase "die hard"), then all contexts matching the pattern can be labeled automatically.

Our goal is to identify the different uses of "hard" that are relevant for sentiment. The basis for our inventory is the Cobuild (Sinclair, 1987)

lexicon entry for "hard". We use Cobuild because it was compiled based on an empirical analysis of corpus data and is therefore more likely to satisfy the requirements of NLP applications than a traditional dictionary.

Cobuild lists 16 senses. One of these senses (3) is split into two to distinguish the adverbial ("to accelerate hard") and adjectival ("hard acceleration") uses of "hard" in the meaning 'intense'. We conflated five senses (2, 4, 9, 10, 11) referring to different types of difficulty: "hard question" (2), "hard work" (4), "hard life" (11) and two variants of "hard on": "hard on someone" (9), "hard on something" (10); and four different senses (3a, 5, 6, 7) referring to different types of intensity: "to work hard" (3a), "to look hard" (5), "to kick hard" (6), "to laugh hard" (7). Furthermore, we identified a number of noncompositional meanings or phrases (lists NEGATIVE-P and NEUTRAL-P in the supplementary material[1]) in addition to the four listed by Cobuild (13, 14, 15, 16). In addition, new senses for "hard" are introduced for opposites of senses of "soft": the opposite of 'quiet/gentle voice/sound' (7: MUSIC; e.g., "hard beat", "not too hard of a song") and the opposite of 'smooth surface/texture' (8: CONTRAST; e.g., "hard line", "hard edge").

Table 1 lists the 10 different uses that are the result of our analysis. For each use, we give the corresponding Cobuild sense numbers, syntactic information, meaning, an example, typical patterns, polarity, and number of occurrences in training and test sets.

7 uses are neutral and 3 are negative. As "hard's" polarity in most sentiment lexicons is negative, but only 3 out of 7 senses are negative, "hard" provides evidence for our hypothesis that senses need to be disambiguated to allow for fine-grained and accurate polarity recognition.

We hired two PhD students to label each of the 200 contexts in the test set with one of the 10 labels in Table 1 ($\kappa = .78$). Disagreement was resolved by a third person.

We have published the labeled data set of 4600+200 contexts as supplementary material.

4 Deep learning features

We use two types of deep learning features to be able to make the fine-grained distinctions necessary for sense disambiguation. First, we use word embeddings similar to other recent work (see below). Second, we use a deep learning language model (LM) to predict the distribution of words for the position at which the word of interest occurs. For example, an LM will predict that words like "granite" and "concrete" are likely in the context "a * countertop" and that words like "serious" and "difficult" are likely in the context "a * problem". This is then the basis for distinguishing contexts in which "hard" is neutral (in the meaning 'firm, solid') from contexts in which it is a sentiment indicator (in the meaning 'difficult'). We will use the term *predicted context distribution* or PCD to refer to the distribution predicted by the LM.

We use the vectorized log-bilinear language model (vLBL) (Mnih and Kavukcuoglu, 2013) because it has three appealing features. (i) It learns state of the art word embeddings (Mnih and Kavukcuoglu, 2013). (ii) The model is a language model and can be used to calculate PCDs. (iii) As a linear model, vLBL can be trained much faster than other models (e.g., Bengio et al. (2003)).

The vLBL trains one set of word embeddings for the input space (R) and one for the target space (Q). We denote the input representation of word w as $\boldsymbol{r}_w$ and the target representation as $\boldsymbol{q}_w$. For a given context $c = w_1, \ldots, w_n$ the model predicts a target representation $\hat{\boldsymbol{q}}$ by linearly combining the context word representations with position dependent weights:

$$\hat{\boldsymbol{q}}(c) = \sum_{i=1}^{n} \boldsymbol{d}_i \odot \boldsymbol{r}_{w_i}$$

where $\boldsymbol{d}_i \in D$ is the weight vector associated with position i in the context and $\odot$ is pointwise multiplication. Given the model parameters $\theta = \{R, Q, D, \boldsymbol{b}\}$ the similarity between $\hat{\boldsymbol{q}}$ and the correct target word embedding is computed by the similarity function

$$s_\theta(w, c) = \hat{\boldsymbol{q}}(c)^T \boldsymbol{q}_w + b_w$$

where b_w is a bias term.

We train the model with stochastic gradient descent on mini-batches of size 100, following the noise-contrastive estimation training procedure of Mnih and Kavukcuoglu (2013). We use AdaGrad (Duchi et al., 2011) with the initial learning rate set to $\eta = 0.5$. The embeddings size is set to 100.

[1]All supplementary material is available at `http://www.cis.lmu.de/ebert`.

			ngram	PCD	embed	acc	prec	rec	F_1
	bl	1				.62	.62	1.00	.76
development	fully	2	+			.90	.91	.94	.92
		3		+		.90	.91	.92	.92
		4			+	.87	.87	.92	.90
		5	+	+		.92	.92	.94	.93
		6	+		+	.91	.90	.95	.92
		7		+	+	.86	.83	.96	.89
		8	+	+	+	.92	.93	.95	**.94**
	semi	9	+			.85	.87	.89	.88
		10		+		.85	.87	.89	.88
		11			+	.76	.73	.98	.83
		12	+	+		.85	.87	.89	.88
		13	+		+	.85	.87	.89	.88
		14		+	+	.85	.89	.87	.88
		15	+	+	+	.86	.87	.90	**.89**
test	bl	16				.66	.66	1.00	.80
	fully	17	+	+	+	.90	.89	.96	.92
	semi	18	+	+	+	.85	.85	.91	.88

Table 2: Classification results; bl: baseline

During training we do not need to normalize the similarity explicitly, because the normalization is implicitly learned by the model. However, normalization is still necessary for prediction. The normalized PCD for a context c of word w is computed using the softmax function:

$$P_\theta^c(w) = \frac{\exp(s_\theta(w, c))}{\sum_{w'} \exp(s_\theta(w', c))}$$

We use a window size of $ws = 7$ for training the model. We found that the model did not capture enough contextual phenomena for $ws = 3$ and that results for $ws = 11$ did not have better quality than $ws = 7$, but had a negative impact on the training time. Using a vocabulary of the 100,000 most frequent words, we train the vLBL model for 4 epochs on 1.3 billion 7-grams randomly selected from the English Wikipedia.

5 Experiments

The lexicon entry of "hard" in CESL consists of (i) the senses, (ii) the polarity annotations (neutral or negative) and (iii) the sense disambiguation data structure. Components (i) and (ii) are shown in Table 1. In this section, we evaluate two different options for (iii) on the task of sentiment classification.

	1	2	3	4	5	6	7	8
1								
2	‡							
3	‡							
4	‡	‡	·					
5	‡			‡				
6	‡			‡				
7	‡	‡	*		‡	‡		
8	‡	*	*	‡		*	‡	

Table 3: Significant differences of lines 1–8 in Table 2; ‡: p=0.01, *: p=0.05, ·: p=0.1

The first approach is to use a statistical classification model as the sense disambiguation structure. We use liblinear (Fan et al., 2008) with standard parameters for classification based on three different feature types: ngrams, embeddings (embed) and PCDs. Ngram features are all n-grams for $n \in \{1, 2, 3\}$. As embedding features we use (i) the mean of the input space (R) embeddings and (ii) the mean of the target space (Q) embeddings of the words in the context (see Blacoe and Lapata (2012) for justification of using simple mean). As PCD features we use the PCD predicted by vLBL for the sentiment word of interest, in our case "hard".

We split the set of 4600 contexts introduced in Section 3 into a training set of 4000 and a development set of 600.

Table 2 (lines 1–8) shows the classification results on the development set for all feature type combinations. Significant differences between results – computed using the approximate randomization test (Padó, 2006) – are given in Table 3. The majority baseline (bl), which assigns a negative label to all examples, reaches $F_1 = .76$. The classifier is significantly better than the baseline for all feature combinations with F_1 ranging from .89 to .94. We obtain the best classification result (.94) when all three feature types are combined (significantly better than all other feature combinations except for 5).

Manually labeling all occurrences of a word is expensive. As an alternative we investigate *clustering of the contexts of the word of interest*. Therefore, we represent each of the 4000 contexts of "hard" in the training set as its PCD[2], use

[2]To transform vectors into a format that is more appropriate for the underlying Gaussian model of kmeans, we take the square root of each probability in the PCD vectors.

kmeans clustering with $k = 100$ and then label each cluster. This decreases the cost of labeling by an order of magnitude since only 100 clusters have to be labeled instead of 4000 training set contexts.

Table 2 (lines 9–15) shows results for this semisupervised approach to classification, using the same classifier and the same feature types, but the cluster-based labels instead of manual labels.

For most feature combinations, F_1 drops compared to fully supervised classification. The best performing model for supervised classification (ngram+PCD+embed) loses 5%.

This is not a large drop considering the savings in manual labeling effort. All results are significantly better than the baseline. There are no significant differences between the different feature sets (lines 9–15) with the exception of embed, which is significantly worse than the other 6 sets.

The centroids of the 100 clusters can serve as an alternative sense disambiguation structure for the lexicon entry of "hard" in CESL.[3] Each sense s is associated with the centroids of the clusters whose majority sense is s.

As final experiment (lines 16–18 in Table 2), we evaluate performance for the baseline and for PCD+ngram+embed – the best feature set – on the test set. On the test set, baseline performance is .80 (.04 higher than .76 on line 1, Table 2); F_1 of PCD+ngram+embed is .92 (.02 less than development set) for supervised classification and is .88 (.01 less) for semisupervised classification (comparing to lines 8 and 15 in Table 2). Both results (.92 and .88) are significantly higher than the baseline (.80).

6 Conclusion

The sentiment of a sentence or document is the output of a causal chain that involves complex linguistic processes like contextual modification and negation. Our hypothesis in this paper was that for high-accuracy sentiment analysis, we need to model the root causes of this causal chain: the meanings of individual words. This is in contrast to other work in sentiment analysis that conflates different linguistic phenomena (word sense ambiguity, contextual effects, negation) and attempts to address all of them with a single model.

For sense disambiguation, the first step in the causal chain of generating sentiment, we proposed CESL, a contextually enhanced sentiment lexicon that for each word w holds the inventory of senses of w, polarity annotations of these senses and a data structure for assigning contexts of w to the senses. We introduced new features for sentiment analysis to be able to perform the fine-grained modeling of context needed for CESL. In a case study for the word "hard", we showed that high accuracy in sentiment disambiguation can be achieved using our approach. In future work, we would like to show that our findings generalize from the case of "hard" to the entire sentiment lexicon.

[3] Included in supplementary material.

Acknowledgments

This work was supported by DFG (grant SCHU 2246/10). We thank Lucia Krisnawati and Sascha Rothe for their help with annotation.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *International Conference on Language Resources and Evaluation*, pages 2200–2204.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *International Conference on Web Search and Web Data Mining*, pages 219–230.

Andrew L. Maas and Andrew Y. Ng. 2010. A probabilistic model for semantic word vectors. In *Annual Conference on Advances in Neural Information Processing Systems: Deep Learning and Unsupervised Feature Learning Workshop*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Annual Meeting of the Association for Computational Linguistics*, pages 142–150.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Annual Conference on Advances in Neural Information Processing Systems*, pages 2265–2273.

Sebastian Padó, 2006. *User's guide to `sigf`: Significance testing by approximate randomisation*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Conference on Empirical Methods in Natural Language Processing*, pages 79–86.

Ellen Riloff, Janyce Wiebe, and Theresa Ann Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Conference on Natural Language Learning*, volume 4, pages 25–32.

John Sinclair. 1987. *Looking Up: Account of the Cobuild Project in Lexical Computing*. Collins CoBUILD.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

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

Peter D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Annual Meeting of the Association for Computational Linguistics*, pages 417–424.

Casey Whitelaw, Navendu Garg, and Shlomo Argamon. 2005. Using appraisal groups for sentiment analysis. In *International Conference on Information and Knowledge Management*, pages 625–631. ACM.

Janyce Wiebe and Rada Mihalcea. 2006. Word sense and subjectivity. In *Annual Meeting of the Association for Computational Linguistics*, pages 1065–1072.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2009. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational Linguistics*, 35(3):399–433.

David Yarowsky. 1992. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *International Conference on Computational Linguistics*, pages 454–460.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Conference on Empirical Methods in Natural Language Processing*, pages 129–136.

An Iterative Link-based Method for Parallel Web Page Mining

Le Liu[1], Yu Hong[1], Jun Lu[2], Jun Lang[2], Heng Ji[3], Jianmin Yao[1]
[1]School of Computer Science & Technology, Soochow University, Suzhou, 215006, China
[2]Institute for Infocomm Research, Singapore, 138632
[3]Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY 12180, USA
giden@sina.cn,{tianxianer,lujun59,billlangjun}@gmail.com
jih@rpi.edu,jyao@suda.edu.cn

Abstracts

Identifying parallel web pages from bilingual web sites is a crucial step of bilingual resource construction for cross-lingual information processing. In this paper, we propose a link-based approach to distinguish parallel web pages from bilingual web sites. Compared with the existing methods, which only employ the internal translation similarity (such as content-based similarity and page structural similarity), we hypothesize that the external translation similarity is an effective feature to identify parallel web pages. Within a bilingual web site, web pages are interconnected by hyperlinks. The basic idea of our method is that the translation similarity of two pages can be inferred from their neighbor pages, which can be adopted as an important source of external similarity. Thus, the translation similarity of page pairs will influence each other. An iterative algorithm is developed to estimate the external translation similarity and the final translation similarity. Both internal and external similarity measures are combined in the iterative algorithm. Experiments on six bilingual websites demonstrate that our method is effective and obtains significant improvement (6.2% F-Score) over the baseline which only utilizes internal translation similarity.

1 Introduction

Parallel corpora have played an important role in multilingual Natural Language Processing, especially in Machine Translation (MT) and Cross-lingual Information Retrieval(CLIR). However, it's time-consuming to build parallel corpora manually. Some existing parallel corpora are subject to subscription or license fee and thus not freely available, while others are domain-specific. Therefore, a lot of previous research has focused on automatically mining parallel corpora from the web.

In the past decade, there have been extensive studies on parallel resource extraction from the web (e.g., Chen and Nie, 2000; Resnik 2003; Jiang et al., 2009) and many effective Web mining systems have been developed such as STRAND, PTMiner, BITS and WPDE. For most of these mining systems, there is a typical parallel resource mining strategy which involves three steps: (1) locate the bilingual websites (2) identify parallel web pages from these bilingual websites and (3) extract bilingual resources from the parallel web pages.

In this paper, we focus on the step (2) which is regarded as the core of the mining system (Chunyu, 2007). Estimating the translation similarity of two pages is the most basic and key problem in this step. Previous approaches have tried to tackle this problem by using the information within the pages. For example, in the STRAND and PTMiner system, a structural filtering process that relies on the analysis of the underlying HTML structure of pages is used to determine a set of pair-specific structural values, and then the values are used to decide whether the pages are translations of one another. The BITS system filters out bad pairs by using a large bilingual dictionary to compute a content-based similarity score and comparing the score with a threshold. The WPDE system combines URL similarity, structure similarity with content-based similarity to discover and verify candidate parallel page pairs. Some other features or rules such as page size ratio, predefined hypertexts which link to different language versions of a web page are also used in most of these systems. Here, all of the mining systems are simply using the information within the page in the process of find-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1216–1224,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

ing parallel web pages. In this paper, we attempt to explore other information to identify parallel web pages.

On the Internet, most web pages are linked by hyperlinks. We argue that the translation similarity of two pages depends on not only their internal information but also their neighbors. The neighbors of a web page are a set of pages, which link to the page. We find that the similarity of neighbors can provide more reliable evidence in estimating the translation similarity of two pages.

The main issues are discussed in this paper as follows:

- *Can the neighbors of candidate page pairs really contribute to estimating the translation similarity?*
- *How to estimate the translation similarity of candidate page pairs by using their neighbors?*

Our method has the following advantages:

High performance

The external and internal information is combined to verify parallel page pairs in our method, while in previous mining systems, only internal information was used. Experimental results show that compared with existing parallel page pair identification technologies, our method obtains both higher precision and recall (6.2% and 6.3% improvement than the baseline, respectively). In addition, the external information used in our method is a more effective feature than internal features alone such as structural similarity and content-based similarity.

Language independent

In principle, our method is language independent and can be easily ported to new language pairs, except for the language-specific bilingual lexicons. Our method takes full advantage of the link information that is language-independent. For the bilingual lexicons in our experiments, compared to previous methods, our method does not need a big bilingual lexicon, which is good news to less-resource language pairs.

Unsupervised and fewer parameters

In previous work, some parameters need to be optimized. Due to the diversity of web page styles, it is not trivial to obtain the best parameters. Some previous researches(Resnik, 2003; Zhang et al., 2006) attempt to optimize parameters by employing machine learning method. In contrast, in our method, only two parameters need to be estimated. One parameter remains stable for different style websites. Another parameter can be easily adjusted to achieve the best performance. Therefore, our method can be used in other websites with different styles, without much effort to optimize these parameters.

2 Related Work

A large amount of literature has been published on parallel resource mining from the web. According to the existing form of the parallel resource on the Internet, related work can be categorized as follows:

Mining from bilingual websites

Most existing web mining systems aimed at mining bilingual resource from the bilingual websites, such as PTMiner (Nie et al., 1999), STRAND (Resnik and Smith, 2003), BITS (Ma and Liberman, 1999), PTI (Chen et al., 2004). PTMiner uses search engines to pinpoint the candidate sites that are likely to contain parallel pages, and then uses the collected URLs as seeds to further crawl each web site for more URLs. Web page pairs are extracted based on manually defined URL pattern matching, and further filtered according to several criteria. STRAND uses a search engine to search for multilingual websites and generated candidate page pairs based on manually created substitution rules. Then, it filters some candidate pairs by analyzing the HTML pages. PTI crawls the web to fetch (potentially parallel) candidate multilingual web documents by using a web spider. To determine the parallelism between potential document pairs, a filename comparison module is used to check filename resemblance, and a content analysis module is used to measure the semantic similarity. BITS was the first to obtain bilingual websites by employing a language identification module, and then for each bilingual website, it extracts parallel pages based on their content.

Mining from bilingual web pages

Parallel/bilingual resources may exist not only in two parallel monolingual web pages, but also in single bilingual web pages. Jiang et al. (2009) used an adaptive pattern-based method to mine interesting bilingual data based on the observation that bilingual data usually appears collectively following similar patterns. They found that bilingual web pages are a promising source of up-to-date bilingual terms/sentences which cover many domains and application scenarios. In addition, Feng et al. (2010) proposed a new method

to automatically acquire bilingual web pages from the result pages of a search engine.

Mining from comparable corpus

Several attempts have been made to extract parallel resources from comparable corpora. Zhao et al. (2002) proposed a robust, adaptive approach for mining parallel sentences from a bilingual comparable news collection. In their method, sentence length models and lexicon-based models were combined under a maximum likelihood criterion. Smith et al. (2010) found that Wikipedia contains a lot of comparable documents, and adopted a ranking model to select parallel sentence pairs from comparable documents. Bharadwaj et al. (2011) used a SVM classifier with some new features to identify parallel sentences from Wikipedia.

3 Iterative Link-based Parallel Web Pages Mining

As mentioned, the basic idea of our method is that the similarity of two pages can be inferred from their neighbors. This idea is illustrated in Figure 1.

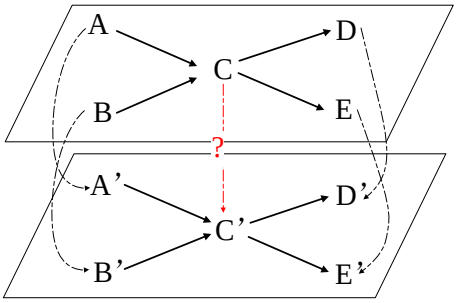


Figure 1 Illustration of the link-based method

In Figure 1, *A*, *B*, *C*, *D* and *E* are some pages in the same language; while *A'*, *B'*, *C'*, *D'* and *E'* are some pages in another language. The solid black arrows indicate the links between these pages. For example, page *A* points to *C*, page *B'* points to *C'* and so on. Then the page set {*A*, *B*, *D*, *E*} is called the neighbors of page *C*. Similarly, the page set {*A'*, *B'*, *D'*, *E'*} contains the neighbors of page *C'*. If the page pairs : <*A*, *A'*>, <*B*, *B'*>, <*D*, *D'*> and <*E*, *E'*> have high translation similarities, then it can be inferred that page *C* and *C'* have a high probability to be a pair of parallel pages. Every page has its own neighbors. For each web page, our method views link-in and link-out hyperlinks as the same. Thus, the linked pages will influence each other in estimating the translation similarity. For example, the similarities of two pairs <*A*, *A'*> and <*C*, *C'*> will influence each other. It is an iterative process. We will elaborate the process in the following sections.

Since our goal is to find parallel pages in a specific website, the key task is to evaluate the translation similarity of two pages (which are in different languages) as accurately as possible. The final similarity of two pages should depend both on their internal similarity and external similarity. The internal similarity means the similarity estimated by using the information in the page itself, such as the structure similarity and the content-based similarity of the two pages. On the other hand, the external similarity of two pages is the similarity depending on their neighbors. The final translation similarity is called the **Enhanced Translation Similarity** (*ETS*). The *ETS* of two pages can be calculated as follows:

$$ETS(e,c) = \alpha \cdot S_{ext}(e,c) + (1-\alpha) \cdot S_{in}(e,c), \alpha \in [0,1] \quad (1)$$

Where, $S_{in}(e,c)$ is the internal translation similarity of two pages: e and c; $S_{ext}(e,c)$ represents the external translation similarity of pages e and c. $ETS(e,c)$ indicates the final similarity of two pages, which combines the internal with external translation similarity.

In this paper, we conduct the experiments on English-Chinese parallel page pair mining. However, our method is language-independent. Thus, it can be applied to other language pairs by only replacing a bilingual lexicon. The symbol e and c always indicate an English page and a Chinese page respectively in this paper. In the following sections, we will describe how to calculate the $S_{in}(e,c)$ and $S_{ext}(e,c)$ step by step.

3.1 Preprocessing

The input of our method is a bilingual website. This paper aims to find English/Chinese parallel pages. So a 3-gram language model is used to identify (or classify) the language of a certain document. The performance of the language identification module achieves 99.5% accuracy through in-house testing. As a result, a set of English pages and a set of Chinese pages are obtained. In order to get the neighbors of a page, for each bilingual website, two networks are constructed based on the hyperlinks, one for English pages and another for Chinese pages.

3.2 The Internal Translation Similarity

Following Resnik and Smith (2003), three features are used to evaluate the internal translation similarity of two pages:

The size ratio of two pages

The length ratio of two documents is the simplest criterion for determining whether two documents are parallel or not. Parallel documents tend to be similar in length. And it is reasonable to assume that for text *E* in one language and text *F* in another language, length(*E*) ≈ *C*•length(*F*), where *C* is a constant that depends on the language pair. Here, the content length of a web page is regarded as its length.

The structure similarity of two pages

The HTML tags describe and control a web page's structure. Therefore, the structure similarity of two pages can be calculated by their HTML tags. Here, the HTML tags of each page are extracted (except the visual tags such as "B", "FONT".) as a linear sequence. Then the structure similarity of two pages is computed by comparing their linearized sequences. In this paper, the LCS algorithm (Dan, 1997) is adopted to find the longest common sequences of the two HTML tag sequences. The ratio of LCS length and the average length of two HTML tag sequences are used as the structure similarity of the two pages.

The content-based translation similarity of two pages

The basic idea is that if two documents are parallel, they will contain word pairs that are mutual translations (Ma, 1999). So the percentage of translation word pairs in the two pages can be considered as the content-based similarity. The translation words of two documents can be extracted by using a bilingual lexicon. Here, for each word in English document, we will try to find a corresponding word in Chinese document.

Finally, the internal translation similarity of two pages is calculated as follows:

$$S_{in}(e,c) = \beta \cdot S_{cb}(e,c) + (1-\beta) \cdot S_{struct}(e,c), \beta \in [0,1] \quad (2)$$

Where, $S_{cb}(e,c)$ and $S_{struct}(e,c)$ are the content-based and structural similarity of page e and c respectively. In addition, the size ratio of two pages is used to filter invalid page pairs.

3.3 The External and Enhanced Translation Similarity

As described above, the external translation similarity of two pages depends on their neighbors:

$$S_{ext}(e,c) = Sim\big(PG(e), PG(c)\big) \quad (3)$$

Where, *PG(x)*, a set of pages, is the neighbors of page *x*. Obviously, the similarity of two sets relies on the similarity of the elements in the two sets. Here, the elements are namely web pages. So, $S_{ext}(e,c)$ equals to $Sim\big(PG(e), PG(c)\big)$, and $Sim\big(PG(e), PG(c)\big)$ depends on $ETS\big(e_i, c_j\big)$ (e_i, c_j belongs to $PG(e)$, $PG(c)$, respectively) and $ETS(e,c)$. According to Equation (1), $ETS(e,c)$ depends on $S_{in}(e,c)$ and $S_{ext}(e,c)$. Therefore, it is a process of iteration. $ETS(e,c)$ will converge after a certain number of iterations. Thus, $ETS^i(e,c)$ is defined as the enhanced similarity of page *e* and *c* after the *i*-th iteration, and the same is for $S^i_{ext}(e,c)$ and $Sim^i\big(PG(e), PG(c)\big)$. $Sim^i\big(PG(e), PG(c)\big)$ is computed by the following algorithm:

Algorithm 1: Estimating the external translation similarity

Input: $PG(e), PG(c)$
Output: $S^i_{ext}(e,c)$

Procedure:

$sum \leftarrow 0$
$e\_set \leftarrow PG(e)$
$c\_set \leftarrow PG(c)$
While e_set and c_set are both not empty:
  $< x, y >$
  $\leftarrow arg\,max_{x \in e\_set,\ y \in c\_set}\ (ETS^{i-1}(x,y))$
  $sum \leftarrow sum +$ $\mathrm{ETS}^{i-1}(x,y)$
  Remove x from e_set
  Remove y from c_set
$S^i_{ext}(e,c) = Sim^i\big(p(e), p(c)\big)$
$= 2 \cdot sum \,/\, (|PG(e)| + |PG(c)|)$

Algorithm 2 Estimating the enhanced translation similarity

Input: P_e, P_c, (*the English and Chinese page set*)
Output: ETS(e, c), $e \in P_e, c \in P_c$
Initialization: Set *ETS*(*e*, *c*) random value or small value

Procedure:

LOOP:
  For each e in P_e :
    For each c in P_c:
      $ETS^i(e,c) = \alpha \cdot S^i_{ext}(e,c) + (1-\alpha) \cdot S_{in}(e,c)$
  Parameters normalization
UNTIL $ETS(e,c)$ is stable

Algorithm 1 tries to find the real parallel pairs from $PG(e)$ and $PG(c)$. The similarity of $PG(e)$ and $PG(c)$ is calculated based on the similarity

values of these pairs. Finally, $ETS(e,c)$ is calculated by the following algorithm 2.

In Algorithm 2, the input P_e and P_c are English and Chinese page sets in a certain bilingual website. We use algorithm 2 to estimate the enhanced translation similarity.

3.4 Find the Parallel Page Pairs

At last, the enhanced translation similarity of every pair is obtained, and the parallel page pairs can be extracted in terms of these similarities:

Algorithm 3 Finding parallel page pairs

Input: P_e, P_c
$ETS(x,y),\ \ x \in P_e, y \in P_c$
MAX_P (or MIN_SIM)

Output: Parallel Page Pairs List : PPL

Procedure:

LOOP:
$< x,y > = \ arg\,max_{x \in P_e, y \in P_c}\,(ETS(x,y))$
Add $< x,y >$ to PPL
Remove x from P_e
Remove y from P_c
UNTIL size of PPL > MAX_P (or $ETS(x,y)$ < MIN_SIM)

This algorithm is similar to Algorithm 1 in each bilingual website. The input MAX_P is an integer threshold which means that only top MAX_P page pairs will be extracted in a certain website. It needs to be noted that MAX_P is always less than $|P_e|$ and $|P_c|$. While the input MIN_SIM is another kind of threshold that is used for extracting page pairs with high translation similarity.

4 Experiments and Analysis

4.1 Experimental setup

Our experiments focus on six bilingual websites. Most of them are selected from HK government websites. All the web pages were retrieved by using a web site download tool: HTTrack[1]. We notice that a small amount of pages doesn't always contain valuable contents. So, we put a threshold (100 bytes in our experiment) on the web pages' content to filter meaningless pages. In order to evaluate our method, the bilingual page pairs of each website are annotated by a human annotator. Finally, we got 23109 pages and 11684 bilingual page pairs in total for testing. The basic information of these websites is listed in Table 1.

It's time-consuming to annotate whether two pages is parallel or not. Note that if a website contains *N* English pages and *M* Chinese pages, an annotator has to label *N*M* page pairs. To the best of our knowledge, there is no large scale and public parallel page pair dataset with human annotation. So we try to build a reliable and large-scale dataset.

In our experiments, *URL* similarity is used to reduce the workload for annotation. For a certain website, firstly, we obtain its *URL* pattern between English and Chinese pages manually. For example, in the website "www.gov.hk", the *URL* pairs like:

http://www.gov.hk/en/about/govdirectory/ (English)

http://www.gov.hk/sc/about/govdirectory/ (Chinese)

The *URL* pairs always point to a pair of parallel pages. So <"/en/","/sc/"> is considered as a *URL* pattern that was used to find parallel pages. For the other *URLs* that can't match the pattern, we have to label them by hand. The column "No pattern pairs" in Table 1 shows that the number of parallel page pairs which mismatch any patterns.

Table 1 Number of pages and bilingual page pairs of each websites

Site ID	En/Ch pages	Total pairs	No pattern pairs	URL
S1	1101/1098	1092	20	www.gov.hk
S2	501/497	487	7	www.customs.gov.hk
S3	995/775	768	12	www.sbc.edu.sg
S4	4085/3838	3648	4	www.swd.gov.hk
S5	660/637	637	0	www.landsd.gov.hk
S6	4733/4626	4615	8	www.td.gov.hk
total	12075/11471	11684	51	

Each website listed in Table 1 has a *URL* pattern for most parallel web pages. Some previous researches used the *URL* similarity or patterns to find parallel page pairs. However, due to the diversity of web page styles and website maintenance mechanisms, bilingual websites adopt varied naming schemes for parallel documents (Shi, et al, 2006). The effect of *URL* pattern-based mining always depends on the style of website. In order to build a large dataset, the *URL* pattern is not used in our method. Our method is able to handle bilingual websites without *URL* pattern rules.

In addition, an English-Chinese dictionary with 64K words pairs is used in our experiments. Algorithm 3 needs a threshold MAX_P or

[1] http://www.httrack.com/

MIN_SIM. It is very hard to tune the MIN_SIM because it varies a lot in different websites and language pairs. However, Table 1 shows that the number of parallel pages is smaller than that of English and Chinese pages. Here, for each website, the MAX_P is set to the number of Chinese pages (which is always smaller than that of English pages). In this way, the precision will never reach 100%, but it is more practical in a real application. As a result, in some experiments, we only report the F-score, and the precision and recall can be calculated as follows:

$$Precision = \frac{F_{score} \cdot (N_{Pairs} + MAX\_P)}{2 \cdot MAX\_P} \quad (4)$$

$$Recall = \frac{F_{score} \cdot (N_{Pairs} + MAX\_P)}{2 \cdot N_{Pairs}} \quad (5)$$

Where, N_{Pairs} for each website is listed in the "Total pairs" column of Table 1.

4.2 Results and Analysis

Performance of the Baseline

Let's start by presenting the performance of a baseline method as follows. The baseline only employs the internal translation similarity for parallel web pages mining. Algorithm 3 is also used to get the page pairs in baseline system. Here, the input $ETS(x, y)$ is replaced by $S_{in}(x, y)$. The parameter β in Equation 2 is a discount factor. For different β values, the performance of baseline system on six websites is shown in Figure 2. In the Figure 2, it shows that when β is set to 0.6, the baseline system achieves the best performance. The precision, recall and F-score are 85.84%, 87.55% and 86.69% respectively. So in the following experiments, we always set β to 0.6.

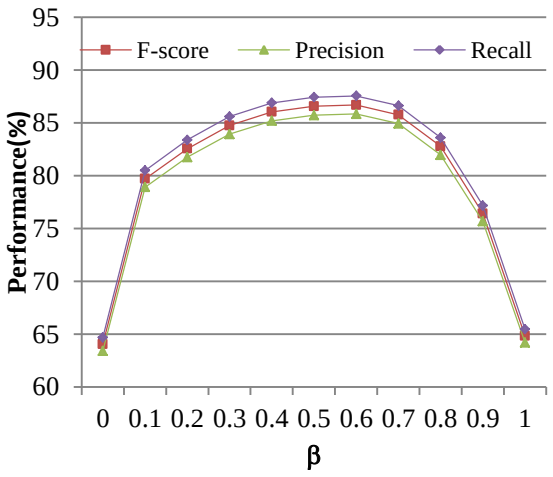


Figure 2 Performances of baseline system with different β value

Performance of Our Method

As described in Section 3, our method combines the internal with external translation similarity in estimating the final translation similarity (i.e., ETS) of two pages. So, the discount factor α in Equation (1) is important in our method. Besides, as shown in Algorithm 2, the iterative algorithm is used to calculate the similarity. Then, one question is that how many iterations are required in our algorithm. Figure 3 shows the performance of our method on each website. Its horizontal axis represents the number of iterations and the vertical axis represents the F-score. And for each website, the F-scores with different α (range from 0.2 to 0.8) are also reported in this figure. From Figure 3, it is very easy to find that the best iteration number is 3. For almost all the websites, the performance of our method achieves the maximal values and converges after the third iteration. In addition, Figure 3 also indicates that our method is robust for different websites. In the following experiments, the iteration number is set to 3.

Next, let's turn to the discount factor α. Figure 4 reports the experimental results on the whole dataset. Here, the horizontal axis represents the discount factor α and the vertical axis represents the F-score. $\alpha = 0$ means that only the internal similarity is used in the algorithm, so the F-score equals to that in Figure 2 when $\beta = 0.6$. On the contrary, $\alpha = 1$ means that only the external similarity is used in the method, and the F-score is 80.20%. The performance is lower than the baseline system when only the external link information is used, but it is much better than the performance of the content-based method and structure-based method whose F-scores are 64.82% and 64.0% respectively. Besides, it is shown from Figure 4, the performance is improved significantly when the internal and external similarity measures are combined together. Furthermore, it is somewhat surprising that the discount factor α is not important as we previously expected. In fact, if we discard the cases that α equals to 0 or 1, the difference between the maximum and minimum F-score will be 0.76% which is very small. This finding indicates that the internal and external similarity can easily be combined and we don't need to make many efforts to tune this parameter when our method is applied to other websites. The reason of this phenomenon is that, no matter how much weight (i.e., 1- α) was assigned to the internal similarity, the internal similarity always provides a relatively good initial

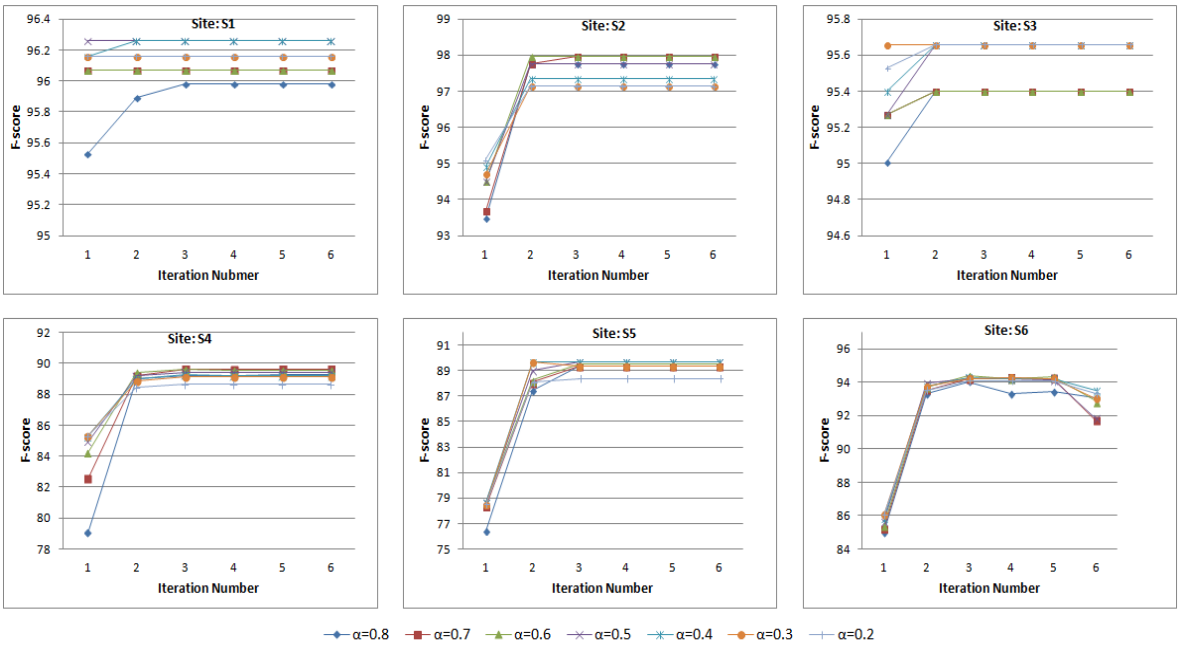


Figure 3 Experiment results of our method on each website

iterative direction. In the following experiments, the parameter α is set to 0.6.

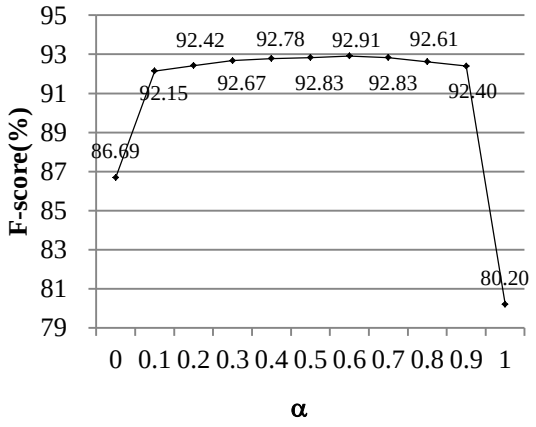


Figure 4 The F-scores of our method with different the value of α

The weight of pages

The weight of the neighbor pages should also be considered. For example, in the most websites, it is very common that most of the web pages contain a hyperlink which points to the homepage of the website. While in most of the English/Chinese websites, almost every English page will link to the English homepage and each Chinese page will point to Chinese homepage. The English and Chinese homepages are probably parallel, but they will be helpless to find parallel web pages, because they are neighbors of almost every page in the site. On the contrary, sometimes the parallel homepages have negative effects on finding parallel pages They will increase the translation similarity of two pages which are not indeed mutual translations. So it is necessary to amend the Algorithm 1.

The weight of each page is calculated according to its popularity:

$$w(p) = log\frac{N+c}{Freq(p)+c} \quad (6)$$

where $w(p)$ indicates the weight of page p, N is the number of all pages, $Freq(p)$ is the number of pages pointing to page p and c is a constant for smoothing.

In this paper, the weights of pages are used in two ways:

Weight 1: The 9th line of Algorithm 1 is amended by the page weight as follows:

$sum \leftarrow sum + ETS^{i-1}(x,y) \cdot (w(x)+w(y))/2$

Weight 2: The pages with low weight are removed from the input of Algorithm 1.

The experiment results are shown in Table 2.

Table 2 The effect of page weight

Type	No Weight	Weight 1	Weight 2
F-score (%)	92.91	92.78	92.75

Surprisingly, no big differences are found after the introduction of the page weight. The side effect of popular pages is not so large in our method. In the neighbor pages of a certain page, the popular pages are the minority. Besides, the iterative process makes our method more stable and robust.

The impact of the size of bilingual lexicon

The baseline system mainly combines the content-based similarity with structure similarity.

And two kinds of similarity measures are also used in our method. As Ma and Liberman (1999) pointed out, not all translators create translated pages that look like the original page which means that the structure similarity does not always work well. Compared to the structure similarity, the content-based is more reliable and has wider applicability. Furthermore, the bilingual lexicon is the only information that relates to the language pairs, and other features (such as structure and link information) are all language independent. So, it's important to investigate the effect of lexicon size in our method. We test the performance of our method with different size of the bilingual dictionary. The experiment results are shown in Figure 5. In this figure, the horizontal axis represents the bilingual lexicon size and the vertical axis represents the F-score. With the decline of the lexicon size, the performances of both the baseline method and our method are decreased. However, we can find that the descent rate of our method is smaller than that of the baseline. It indicates that our method does not need a big bilingual lexicon which is good news for the low-resource language pairs.

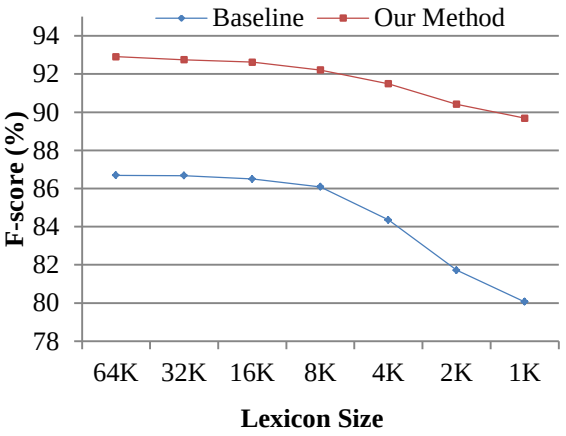


Figure 5 The impact of the size of bilingual lexicon

Error analysis

Errors occur when the two pages are similar in terms of structure, content and their neighbors. For example, Figure 6 illustrates a typical web page structure. There are 5 parts in the web page: U, L, M, R and B. Part M always contains the main content of this page. While part U, L, R and B always contain some hyperlinks such as "home" in part U and "About us" in part B. Links in L and R sometimes relate to the content of the page. For such a kind of non-parallel page pairs, let's assume that the two pages have the same structure (as shown in Figure 6). In addition, their content part M is very short and contains the same or related topics. As a result, the links in other 4 parts are likely to be similar. In this case, our method is likely to regard the two pages as parallel.

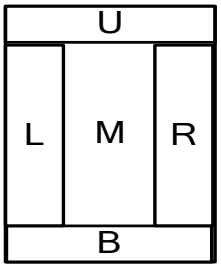


Figure 6 A typical web page structure

There are about 920 errors when our system obtains its best performance. By carefully investigating the error page pairs, we find that more than 90% errors fall into the category discussed above. The websites used in our experiments mainly come from Hong Kong government websites. Some government departments regularly publish quarterly or monthly work reports on one issue through their websites. These reports look very similar except the publish date and some data in them. The other 10% errors happen because of the particularity of the web pages, e.g. very short pages, broken pages and so on.

5 Conclusions and Future Work

Parallel corpora are valuable resources for a lot of NLP research problems and applications, such as MT and CLIR. This paper introduces an efficient and effective solution to bilingual language processing. We first explore how to extract parallel page pairs in bilingual websites with link information between web pages. Firstly, we hypothesize that the translation similarity of pages should be based on both internal and external translation similarity. Secondly, a novel iterative method is proposed to verify parallel page pairs. Experimental results show that our method is much more effective than the baseline system with 6.2% improvement on F-Score. Furthermore, our method has some significant contributions. For example, compared to previous work, our method does not depend on bilingual lexicons, and the parameters in our method have little effect on the final performance. These features improve the applicability of our method.

In the future work, we will study some method on extracting parallel resource from existing parallel page pairs, which are challenging tasks due to the diversity of page structures and styles. Besides, we will evaluate the effectiveness of our mined data on MT or other applications.

Acknowledgments

This research work has been sponsored by National Natural Science Foundation of China (Grants No.61373097 and No.61272259), one National Natural Science Foundation of Jiangsu Province (Grants No.BK2011282), one Major Project of College Natural Science Foundation of Jiangsu Province (Grants No.11KJA520003) and one National Science Foundation of Suzhou City (Grants No.SH201212).

The corresponding author of this paper, according to the meaning given to this role by School of computer science and technology at Soochow University, is Yu Hong

Reference

Chen, Jiang and Jianyun Nie. 2000. Automatic construction of parallel English-Chinese corpus for cross-language information retrieval. Proceedings of the sixth conference on Applied Natural Language Processing, 21–28.

Resnik, Philip and Noah A. Smith. 2003. The Web as a Parallel Corpus. Meeting of the Association for Computational Linguistics 29(3). 349–380.

Kit, Chunyu and Jessica Yee Ha Ng. 2007. An Intelligent Web Agent to Mine Bilingual Parallel Pages via Automatic Discovery of URL Pairing Patterns. Web Intelligence and Intelligent Agent Technology Workshops, 526–529.

Zhang, Ying, Ke Wu, Jianfeng Gao and Phil Vines. 2006. Automatic Acquisition of Chinese-English Parallel Corpus from the Web. Joint Proceedings of the Association for Computational Linguistics and the International Conference on Computational Linguistics, 420–431.

Nie, Jianyun, Michel Simard, Pierre Isabelle and Richard Durand. 1999. Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the Web. Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, 74–81.

Ma, Xiaoyi and Mark Y. Liberman. 1999. BITS: A Method for Bilingual Text Search over the Web. Machine Translation Summit VII.

Chen, Jisong, Rowena Chau and Chung-Hsing Yeh. 2004. Discovering Parallel Text from the World Wide Web. The Australasian Workshop on Data Mining and Web Intelligence, vol. 32, 157–161. Dunedin, New Zealand.

Jiang, Long, Shiquan Yang, Ming Zhou, Xiaohua Liu and Qingsheng Zhu. 2009. Mining Bilingual Data from the Web with Adaptively Learnt Patterns. Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, vol. 2, 870–878.

Yanhui Feng, Yu Hong, Zhenxiang Yan, Jianmin Yao and Qiaoming Zhu. 2010. A novel method for bilingual web page acquisition from search engine web records. Proceedings of the 23rd International Conference on Computational Linguistics: Posters, 294–302.

Zhao, Bing and Stephan Vogel. 2002. Adaptive Parallel Sentences Mining from Web Bilingual News Collection. IEEE International Conference on Data Mining, 745–748.

Smith, Jason R., Chris Quirk and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 403–411.

Bharadwaj, Rohit G. and Vasudeva Varma. 2011. Language independent identification of parallel sentences using wikipedia. Proceedings of the 20th International Conference Companion on World Wide Web, 11–12. Hyderabad, India.

Gusfield, Dan. 1997. Algorithms on Strings, Trees and Sequences: Computerss Science and Computational Biology. Cambridge University Press

Shi, Lei, Cheng Niu, Ming Zhou and Jianfeng Gao. 2006. A DOM Tree Alignment Model for Mining Parallel Data from the Web. Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, 489–496.

Human Effort and Machine Learnability in Computer Aided Translation

Spence Green, Sida Wang, Jason Chuang,* Jeffrey Heer,* Sebastian Schuster,
and **Christopher D. Manning**

Computer Science Department, Stanford University
{spenceg,sidaw,sebschu,manning}@stanford.edu

*Computer Science Department, University of Washington
{jcchuang,jheer}@uw.edu

Abstract

Analyses of computer aided translation typically focus on either frontend interfaces and human effort, or backend translation and machine learnability of corrections. However, this distinction is artificial in practice since the frontend and backend must work in concert. We present the first holistic, quantitative evaluation of these issues by contrasting two assistive modes: post-editing and interactive machine translation (MT). We describe a new translator interface, extensive modifications to a phrase-based MT system, and a novel objective function for re-tuning to human corrections. Evaluation with professional bilingual translators shows that post-edit is faster than interactive at the cost of translation quality for French-English and English-German. However, re-tuning the MT system to interactive output leads to larger, statistically significant reductions in HTER versus re-tuning to post-edit. Analysis shows that tuning directly to HTER results in fine-grained corrections to subsequent machine output.

1 Introduction

The goal of machine translation has always been to reduce human effort, whether by partial assistance or by outright replacement. However, preoccupation with the latter—fully automatic translation—at the exclusion of the former has been a feature of the research community since its first nascent steps in the 1950s. Pessimistic about progress during that decade and future prospects, Bar-Hillel (1960, p.3) argued that more attention should be paid to a "machine-post-editor partnership," whose decisive problem is "the region of optimality in the continuum of possible divisions of labor." Today, with human-quality, fully automatic machine translation (MT) elusive still, that decades-old recommendation remains current.

This paper is the first to look at both sides of the partnership in a single user study. We compare two common flavors of machine-assisted translation: post-editing and interactive MT. We analyze professional, bilingual translators working in both modes, looking first at user productivity. Does the additional machine assistance available in the interactive mode affect translation time and/or quality?

Then we turn to the machine side of the partnership. The user study results in corrections to the baseline MT output. Do these corrections help the MT system, and can it learn from them quickly enough to help the user? We perform a re-tuning experiment in which we directly optimize human Translation Edit Rate (HTER), which correlates highly with human judgments of fluency and adequacy (Snover et al., 2006). It is also an intuitive measure of human effort, making fine distinctions between 0 (no editing) and 1 (complete rewrite).

We designed a new user interface (UI) for the experiment. The interface places demands on the MT backend—not the other way around. The most significant new MT system features are *prefix decoding*, for translation completion based on a user prefix; and *dynamic phrase table augmentation*, to handle target out-of-vocabulary (OOV) words. Discriminative re-tuning is accomplished with a novel cross-entropy objective function.

We report three main findings: (1) post-editing is faster than interactive MT, corroborating Koehn (2009a); (2) interactive MT yields higher quality translation when baseline MT quality is high; and (3) re-tuning to interactive feedback leads to larger held-out HTER gains relative to post-edit. Together these results show that a human-centered approach to computer aided translation (CAT) may involve tradeoffs between human effort and machine learnability. For example, if speed is the top priority, then a design geared toward post-editing

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1225–1236,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

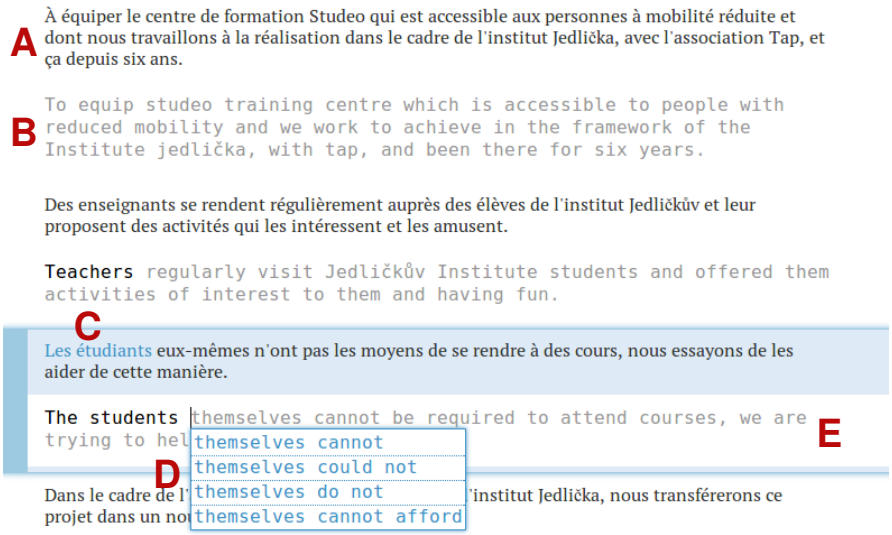


Figure 1: Main translation interface. The user sees the full document context, with French source inputs (A) interleaved with suggested English translations (B). The sentence in focus is indicated by the blue rectangle, which can be moved via two hot keys. Source coverage (C) of the user prefix—shaded in blue—updates as the user works, as do autocomplete suggestions (D) and a full completion (E).

is appropriate. However, if reductions in HTER ultimately correspond to lower human effort, then investing slightly more time in the interactive mode, which results in more learnable output, may be optimal. Mixed UI designs may offer a compromise. Code and data from our experiments are available at: `http://nlp.stanford.edu/software/phrasal/`

A holistic comparison with human subjects necessarily involves many moving parts. Section 2 briefly describes the interface, focusing on NLP components. Section 3 describes changes to the backend MT system. Section 4 explains the user study, and reports human translation time and quality results. Section 5 describes the MT re-tuning experiment. Analysis (section 6) and related work (section 7) round out the paper.

2 New Translator User Interface

Figure 1 shows the translator interface, which is designed for expert, bilingual translators. Previous studies have shown that expert translators work and type quickly (Carl, 2010), so the interface is designed to be very responsive, and to be primarily operated by the keyboard. Most aids can be accessed via either typing or four hot keys. The current design focuses on the point of text entry and does not include conventional translator workbench features such as workflow management, spell checking, and text formatting tools.

In the trivial post-edit mode, the interactive aids are disabled and a 1-best translation pre-populates the text entry box.

We have described the HCI-specific motivations for and contributions of this new interface in Green et al. (2014c). This section focuses on interface elements built on NLP components.

2.1 UI Overview and Walkthrough

We categorized interactions into three groups: **source comprehension**: word lookups, *source coverage highlighting*; **target gisting**: 1-best translation, *real-time target completion*; **target generation**: real-time autocomplete, *target reordering*, insert complete translation. The interaction designs are novel; those in *italic* have, to our knowledge, never appeared in a translation workbench.

Source word lookup When the user hovers over a source word, a menu of up to four ranked translation suggestions appears (Figure 2). The menu is populated by a phrase-table query of the word plus one token of left context. This query usually returns in under 50ms. The width of the horizontal bars indicates confidence, with the most confident suggestion 'regularly' placed at the bottom, nearest to the cursor. The user can insert a translation suggestion by clicking.

Source coverage highlighting The source coverage feature (Figure 1C) helps the user quickly find untranslated words in the source. The interaction is

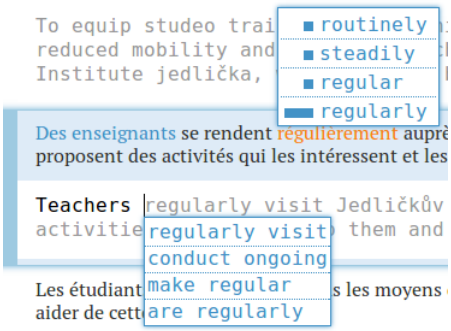


Figure 2: Source word lookup and target autocomplete menus. The menus show different suggestions. The word lookup menu (top) is not dependent on the target context *Teachers*, whereas the autocomplete dropdown (bottom) is.

based on the word alignments between source and target generated by the MT system. We found that the raw alignments are too noisy to show users, so the UI filters them with phrase-level heuristics.

1-best translation The most common use of MT output is *gisting* (Koehn, 2010, p.21). The gray text below each black source input shows the best MT system output (Figure 1B).

Real-time target completion When the user extends the black prefix, the gray text will update to the most probable completion (Figure 1E). This update comes from decoding under the full translation model. All previous systems performed inference in a word lattice.

Real-time autocomplete The autocomplete dropdown at the point of text entry is the main translation aid (Figures 1D and 2). Each real-time update actually contains a distinct 10-best list for the full source input. The UI builds up a trie from these 10-best lists. Up to four distinct suggestions are then shown at the point of translation. The suggestion length is based on a syntactic parse of the fixed source input. As an offline, pre-processing step, we parse each source input with Stanford CoreNLP (Manning et al., 2014). The UI combines those parses with word alignments from the full translation suggestions to project syntactic constituents to each item on the n-best list. Syntactic projection is a very old idea that underlies many MT systems (see: Hwa et al. (2002)). Here we make novel use of it for suggestion prediction filtering.[1] Presently, we project noun phrases, verb phrases (minus the verbal arguments), and prepositional phrases. Crucially, these units are natural to humans, unlike statistical target phrases.

Target Reordering Carl (2010) showed that expert translators tend to adopt *local planning*: they read a few words ahead and then translate in a roughly online fashion. However, word order differences between languages will necessarily require longer range planning and movement. To that end, the UI supports keyboard-based reordering. Suppose that the user wants to move a span in gray text to the insertion position for editing. Typing the prefix of this string will update the autocomplete dropdown with matching strings from the gray text. Consequently, sometimes the autocomplete dropdown will contain suggestions from several positions in the full suggested translation.

Insert complete translation The user can insert the full completion via a hot key. Notice that if the user presses this hot key immediately, all gray text becomes black, and the interface effectively switches to post-edit mode. This feature greatly accelerates translation when the MT is mostly correct, and the user only wants to make a few changes.

2.2 User Activity Logging

A web application serves the Javascript-based interface, relays translation requests to the MT system, and logs user records to a database. Each user record is a tuple of the form $(f, \hat{e}, h, u)$, where f is the source sequence, $\hat{e}$ is the latest 1-best machine translation of f, h is the correction of $\hat{e}$, and u is the log of interaction events during the translation session. Our evaluation corpora also include independently generated references e for each f.

3 Interactive MT Backend

Now we describe modifications to Phrasal (Green et al., 2014b), the phrase-based MT system that supports the interface. Phrasal follows the log-linear approach to phrase-based translation (Och and Ney, 2004) in which the decision rule has the familiar linear form

$$\hat{e} = \arg\max_{e} w^{\top} \phi(e, f) \quad (1)$$

[1]The classic TransType system included a probabilistic prediction length component (Foster et al., 2002), but we find that the simpler projection technique works well in practice.

where $w \in \mathbb{R}^d$ is the model weight vector and $\phi(\cdot) \in \mathbb{R}^d$ is a feature map.

3.1 Decoding

The default Phrasal search algorithm is cube pruning (Huang and Chiang, 2007). In the post-edit condition, search is executed as usual for each source input, and the 1-best output is inserted into the target textbox. However, in interactive mode, the full search algorithm is executed *each time* the user modifies the partial translation. Machine suggestions $\hat{e}$ must match user prefix h. Define indicator function $\text{pref}(\hat{e}, h)$ to return true if $\hat{e}$ begins with h, and false otherwise. Eq. 1 becomes:

$$\hat{e} = \underset{e \text{ s.t. } \text{pref}(e,h)}{\arg\max} \; w^\top \phi(e, f) \qquad (2)$$

Cube pruning can be straightforwardly modified to satisfy this constraint by simple string matching of candidate translations. Also, the pop limit must be suspended until at least one legal candidate appears on each beam, or the priority queue of candidates is exhausted. We call this technique *prefix decoding*.[2]

There is another problem. Human translators are likely to insert unknown target words, including new vocabulary, misspellings, and typographical errors. They might also reorder source text so as to violate the phrase-based distortion limit. To solve these problems, we perform *dynamic phrase table augmentation*, adding new synthetic rules specific to each search. Rules allowing any source word to align with any unseen or ungeneratable (due to the distortion limit) target word are created.[3] These synthetic rules are given rule scores lower than any other rules in the set of queried rules for that source input f. Then candidates are allowed to compete on the beam. Candidates with spurious alignments will likely be pruned in favor of those that only turn to synthetic rules as a last resort.

3.2 Tuning

We choose BLEU (Papineni et al., 2002) for baseline tuning to independent references, and HTER for re-tuning to human corrections. Our rationale is as follows: Cer et al. (2010) showed that BLEU-tuned systems score well across automatic metrics and also correlate with human judgment better than systems tuned to other metrics. Conversely, systems tuned to edit-distance-based metrics like TER tend to produce short translations that are heavily penalized by other metrics.

When human corrections become available, we switch to HTER, which correlates with human judgment and is an interpretable measure of editing effort. Whereas TER is computed as $\text{TER}(e, \hat{e})$, HTER is $\text{HTER}(h, \hat{e})$. HBLEU is an alternative, but since BLEU is invariant to some permutations (Callison-Burch et al., 2006), it is less interpretable. We find that it also does not work as well in practice.

We previously proposed a fast, online tuning algorithm (Green et al., 2013b) based on AdaGrad (Duchi et al., 2011). The default loss function is expected error (EE) (Och, 2003; Cherry and Foster, 2012). Expected BLEU is an example of EE, which we found to be unstable when switching metrics. This may result from direct incorporation of the error metric into the gradient computation.

To solve this problem, we propose a *cross-entropy loss* which, to our knowledge, is new in MT. Let $\hat{E} = \{\hat{e}_i\}_{i=1}^n$ be an n-best list ranked by a gold metric $G(e, \hat{e}) \geq 0$. Assume we have a preference of a higher G (e.g., BLEU or $1-$HTER). Define the model distribution over $\hat{E}$ as $q(\hat{e}|f) \propto \exp[w^\top \phi(\hat{e}, f)]$ normalized so that $\sum_{\hat{e} \in \hat{E}} q(\hat{e}|f) = 1$; q indicates how much the model prefers each translation. Similarly, define $p(\hat{e}|f)$ based on any function of the gold metric so that $\sum_{\hat{e} \in \hat{E}} p(\hat{e}|f) = 1$; p indicates how much the metric prefers each translation. We choose a DCG-style[4] parameterization that skews the p distribution toward higher-ranked items on the n-best list: $p(\hat{e}_i|f) \propto G(e, \hat{e}_i)/\log(1+i)$ for the ith ranked item. The cross-entropy (CE) loss function is:

$$\ell_{\text{CE}}(w; E) = \mathbb{E}_{p(\hat{e}|f)}[-\log(q(\hat{e}|f)] \qquad (3)$$

It turns out that if p is simply the posterior distribution of the metric, then this loss is related to the log of the standard EE loss:[5]

$$\ell_{\text{EE}}(w; E) = -\log[\mathbb{E}_{p(\hat{e}|f)}[q(\hat{e}|f)]] \qquad (4)$$

We can show that $\ell_{\text{CE}} \geq \ell_{\text{EE}}$ by applying Jensen's inequality to the function $-\log(\cdot)$. So minimizing ℓ_{CE} also minimizes a convex upper bound of the log expected error. This convexity given the n-

[2] Och et al. (2003) describe a similar algorithm for word graphs.

[3] Ortiz-Martínez et al. (2009) describe a related technique in which *all* source and target words can align, with scores set by smoothing.

[4] Discounted cumulative gain (DCG) is widely used in information retrieval learning-to-rank settings. n-best MT learning is standardly formulated as a ranking task.

[5] For expected error, $p(\hat{e}_i) = G(e, \hat{e}_i)$ is not usually normalized. Normalizing p adds a negligible constant.

best list does not mean that the overall MT tuning loss is convex, since the n-best list contents and order depend on the parameters w. However, all regret bounds and other guarantees of online convex optimization would now apply in the CE case since $\ell_{\text{CE},t}(w_{t-1}; E_t)$ is convex for each t. This is attractive compared to expected error, which is non-convex even given the n-best list. We empirically observed that CE converges faster and is less sensitive to hyperparameters than EE.

Faster decoding trick We found that online tuning also permits a trick that speeds up decoding during deployment. Whereas the Phrasal default beam size is 1,200, we were able to reduce the beam size to 800 and run the tuner longer to achieve the same level of translation quality. For example, at the default beam size for French-English, the algorithm converges after 12 iterations, whereas at the lower beam size it achieves that level after 20 iterations. In our experience, batch tuning algorithms seem to be more sensitive to the beam size.

3.3 Feature Templates

The baseline system contains 19 dense feature templates: the nine Moses (Koehn et al., 2007) baseline features, the eight-feature hierarchical lexicalized re-ordering model of Galley and Manning (2008), the (log) count of each rule in the bitext, and an indicator for unique rules. We found that sparse features, while improving translation quality, came at the cost of slower decoding due to feature extraction and inner products with a higher dimensional feature map ϕ. During prototyping, we observed that users found the system to be sluggish unless it responded in approximately 300ms or less. This budget restricted us to dense features.

When re-tuning to corrections, we extract features from the user logs u and add them to the baseline dense model. For each tuning input f, the MT system produces candidate derivations $d = (f, \hat{e}, a)$, where a is a word alignment. The user log u also contains the *last MT derivation*[6] accepted by the user $d_u = (f, \hat{e}_u, a_u)$. We extract features by comparing d and d_u. The heuristic we take is *intersection*: $\phi(d) \leftarrow \phi(d) \cap \phi(d_u)$.

Lexicalized and class-based alignments Consider the alignment in Figure 3. We find that user derivations often contain many unigram rules,

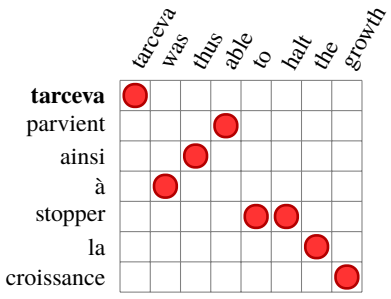


Figure 3: User translation word alignment obtained via prefix decoding and dynamic phrase table augmentation.

which are less powerful than larger phrases, but nonetheless provide high-precision lexical choice information. We fire indicators for both unigram links and multiword cliques. We also fire class-based versions of this feature.

Source OOV blanket Source OOVs are usually more frequent when adapting to a new domain. In the case of European languages—our experimental setting—many of the words simply transfer to the target, so the issue is where to position them. In Figure 3, the proper noun *tarceva* is unknown, so the decoder OOV model generates an identity translation rule. We add features in which the source word is concatenated with the left, right, and left/right contexts in the target, e.g., {`<s>-tarceva, tarceva-was, <s>-tarceva-was`}. We also add versions with target words mapped to classes.

3.4 Differences from Previous Work

Our backend innovations support the UI and enable feature-based learning from human corrections. In contrast, most previous work on incremental MT learning has focused on extracting new translation rules, language model updating, and modifying translation model probabilities (see: Denkowski et al. (2014a)). We regard these features as additive to our own work: certainly extracting new, unseen rules should help translation in a new domain. Moreover, to our knowledge, all previous work on updating the weight vector w has considered *simulated post-editing*, in which the independent references e are substituted for corrections h. Here we extract features from and re-tune to actual corrections to the baseline MT output.

[6]Extracting features from intermediate user editing actions is an interesting direction for future work.

4 Translation User Study

We conducted a human translation experiment with a 2 (translation conditions) $\times$ n (source sentences) mixed design, where n depended on the language pair. Translation conditions (post-edit and interactive) and source sentences were the independent variables (factors). Experimental subjects saw all factor levels, but not all combinations, since one exposure to a sentence would influence another.

Subjects completed the experiment remotely on their own hardware. They received personalized login credentials for the translation interface, which administered the experiment. Subjects first completed a demographic questionnaire about prior experience with CAT and language proficiency. Next, they completed a training module that included a 4-minute tutorial video and a practice "sandbox" for developing proficiency with the UI. Then subjects completed the translation experiment. Finally, they completed an exit questionnaire.

Unlike the experiment of Koehn (2009a), subjects were under time pressure. An idle timer prevented subjects from pausing for more than three minutes while the translator interface was open. This constraint eliminates a source of confound in the timing analysis.

We randomized the order of translation conditions and the assignment of sentences to conditions. At most five sentences appeared per screen, and those sentences appeared in the source document order. Subjects could move among sentences within a screen, but could not revise previous screens. Subjects received untimed breaks both between translation conditions and after about every five screens within a translation condition.

4.1 Linguistic Materials

We chose two language pairs: French-English (Fr-En) and English-German (En-De). Anecdotally, French-English is an easy language pair for MT, whereas English-German is very hard due to reordering and complex German morphology.

We chose three text genres: software, medical, and informal news. The software text came from the graphical interfaces of Autodesk AutoCAD and Adobe Photoshop. The medical text was a drug review from the European Medicines Agency. These two data sets came from TAUS[7] and included independent reference translations. The informal news text came from the WMT 2013 shared task test set (Bojar et al., 2013). The evaluation corpus was constructed from equal proportions of the three genres.

The Fr-En dataset contained 3,003 source tokens (150 segments); the En-De dataset contained 3,002 (173 segments). As a rule of thumb, a human translator averages about 2,700 source tokens per day (Ray, 2013, p.36), so the experiment was designed to replicate a slightly demanding work day.

4.2 Selection of Subjects

For each language pair, we recruited 16 professional, freelance translators on Proz, which is the largest online translation community.[8] We posted ads for both language pairs at a fixed rate of $0.085 per source word, an average rate in the industry. In addition, we paid $10 to each translator for completing the training module. All subjects had significant prior experience with a CAT workbench.

4.3 Results

We analyze the translation conditions in terms of two response variables: time and quality. We excluded one Fr-En subject and two En-De subjects from the models. One subject misunderstood the instructions of the experiment and proceeded without clarification; another skipped the training module entirely. The third subject had a technical problem that prevented logging. Finally, we also filtered segment-level sessions for which the log of translation time was greater than 2.5 standard deviations from the mean.

4.3.1 Translation Time

We analyze time with a linear mixed effects model (LMEM) estimated with the `lme4` (Bates, 2007) R package. When experimental factors are sampled from larger populations—e.g., humans, sentences, words—LMEMs are more robust to type II errors (see: Baayen et al. (2008)). The log-transformed time is the response variable and translation condition is the main independent variable. The maximal random effects structure (Barr et al., 2013) contains intercepts for subject, sentence id, and text genre, each with random slopes for translation condition.

We found significant main effects for translation condition (Fr-En, $p < 0.05$; En-De, $p < 0.01$). The orientation of the coefficients indicates that interactive is slower for both language pairs. For Fr-En, the LMEM predicts a mean time (intercept) of 46.0 sec/sentence in post-edit vs. 54.6 sec/sentence

[7] `http://www.tausdata.org/`

[8] `http://www.proz.com`

	Fr-En		En-De	
	TER	**HTER**	**TER**	**HTER**
post-edit	47.32	23.51	56.16	37.15
interactive	47.05	24.14	55.89	39.55

Table 1: Automatic assessment of translation quality. Here we change the definitions of TER and HTER slightly. TER is the human translations compared to the independent references. HTER is the baseline MT compared to the human corrections.

in interactive, or 18.7% slower. For En-De, the mean is 51.8 sec/sentence vs. 63.3 sec/sentence in interactive, or 22.1% slower.

We found other predictive covariates that reveal more about translator behavior. When subjects did not edit the MT suggestion, they were significantly faster. When token edit distance from MT or source input length increased, they were slower. Subjects were usually faster as the experiment progressed, a result that may indicate increased proficiency with practice. Note that all subjects reported professional familiarity with post-edit, whereas the interactive mode was entirely new to them. In the exit survey many translators suggested that with more practice, they could have been as fast in the interactive mode.[9]

4.3.2 Translation Quality

We evaluated translation quality with both automatic and manual measures. Table 1 shows that in the interactive mode, TER is lower and HTER is higher: subjects created translations closer to the references (lower TER), but performed more editing (higher HTER). This result suggests better translations in the interactive mode.

To confirm that intuition, we elicited judgments from professional human raters. The setup followed the manual quality evaluation of the WMT 2014 shared task (Bojar et al., 2014). We hired six raters—three for each language pair—who were paid between $15–20 per hour. The raters logged into Appraise (Federmann, 2010) and for each source segment, ranked five randomly selected translations. From these 5-way rankings we extracted pairwise judgments $\pi = \{<, =\}$, where $u_1 < u_2$ indicates that subject u_1 provided a better translation than subject u_2 for a given source input (Table 2).

[9]See (Green et al., 2014c) for significance levels of the other covariates along with analysis of subject learning rates, subject behavior, and qualitative feedback.

	Fr-En		En-De	
#pairwise	14,211		15,001	
#ties (=)	5,528		2,964	
IAA	0.419	(0.357)	0.407	(0.427)
EW (inter.)	0.512		0.491	

Table 2: Pairwise judgments for the manual quality assessment. Inter-annotator agreement (*IAA*) κ scores are measured with the official WMT14 script. For comparison, the WMT14 IAA scores are given in parentheses. *EW (inter.)* is expected wins of interactive according to Eq. (6).

	Fr-En		En-De	
	sign	p	sign	p
ui (interactive)	$+$	$\bullet$	$-$	
log edit distance	$-$	$\bullet\bullet\bullet$	$+$	$\bullet\bullet\bullet$
gender (female)	$-$		$+$	$\bullet$
log session order	$-$		$+$	$\bullet$

Table 3: LMEM manual translation quality results for each fixed effect with contrast conditions for binary predictors in (). The signs of the coefficients can be interpreted as in ordinary regression. *edit distance* is token-level edit distance from baseline MT. *session order* is the order in which the subject translated the sentence during the experiment. Statistical significance was computed with a likelihood ratio test: $\bullet\bullet\bullet\ p < 0.001$; $\bullet\ p < 0.05$.

In WMT the objective is to rank individual systems; here we need only compare interface conditions. However, we should control for translator variability. Therefore, we build a binomial LMEM for quality. The model is motivated by the simple and intuitive *expected wins* (EW) measure used at WMT. Let S be the set of pairwise judgments and $\text{wins}(u_1, u_2) = |\{(u_1, u_2, \pi) \in S \mid \pi = <\}|$. The standard EW measure is:

$$e(u_1) = \frac{1}{|S|} \sum_{u_1 \neq u_2} \frac{\text{wins}(u_1, u_2)}{\text{wins}(u_1, u_2) + \text{wins}(u_2, u_1)} \quad (5)$$

Sakaguchi et al. (2014) showed that, despite its simplicity, Eq. (5) is nearly as effective as model-based methods given sufficient high-quality judgments. Since we care only about the two translation conditions, we reinterpret the u_i as interface conditions, i.e., $u_1 =$ int and $u_2 =$ pe. We can then disregard

the normalizing term to obtain:

$$e(u_1) = \frac{\text{wins}(u_1, u_2)}{\text{wins}(u_1, u_2) + \text{wins}(u_2, u_1)} \quad (6)$$

which is the expected value of a Bernoulli distribution (so $e(u_2) = 1 - e(u_1)$). The intercept-term of the binomial LMEM will be approximately this value subject to other fixed and random effects.

To estimate the model, we convert each pairwise judgment $u_1 < u_2$ to two examples where the response is 1 for u_1 and 0 for u_2. We add the fixed effects shown in Table 3, where the numeric effects are centered and scaled by their standard deviations. The maximal random effects structure contains intercepts for sentence id nested within subject along with random slopes for interface condition.

Table 3 shows the p-values and coefficient orientations. The models yield probabilities that can be interpreted like Eq. (6) but with all fixed predictors set to 0. For Fr-En, the value for post-edit is 0.472 vs. 0.527 for interactive. For En-De, post-edit is 0.474 vs. 0.467 for interactive. The difference is statistically significant for Fr-En, but not for En-De.

When MT quality was anecdotally high (Fr-En), high token-level edit distance from the initial suggestion decreased quality. When MT was poor (En-De), significant editing improved quality. Female En-De translators were better than males, possibly due to imbalance in the subject pool (12 females vs. 4 males). En-De translators seemed to improve with practice (positive coefficient for *session order*).

The Fr-En results are the first showing an interactive UI that improves over post-edit.

5 MT Re-tuning Experiment

The human translators corrected the output of the BLEU-tuned, baseline MT system. No updating of the MT system occurred during the experiment to eliminate a confound in the time and quality analyses. Now we investigate re-tuning the MT system to the corrections by simply re-starting the online learning algorithm from the baseline weight vector w, this time scoring with HTER instead of BLEU.

Conventional incremental MT learning experiments typically resemble domain adaptation: small-scale baselines are trained and tuned on mostly out-of-domain data, and then re-tuned incrementally on in-domain data. In contrast, we start with large-scale systems. This is more consistent with a professional translation environment where translators receive suggestions from state-of-the-art systems like Google Translate.

	Bilingual		**Monolingual**
	#Segments	***#Tokens***	***#Tokens***
En-De	4.54M	224M	1.7B
Fr-En	14.8M	842M	2.24B

Table 4: Gross statistics of MT training corpora.

	En-De	**Fr-En**
baseline-tune	9,469	8,931
baseline-dev	9,012	9,030
int-tune	680	589
int-test	457	368
pe-tune	764	709
pe-test	492	447

Table 5: Tuning, development, and test corpora (#segments). **tune** and **dev** were used for baseline system preparation. Re-tuning was performed on **int-tune** and **pe-tune**, respectively. We report held-out results on the two test data sets. All sets are supplied with independent references.

5.1 Datasets

Table 4 shows the monolingual and parallel training corpora. Most of the data come from the constrained track of the WMT 2013 shared task (Bojar et al., 2013). We also added 61k parallel segments of TAUS data to the En-De bitext, and 26k TAUS segments to the Fr-En bitext. We aligned the parallel data with the Berkeley Aligner (Liang et al., 2006) and symmetrized the alignments with the grow-diag heuristic. For each target language we used lmplz (Heafield et al., 2013) to estimate unfiltered, 5-gram Kneser-Ney LMs from the concatenation of the target side of the bitext and the monolingual data. For the class-based features, we estimated 512-class source and target mappings with the algorithm of Green et al. (2014a).

The upper part of Table 5 shows the baseline tuning and development sets, which also contained 1/3 TAUS medical text, 1/3 TAUS software text, and 1/3 WMT newswire text (see section 4).

The lower part of Table 5 shows the organization of the human corrections for re-tuning and testing. Recall that for each unique source input, eight human translators produced a correction in each condition. First, we filtered all corrections for which a log u was not recorded (due to technical problems). Second, we de-duplicated the corrections so that each h was unique. Finally, we split the unique (f, h) tuples according to a natural division in the

System	tune	BLEU↑	TER↓	HTER
baseline	bleu	23.12	60.29	44.05
re-tune	hter	22.18	60.85	43.99
re-tune+feat	hter	21.73	*59.71*	**42.35**

(a) En-De int-test results.

System	tune	BLEU↑	TER↓	HTER
baseline	bleu	39.33	45.29	28.28
re-tune	hter	39.99	45.73	26.96
re-tune+feat	hter	*40.30*	45.28	**26.40**

(b) Fr-En int-test results.

Table 6: Main re-tuning results for interactive data. **baseline** is the BLEU-tuned system used in the translation user study. **re-tune** is the baseline feature set re-tuned to HTER on int-tune. **re-tune+feat** adds the human feature templates described in section 3.3. **bold** indicates statistical significance relative to the baseline at $p < 0.001$; *italic* at $p < 0.05$ by the permutation test of Riezler and Maxwell (2005).

data. There were five source segments per document, and each document was rendered as a single screen during the translation experiment. *Segment order was not randomized*, so we could split the data as follows: assign the first three segments of each screen to tune, and the last two to test. This is a clean split with no overlap.

This tune/test split has two attractive properties. First, if we can quickly re-tune on the first few sentences on a screen and provide better translations for the last few, then presumably the user experience improves. Second, source inputs f are repeated—eight translators translated each input in each condition. This means that a reduction in HTER means better average suggestions for multiple human translators. Contrast this experimental design with tuning to the corrections of a single human translator. There the system might overfit to one human style, and may not generalize to other human translators.

5.2 Results

Table 6 contains the main results for re-tuning to interactive MT corrections. For both language pairs, we observe large statistically significant reductions in HTER. However, the results for BLEU and TER—which are computed with respect to the independent references—are mixed. The lower En-De BLEU score is explained by a higher brevity penalty for the re-tuned output (0.918 vs. 0.862). However, the re-tuned 4-gram and 3-gram precisions are significantly higher. The unchanged Fr-En TER value can be explained by the observation that no human translators produced TER scores higher than the baseline MT. This odd result has also been observed for BLEU (Culy and Riehemann, 2003), although here we do observe a slight BLEU improvement.

System	HTER↓ int	System	HTER↓ pe
baseline	44.05	baseline	41.05
re-tune (int)	43.99	re-tune (pe)	**40.34**
re-tune+feat	**42.35**	–	–
Δ	−1.80		−0.71

Table 7: En-De test results for re-tuning to post-edit (pe) vs. interactive (int). Features cannot be extracted from the post-edit data, so the re-tune+feat system cannot be learned. The Fr-En results are similar but are omitted due to space.

The additional features (854 for Fr-En; 847 for En-De) help significantly and do not slow down decoding. We used the same L_1 regularization strength as the baseline, but feature growth could be further constrained by increasing this parameter. Tuning is very fast at about six minutes *for the whole dataset*, so tuning during a live user session is already practical.

Table 7 compares re-tuning to interactive vs. post-edit corrections. Recall that the int-test and pe-test datasets are different and contain different references. The post-edit baseline is lower because humans performed less editing in the baseline condition (see Table 1). Features account for the greatest reduction in HTER. Of course, the features are based mostly on word alignments, which could be obtained for the post-edit data by running an online word alignment tool (see: Farajian et al. (2014)). However, the interactive logs contain much richer user state information that we could not exploit due to data sparsity. We also hypothesize that the final interactive corrections might be more useful since suggestions prime translators (Green et al., 2013a), and the MT system was able to refine its suggestions.

6 Re-tuning Analysis

Tables 6 and 7 raise two natural questions: what accounts for the reduction in HTER, and why are the TER/BLEU results mixed? Comparison of the BLEU-tuned baseline to the HTER re-tuned systems gives some insight. For both questions, fine-

grained corrections appear to make the difference.

Consider this French test example (with gloss):

(1) une ligne de chimiothérapie antérieure
one line of chemotherapy previous

The independent reference for *une ligne de chimiothérapie* is 'previous chemotherapy treatment', and the baseline produces 'previous chemotherapy line.' The source sentence appears seven times with the following user translations: 'one line or more of chemotherapy', 'one prior line of chemotherapy', 'one previous line of chemotherapy' (2), 'one line of chemotherapy before' (2), 'one protocol of chemotherapy'. The re-tuned, feature-based system produces 'one line of chemotherapy before', matching two of the humans exactly, and six of the humans in terms of idiomatic medical jargon ('line of chemotherapy' vs. 'chemotherapy treatment'). However, the baseline output would have received better BLEU and TER scores.

Sometimes re-tuning improves the translations with respect to both the reference and the human corrections. This English phrase appears in the En-De test set:

(2) depending on the file
abhängig von der datei

The baseline produces exactly the gloss shown in Ex. (2). The human translators produced: 'je nach datei' (6), 'das dokument', and 'abhängig von der datei'. The re-tuned system rendered the phrase 'je nach dokument', which is closer to both the independent reference 'je nach datei' and the human corrections. This change improves TER, BLEU, and HTER.

7 Related Work

The process study most similar to ours is that of Koehn (2009a), who compared scratch, post-edit, and simple interactive modes. However, he used undergraduate, non-professional subjects, and did not consider re-tuning. Our experimental design with professional bilingual translators follows our previous work Green et al. (2013a) comparing scratch translation to post-edit.

Many research translation UIs have been proposed including TransType (Langlais et al., 2000), Caitra (Koehn, 2009b), Thot (Ortiz-Martínez and Casacuberta, 2014), TransCenter (Denkowski et al., 2014b), and CasmaCat (Alabau et al., 2013). However, to our knowledge, none of these interfaces were explicitly designed according to mixed-initiative principles from the HCI literature.

Incremental MT learning has been investigated several times, usually starting from no data (Barrachina et al., 2009; Ortiz-Martínez et al., 2010), via simulated post-editing (Martínez-Gómez et al., 2012; Denkowski et al., 2014a), or via re-ranking (Wäschle et al., 2013). No previous experiments combined large-scale baselines, full re-tuning of the model weights, and HTER optimization.

HTER tuning can be simulated by re-parameterizing an existing metric. Snover et al. (2009) tuned TERp to correlate with HTER, while Denkowski and Lavie (2010) did the same for METEOR. Zaidan and Callison-Burch (2010) showed how to solicit MT corrections for HTER from Amazon Mechanical Turk.

Our learning approach is related to coactive learning (Shivaswamy and Joachims, 2012). Their basic preference perceptron updates toward a correction, whereas we use the correction for metric scoring and feature extraction.

8 Conclusion

We presented a new CAT interface that supports post-edit and interactive modes. Evaluation with professional, bilingual translators showed post-edit to be faster, but prior subject familiarity with post-edit may have mattered. For French-English, the interactive mode enabled higher quality translation. Re-tuning the MT system to interactive corrections also yielded large HTER gains. Technical contributions that make re-tuning possible are a cross-entropy objective, prefix decoding, and dynamic phrase table augmentation. Larger quantities of corrections should yield further gains, but our current experiments already establish the feasibility of Bar-Hillel's virtuous "machine-post-editor partnership" which benefits both humans and machines.

Acknowledgements

We thank TAUS for access to their data repository. We also thank John DeNero, Chris Dyer, Alon Lavie, and Matt Post for helpful conversations. The first author is supported by a National Science Foundation Graduate Research Fellowship. This work was also supported by the Defense Advanced Research Projects Agency (DARPA) Broad Operational Language Translation (BOLT) program through IBM. Any opinions, findings, and conclusions or recommendations expressed are those of the author(s) and do not necessarily reflect the view of either DARPA or the US government.

References

V. Alabau, R. Bonk, C. Buck, M. Carl, F. Casacuberta, M. García-Martínez, et al. 2013. Advanced computer aided translation with a web-based workbench. In *2nd Workshop on Post-Editing Technologies and Practice*.

R.H. Baayen, D.J. Davidson, and D.M. Bates. 2008. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59(4):390–412.

Y. Bar-Hillel. 1960. The present status of automatic translation of languages. *Advances in Computers*, 1:91–163.

D. J. Barr, R. Levy, C. Scheepers, and H. J. Tily. 2013. Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, 68(3):255–278.

S. Barrachina, O. Bender, F. Casacuberta, J. Civera, E. Cubel, S. Khadivi, et al. 2009. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28.

D. M. Bates. 2007. `lme4`: Linear mixed-effects models using S4 classes. Technical report, R package version 1.1-5, `http://cran.r-project.org/package=lme4`.

O. Bojar, C. Buck, C. Callison-Burch, C. Federmann, B. Haddow, P. Koehn, et al. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *WMT*.

O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, et al. 2014. Findings of the 2014 Workshop on Statistical Machine Translation. In *WMT*.

C. Callison-Burch, M. Osborne, and P. Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In *EACL*.

M. Carl. 2010. A computational framework for a cognitive model of human translation processes. In *Aslib Translating and the Computer Conference*.

D. Cer, C. D. Manning, and D. Jurafsky. 2010. The best lexical metric for phrase-based statistical MT system optimization. In *NAACL*.

C. Cherry and G. Foster. 2012. Batch tuning strategies for statistical machine translation. In *NAACL*.

C. Culy and S. Z. Riehemann. 2003. The limits of n-gram translation evaluation metrics. In *MT Summit IX*.

M. Denkowski and A. Lavie. 2010. Extending the METEOR machine translation evaluation metric to the phrase level. In *NAACL*.

M. Denkowski, C. Dyer, and A. Lavie. 2014a. Learning from post-editing: Online model adaptation for statistical machine translation. In *EACL*.

M. Denkowski, A. Lavie, I. Lacruz, and C. Dyer. 2014b. Real time adaptive machine translation for post-editing with cdec and TransCenter. In *Workshop on Humans and Computer-assisted Translation*.

J. Duchi, E. Hazan, and Y. Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

M. A. Farajian, N. Bertoldi, and M. Federico. 2014. Online word alignment for online adaptive machine translation. In *Workshop on Humans and Computer-assisted Translation*.

C. Federmann. 2010. Appraise: An open-source toolkit for manual phrase-based evaluation of translations. In *LREC*.

G. Foster, P. Langlais, and G. Lapalme. 2002. User-friendly text prediction for translators. In *EMNLP*.

M. Galley and C. D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *EMNLP*.

S. Green, J. Heer, and C. D. Manning. 2013a. The efficacy of human post-editing for language translation. In *CHI*.

S. Green, S. Wang, D. Cer, and C. D. Manning. 2013b. Fast and adaptive online training of feature-rich translation models. In *ACL*.

S. Green, D. Cer, and C. D. Manning. 2014a. An empirical comparison of features and tuning for phrase-based machine translation. In *WMT*.

S. Green, D. Cer, and C. D. Manning. 2014b. Phrasal: A toolkit for new directions in statistical machine translation. In *WMT*.

S. Green, J. Chuang, J. Heer, and C. D. Manning. 2014c. Predictive Translation Memory: A mixed-initiative system for human language translation. In *UIST*.

K. Heafield, I. Pouzyrevsky, J. H. Clark, and P. Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *ACL, Short Papers*.

L. Huang and D. Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *ACL*.

R. Hwa, P. Resnik, A. Weinberg, and O. Kolak. 2002. Evaluating translational correspondence using annotation projection. In *ACL*.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL, Demonstration Session*.

P. Koehn. 2009a. A process study of computer-aided translation. *Machine Translation*, 23:241–263.

P. Koehn. 2009b. A web-based interactive computer aided translation tool. In *ACL-IJCNLP, Software Demonstrations*.

P. Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press.

P. Langlais, G. Foster, and G. Lapalme. 2000. TransType: a computer-aided translation typing system. In *Workshop on Embedded Machine Translation Systems*.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *NAACL*.

C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL, System Demonstrations*.

P. Martínez-Gómez, G. Sanchis-Trilles, and F. Casacuberta. 2012. Online adaptation strategies for statistical machine translation in post-editing scenarios. *Pattern Recognition*, 45(9):3193–3203.

F. J. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

F. J. Och, R. Zens, and H. Ney. 2003. Efficient search for interactive statistical machine translation. In *EACL*.

F. J. Och. 2003. Minimum error rate training for statistical machine translation. In *ACL*.

D. Ortiz-Martínez and F. Casacuberta. 2014. The new Thot toolkit for fully automatic and interactive statistical machine translation. In *EACL, System Demonstrations*.

D. Ortiz-Martínez, I. García-Varea, and F. Casacuberta. 2009. Interactive machine translation based on partial statistical phrase-based alignments. In *RANLP*.

D. Ortiz-Martínez, I. García-Varea, and F. Casacuberta. 2010. Online learning for interactive statistical machine translation. In *NAACL*.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

R. Ray. 2013. Ten essential research findings for 2013. In *2013 Resource Directory & Index*. Multilingual.

S. Riezler and J. T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing in MT. In *ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.

K. Sakaguchi, M. Post, and B. Van Durme. 2014. Efficient elicitation of annotations for human evaluation of machine translation. In *WMT*.

P. Shivaswamy and T. Joachims. 2012. Online structured prediction via coactive learning. In *ICML*.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *AMTA*.

M. Snover, N. Madnani, B. Dorr, and R. Schwartz. 2009. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *WMT*.

K. Wäschle, P. Simianer, N. Bertoldi, S. Riezler, and M. Federico. 2013. Generative and discriminative methods for online adaptation in SMT. In *MT Summit XIV*.

O. F. Zaidan and C. Callison-Burch. 2010. Predicting human-targeted translation edit rate via untrained human annotators. In *NAACL*.

Exact Decoding for Phrase-Based Statistical Machine Translation

Wilker Aziz[†] **Marc Dymetman**[‡] **Lucia Specia**[†]

[†]Department of Computer Science, University of Sheffield, UK
W.Aziz@sheffield.ac.uk
L.Specia@sheffield.ac.uk
[‡]Xerox Research Centre Europe, Grenoble, France
Marc.Dymetman@xrce.xerox.com

Abstract

The combinatorial space of translation derivations in phrase-based statistical machine translation is given by the intersection between a translation lattice and a target language model. We replace this intractable intersection by a tractable relaxation which incorporates a low-order upperbound on the language model. Exact optimisation is achieved through a coarse-to-fine strategy with connections to adaptive rejection sampling. We perform exact optimisation with unpruned language models of order 3 to 5 and show search-error curves for beam search and cube pruning on standard test sets. This is the first work to tractably tackle exact optimisation with language models of orders higher than 3.

1 Introduction

In Statistical Machine Translation (SMT), the task of producing a translation for an input string $\mathbf{x} = \langle x_1, x_2, \ldots, x_I \rangle$ is typically associated with finding the best derivation $\mathbf{d}^*$ compatible with the input under a linear model. In this view, a derivation is a structured output that represents a sequence of steps that covers the input producing a translation. Equation 1 illustrates this *decoding* process.

$$\mathbf{d}^* = \underset{\mathbf{d} \in \mathcal{D}(\mathbf{x})}{\operatorname{argmax}} f(\mathbf{d}) \tag{1}$$

The set $\mathcal{D}(\mathbf{x})$ is the space of all derivations compatible with $\mathbf{x}$ and supported by a model of translational equivalences (Lopez, 2008). The function $f(\mathbf{d}) = \Lambda \cdot \mathbf{H}(\mathbf{d})$ is a linear parameterisation of the model (Och, 2003). It assigns a real-valued score (or weight) to every derivation $\mathbf{d} \in \mathcal{D}(\mathbf{x})$, where $\Lambda \in \mathbb{R}^m$ assigns a relative importance to different aspects of the derivation independently captured by m feature functions $\mathbf{H}(\mathbf{d}) = \langle H_1(\mathbf{d}), \ldots, H_m(\mathbf{d}) \rangle \in \mathbb{R}^m$.

The fully parameterised model can be seen as a discrete weighted set such that feature functions factorise over the steps in a derivation. That is, $H_k(\mathbf{d}) = \sum_{e \in \mathbf{d}} h_k(e)$, where h_k is a (local) feature function that assesses steps independently and $\mathbf{d} = \langle e_1, e_2, \ldots, e_l \rangle$ is a sequence of l steps. Under this assumption, each step is assigned the weight $w(e) = \Lambda \cdot \langle h_1(e), h_2(e), \ldots, h_m(e) \rangle$. The set $\mathcal{D}$ is typically finite, however, it contains a very large number of structures — exponential (or even factorial, see §2) with the size of $\mathbf{x}$ — making exhaustive enumeration prohibitively slow. Only in very restricted cases combinatorial optimisation techniques are directly applicable (Tillmann et al., 1997; Och et al., 2001), thus it is common to resort to heuristic techniques in order to find an approximation to $\mathbf{d}^*$ (Koehn et al., 2003; Chiang, 2007).

Evaluation exercises indicate that approximate search algorithms work well in practice (Bojar et al., 2013). The most popular algorithms provide solutions with unbounded error, thus precisely quantifying their performance requires the development of a tractable exact decoder. To date, most attempts were limited to short sentences and/or somewhat toy models trained with artificially small datasets (Germann et al., 2001; Iglesias et al., 2009; Aziz et al., 2013). Other work has employed less common approximations to the model reducing its search space complexity (Kumar et al., 2006; Chang and Collins, 2011; Rush and Collins, 2011). These do not answer whether or not current decoding algorithms perform well at real translation tasks with state-of-the-art models.

We propose an exact decoder for phrase-based SMT based on a coarse-to-fine search strategy (Dymetman et al., 2012). In a nutshell, we relax the decoding problem with respect to the Language Model (LM) component. This coarse view is incrementally refined based on evidence col-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1237–1249,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

lected via maximisation. A refinement increases the complexity of the model only slightly, hence dynamic programming remains feasible throughout the search until convergence. We test our decoding strategy with realistic models using standard data sets. We also contribute with optimum derivations which can be used to assess future improvements to approximate decoders. In the remaining sections we present the general model (§2), survey contributions to exact optimisation (§3), formalise our novel approach (§4), present experiments (§5) and conclude (§6).

2 Phrase-based SMT

In phrase-based SMT (Koehn et al., 2003), the building blocks of translation are pairs of phrases (or *biphrases*). A translation derivation $\mathbf{d}$ is an ordered sequence of *non-overlapping* biphrases which covers the input text in arbitrary order generating the output from left to right.[1]

$$f(\mathbf{d}) = \psi(\mathbf{y}) + \sum_{i=1}^{l} \phi(e_i) + \sum_{i=1}^{l-1} \delta(e_i, e_{i-1}) \quad (2)$$

Equation 2 illustrates a standard phrase-based model (Koehn et al., 2003): ψ is a weighted target n-gram LM component, where $\mathbf{y}$ is the yield of $\mathbf{d}$; ϕ is a linear combination of features that decompose over phrase pairs directly (e.g. backward and forward translation probabilities, lexical smoothing, and word and phrase penalties); and δ is an unlexicalised penalty on the number of skipped input words between two adjacent biphrases. The weighted logic program in Figure 1 specifies the fully parameterised weighted set of solutions, which we denote $\langle \mathcal{D}(\mathbf{x}), f(\mathbf{d}) \rangle$.[2]

A weighted logic program starts from its axioms and follows exhaustively deducing new items by combination of existing ones and no deduction happens twice. In Figure 1, a *nonteminal* item summarises partial derivation (or *hypotheses*). It is denoted by $[C, r, \gamma]$ (also known as *carry*), where: C is a coverage vector, necessary to impose the non-overlapping constraint; r is the rightmost position most recently covered, necessary for the computation of δ; and γ is the last $n-1$ words

$$\begin{array}{ll}
\textsc{Item} & \left[\{0,1\}^I, [0, I+1], \Delta^{n-1}\right] \\
\textsc{Goal} & \left[1^I, I+1, \textsc{eos}\right] \\
\textsc{Axiom} & \\
& \dfrac{\langle \textsc{bos} \rightarrow \textsc{bos} \rangle}{[0^I, 0, \textsc{bos}] : \psi(\textsc{bos})} \\
\textsc{Expand} & \\
& \dfrac{\left[C, r, y_{j-n+1}^{j-1}\right] \quad \left\langle x_i^{i'} \xrightarrow{\phi_r} y_j^{j'} \right\rangle}{\left[C', i', y_{j'-n+2}^{j'}\right] : w} \quad \bigoplus_{k=i}^{i'} c_k = \bar{0} \\
& \text{where} \quad c'_k = c_k \text{ if } k < i \text{ or } k > i' \text{ else } \bar{1} \\
& \qquad\quad w = \phi_r \otimes \delta(r, i) \otimes \psi(y_j^{j'} | y_{j-n+1}^{j-1}) \\
\textsc{Accept} & \\
& \dfrac{\left[1^I, r, \gamma\right]}{[1^I, I+1, \textsc{eos}] : \delta(r, I+1) \otimes \psi(\textsc{eos}|\gamma)} \quad r \leq I
\end{array}$$

Figure 1: Specification for the weighted set of translation derivations in phrase-based SMT with unconstrained reordering.

in the yield, necessary for the LM component. The program expands partial derivations by concatenation with a translation rule $\left\langle x_i^{i'} \xrightarrow{\phi_r} y_j^{j'} \right\rangle$, that is, an instantiated biphrase which covers the span $x_i^{i'}$ and yields $y_j^{j'}$ with weight ϕ_r. The side condition imposes the non-overlapping constraint (c_k is the kth bit in C). The antecedents are used to compute the weight of the deduction, and the carry is updated in the consequent (item below the horizontal line). Finally, the rule ACCEPT incorporates the end-of-sentence boundary to complete items.[3]

It is perhaps illustrative to understand the set of weighted translation derivations as the intersection between two components. One that is only locally parameterised and contains all translation derivations (a translation *lattice* or *forest*), and one that re-ranks the first as a function of the interactions between translation steps. The model of translational equivalences parameterised only with ϕ is an instance of the former. An n-gram LM component is an instance of the latter.

2.1 Hypergraphs

A *backward-hypergraph*, or simply *hypergraph*, is a generalisation of a graph where edges have multiple origins and one destination (Gallo et al., 1993). They can represent both finite-state and context-free weighted sets and they have been widely used in SMT (Huang and Chiang, 2007). A hypergraph is defined by a set of nodes (or ver-

[1] Preventing phrases from overlapping requires an exponential number of constraints (the powerset of $\mathbf{x}$) rendering the problem NP-complete (Knight, 1999).

[2] Weighted logics have been extensively used to describe weighted sets (Lopez, 2009), operations over weighted sets (Chiang, 2007; Dyer and Resnik, 2010), and a variety of dynamic programming algorithms (Cohen et al., 2008).

[3] Figure 1 can be seen as a specification for a weighted acyclic finite-state automaton whose states are indexed by $[l, C, r]$ and transitions are labelled with biphrases. However, for generality of representation, we opt for using acyclic hypergraphs instead of automata (see §2.1).

tices) V and a weighted set of edges $\langle E, w \rangle$. An edge e connects a sequence of nodes in its tail $\mathrm{t}[e] \in V^*$ under a head node $\mathrm{h}[e] \in V$ and has weight $w(e)$. A node v is a *terminal node* if it has no incoming edges, otherwise it is a *nonterminal node*. The node that has no outgoing edges, is called *root*, with no loss of generality we can assume hypergraphs to have a single root node.

Hypergraphs can be seen as instantiated logic programs. In this view, an item is a template for the creation of nodes, and a weighted deduction rule is a template for edges. The tail of an edge is the sequence of nodes associated with the antecedents, and the head is the node associated with the consequent. Even though the space of weighted derivations in phrase-based SMT is finite-state, using a hypergraph as opposed to a finite-state automaton makes it natural to encode multi-word phrases using tails. We opt for representing the target side of the biphrase as a sequence of terminals nodes, each of which represents a target word.

3 Related Work

3.1 Beam filling algorithms

Beam search (Koehn et al., 2003) and cube pruning (Chiang, 2007) are examples of state-of-the-art approximate search algorithms. They approximate the intersection between the translation forest and the language model by expanding a limited beam of hypotheses from each nonterminal node. Hypotheses are organised in priority queues according to common traits and a fast-to-compute heuristic view of *outside weights* (cheapest way to complete a hypothesis) puts them to compete at a fairer level. Beam search exhausts a node's possible expansions, scores them, and discards all but the k highest-scoring ones. This process is wasteful in that k is typically much smaller than the number of possible expansions. Cube pruning employs a priority queue at beam filling and computes k high-scoring expansions directly in near best-first order. The parameter k is known as *beam size* and it controls the time-accuracy trade-off of the algorithm.

Heafield et al. (2013a) move away from using the language model as a black-box and build a more involved beam filling algorithm. Even though they target approximate search, some of their ideas have interesting connections to ours (see §4). They group hypotheses that share partial language model state (Li and Khudanpur, 2008) reasoning over multiple hypotheses at once. They fill a beam in best-first order by iteratively visiting groups using a priority queue: if the top group contains a single hypothesis, the hypothesis is added to the beam, otherwise the group is partitioned and the parts are pushed back to the queue. More recently, Heafield et al. (2014) applied their beam filling algorithm to phrase-based decoding.

3.2 Exact optimisation

Exact optimisation for monotone translation has been done using A* search (Tillmann et al., 1997) and finite-state operations (Kumar et al., 2006). Och et al. (2001) design near-admissible heuristics for A* and decode very short sentences (6-14 words) for a word-based model (Brown et al., 1993) with a maximum distortion strategy ($d = 3$).

Zaslavskiy et al. (2009) frame phrase-based decoding as an instance of a generalised Travelling Salesman Problem (TSP) and rely on robust solvers to perform decoding. In this view, a *salesman graph* encodes the translation options, with each node representing a biphrase. Non-overlapping constraints are imposed by the TSP solver, rather than encoded directly in the salesman graph. They decode only short sentences (17 words on average) using a 2-gram LM due to salesman graphs growing too large.[4]

Chang and Collins (2011) relax phrase-based models w.r.t. the non-overlapping constraints, which are replaced by soft penalties through Lagrangian multipliers, and intersect the LM component exhaustively. They do employ a maximum distortion limit ($d = 4$), thus the problem they tackle is no longer NP-complete. Rush and Collins (2011) relax a hierarchical phrase-based model (Chiang, 2005)[5] w.r.t. the LM component. The translation forest and the language model trade their weights (through Lagrangian multipliers) so as to ensure agreement on what each component believes to be the maximum. In both approaches, when the dual converges to a compliant solution, the solution is guaranteed to be optimal. Other-

[4] Exact decoding had been similarly addressed with Integer Linear Programming (ILP) in the context of word-based models for very short sentences using a 2-gram LM (Germann et al., 2001). Riedel and Clarke (2009) revisit that formulation and employ a cutting-plane algorithm (Dantzig et al., 1954) reaching 30 words.

[5] In hierarchical translation, reordering is governed by a synchronous context-free grammar and the underlying problem is no longer NP-complete. Exact decoding remains infeasible because the intersection between the translation forest and the target LM is prohibitively slow.

wise, a subset of the constraints is explicitly added and the dual optimisation is repeated. They handle sentences above average length, however, resorting to compact rulesets (10 translation options per input segment) and using only 3-gram LMs.

In the context of hierarchical models, Aziz et al. (2013) work with unpruned forests using upperbounds. Their approach is the closest to ours. They also employ a coarse-to-fine strategy with the OS* framework (Dymetman et al., 2012), and investigate unbiased sampling in addition to optimisation. However, they start from a coarser upperbound with unigram probabilities, and their refinement strategies are based on exhaustive intersections with small n-gram matching automata. These refinements make forests grow unmanageable too quickly. Because of that, they only deal with very short sentences (up to 10 words) and even then decoding is very slow. We design better upperbounds and a more efficient refinement strategy. Moreover, we decode long sentences using language models of order 3 to 5.[6]

4 Approach

4.1 Exact optimisation with OS*

Dymetman et al. (2012) introduced OS*, a unified view of optimisation and sampling which can be seen as a cross between adaptive rejection sampling (Robert and Casella, 2004) and A* optimisation (Hart et al., 1968). In this framework, a complex goal distribution is upperbounded by a simpler proposal distribution for which optimisation (and sampling) is feasible. This proposal is incrementally refined to be closer to the goal until the maximum is found (or until the sampling performance exceeds a certain level).

Figure 2 illustrates exact optimisation with OS*. Suppose f is a complex target goal distribution, such that we cannot optimise f, but we can assess $f(\mathbf{d})$ for a given $\mathbf{d}$. Let $g^{(0)}$ be an upperbound to f, i.e., $g^{(0)}(\mathbf{d}) \geq f(\mathbf{d})$ for all $\mathbf{d} \in \mathcal{D}(\mathbf{x})$. Moreover, suppose that $g^{(0)}$ is simple enough to be optimised efficiently. The algorithm proceeds by solving $\mathbf{d}_0 = \text{argmax}_{\mathbf{d}}\, g^{(0)}(\mathbf{d})$ and computing the quantity $r_0 = f(\mathbf{d}_0)/g^{(0)}(\mathbf{d}_0)$. If r_0 were sufficiently close to 1, then $g^{(0)}(\mathbf{d}_0)$ would be sufficiently close to $f(\mathbf{d}_0)$ and we would have found the optimum. However, in the illustration $g^{(0)}(\mathbf{d}_0) \gg f(\mathbf{d}_0)$, thus $r_0 \ll 1$. At this point the algorithm has concrete evidence to motivate a refinement of $g^{(0)}$ that can lower its maximum, bringing it closer to $f^* = \max_{\mathbf{d}} f(\mathbf{d})$ at the cost of some small increase in complexity. The refined proposal must remain an upperbound to f. To continue with the illustration, suppose $g^{(1)}$ is obtained. The process is repeated until eventually $g^{(t)}(\mathbf{d}_t) = f(\mathbf{d}_t)$, where $\mathbf{d}_t = \text{argmax}_{\mathbf{d}}\, g^{(t)}(\mathbf{d})$, for some finite t. At which point $\mathbf{d}_t$ is the optimum derivation $\mathbf{d}^*$ from f and the sequence of upperbounds provides a proof of optimality.[7]

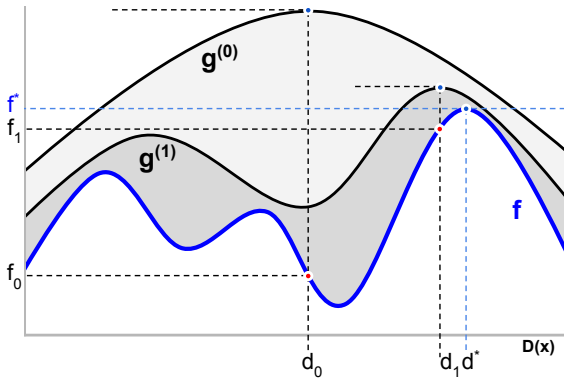


Figure 2: Sequence of incrementally refined upperbound proposals.

4.2 Model

We work with phrase-based models in a standard parameterisation (Equation 2). However, to avoid having to deal with NP-completeness, we constrain reordering to happen only within a limited window given by a notion of *distortion limit*. We require that the last source word covered by any biphrase must be within d words from the leftmost uncovered source position (Lopez, 2009). This is a widely used strategy and it is in use in the Moses toolkit (Koehn et al., 2007).[8]

Nevertheless, the problem of finding the best

[6]The intuition that a full intersection is wasteful is also present in (Petrov et al., 2008) in the context of approximate search. They start from a coarse distribution based on automatic word clustering which is refined in multiple passes. At each pass, hypotheses are pruned a posteriori on the basis of their marginal probabilities, and word clusters are further split. We work with upperbounds, rather than word clusters, with unpruned distributions, and perform exact optimisation.

[7]If $\mathbf{d}$ is a maximum from g and $g(\mathbf{d}) = f(\mathbf{d})$, then it is easy to show by contradiction that $\mathbf{d}$ is the actual maximum from f: if there existed $\mathbf{d}'$ such that $f(\mathbf{d}') > f(\mathbf{d})$, then it follows that $g(\mathbf{d}') \geq f(\mathbf{d}') > f(\mathbf{d}) = g(\mathbf{d})$, and hence $\mathbf{d}$ would not be a maximum for g.

[8]A distortion limit characterises a form of pruning that acts directly in the generative capacity of the model leading to induction errors (Auli et al., 2009). Limiting reordering like that lowers complexity to a polynomial function of I and an exponential function of the distortion limit.

derivation under the model remains impracticable due to nonlocal parameterisation (namely, the n-gram LM component). The weighted set $\langle \mathcal{D}(\mathbf{x}), f(\mathbf{d}) \rangle$, which represents the objective, is a complex hypergraph which we cannot afford to construct. We propose to construct instead a simpler hypergraph for which optimisation by dynamic programming is feasible. This *proxy* represents the weighted set $\langle \mathcal{D}(\mathbf{x}), g^{(0)}(\mathbf{d}) \rangle$, where $g^{(0)}(\mathbf{d}) \geq f(\mathbf{d})$ for every $\mathbf{d} \in \mathcal{D}(\mathbf{x})$. Note that this proposal contains **exactly** the same translation options as in the original decoding problem. The simplification happens only with respect to the parameterisation. Instead of intersecting the complete n-gram LM distribution explicitly, we implicitly intersect a simpler upperbound view of it, where by *simpler* we mean *lower-order*.

$$g^{(0)}(\mathbf{d}) = \sum_{i=1}^{l} \omega(\mathrm{y}[e_i]) + \sum_{i=1}^{l} \phi(e_i) + \sum_{i=1}^{l-1} \delta(e_i, e_{i-1}) \quad (3)$$

Equation 3 shows the model we use as a proxy to perform exact optimisation over f. In comparison to Equation 2, the term $\sum_{i=1}^{l} \omega(\mathrm{y}[e_i])$ replaces $\psi(\mathbf{y}) = \lambda_\psi p_{\text{LM}}(\mathbf{y})$. While ψ weights the yield $\mathbf{y}$ taking into account all n-grams (including those crossing the boundaries of phrases), ω weights edges in isolation. Particularly, $\omega(\mathrm{y}[e_i]) = \lambda_\psi q_{\text{LM}}(\mathrm{y}[e_i])$, where $\mathrm{y}[e_i]$ returns the sequence of target words (a target phrase) associated with the edge, and $q_{\text{LM}}(\cdot)$ is an upperbound on the true LM probability $p_{\text{LM}}(\cdot)$ (see §4.3). It is obvious from Equation 3 that our proxy model is much simpler than the original — the only form of nonlocal parameterisation left is the distortion penalty, which is simple enough to represent exactly.

The program in Figure 3 illustrates the construction of $\langle \mathcal{D}(\mathbf{x}), g^{(0)}(\mathbf{d}) \rangle$. A nonterminal item $[l, C, r]$ stores: the leftmost uncovered position l and a truncated coverage vector C (together they track d input positions); and the rightmost position r most recently translated (necessary for the computation of the distortion penalty). Observe how nonterminal items **do not** store the LM state.[9] The rule ADJACENT expands derivations by concatenation with a biphrase $\left\langle x_i^{i'} \rightarrow y_j^{j'} \right\rangle$ starting at the leftmost uncovered position $i = l$. That causes the coverage window to move ahead to the next leftmost uncovered position: $l' = l + \alpha_1(C) + 1$,

[9] Drawing a parallel to (Heafield et al., 2013a), a nonterminal node in our hypergraph groups derivations while exposing only an empty LM state.

ITEM $\left[[1, I+1], \{0,1\}^{d-1}, [0, I+1]\right]$
GOAL $[I, \emptyset, I+1]$
AXIOMS

$$\frac{\langle \text{BOS} \rightarrow \text{BOS} \rangle}{[1, 0^{d-1}, 0] : \omega(\text{BOS})}$$

ADJACENT

$$\frac{[l, C, r] \quad \left\langle x_i^{i'} \xrightarrow{\phi_r} y_j^{j'} \right\rangle}{[l', C', i'] : \phi_r \otimes \delta(r, i') \otimes \omega(y_j^{j'})} \quad \begin{array}{l} i = l \\ \bigoplus_{k=i-l}^{i'-l} c_k = \bar{0} \end{array}$$

where $l' = l + \alpha_1(C) + 1$
$C' \ll \alpha_1(C) + 1$

NON-ADJACENT

$$\frac{[l, C, r] \quad \left\langle x_i^{i'} \xrightarrow{\phi_r} y_j^{j'} \right\rangle}{[l, C', i'] : \phi_r \otimes \delta(r, i') \otimes \omega(y_j^{j'})} \quad \begin{array}{l} i > l \\ \bigoplus_{k=i-l}^{i'-l} c_k = \bar{0} \\ |r - i + 1| \leq d \\ |i' - l + 1| \leq d \end{array}$$

where $c'_k = c_k$ if $k < i - l$ or $k > i' - l$ else $\bar{1}$

ACCEPT

$$\frac{[I+1, C, r]}{[I+1, \emptyset, I+1] : \delta(r, I+1) \otimes \omega(\text{EOS})} \quad r \leq I$$

Figure 3: Specification of the initial proposal hypergraph. This program allows the same reorderings as (Lopez, 2009) (see logic WLd), however, it does not store LM state information and it uses the upperbound LM distribution $\omega(\cdot)$.

where $\alpha_1(C)$ returns the number of leading 1s in C, and $C' \ll \alpha_1(C) + 1$ represents a left-shift. The rule NON-ADJACENT handles the remaining cases $i > l$ provided that the expansion skips at most d input words $|r - i + 1| \leq d$. In the consequent, the window C is simply updated to record the translation of the input span $i..i'$. In the non-adjacent case, a *gap constraint* imposes that the resulting item will require skipping no more than d positions before the leftmost uncovered word is translated $|i' - l + 1| \leq d$.[10] Finally, note that deductions incorporate the weighted upperbound $\omega(\cdot)$, rather than the true LM component $\psi(\cdot)$.[11]

4.3 LM upperbound and Max-ARPA

Following Carter et al. (2012) we compute an upperbound on n-gram conditional probabilities by precomputing max-backoff weights stored in a "Max-ARPA" table, an extension of the ARPA format (Jurafsky and Martin, 2000).

A standard ARPA table T stores entries

[10] This constraint prevents items from becoming dead-ends where incomplete derivations require a reordering step larger than d. This is known to prevent many search errors in beam search (Chang and Collins, 2011).

[11] Unlike Aziz et al. (2013), rather than unigrams only, we score all n-grams within a translation rule (including incomplete ones).

$\langle \mathtt{Z}, \mathtt{Z.p}, \mathtt{Z.b} \rangle$, where $\mathtt{Z}$ is an n-gram equal to the concatenation $\mathtt{Pz}$ of a prefix $\mathtt{P}$ with a word $\mathtt{z}$, $\mathtt{Z.p}$ is the conditional probability $p(\mathtt{z}|\mathtt{P})$, and $\mathtt{Z.b}$ is a so-called "backoff" weight associated with $\mathtt{Z}$. The conditional probability of an arbitrary n-gram $p(\mathtt{z}|\mathtt{P})$, whether listed or not, can then be recovered from T by the simple recursive procedure shown in Equation 4, where tail deletes the first word of the string $\mathtt{P}$.

$$p(\mathtt{z}|\mathtt{P}) = \begin{cases} p(\mathtt{z}|\,\mathrm{tail}(\mathtt{P})) & \mathtt{Pz} \notin T \text{ and } \mathtt{P} \notin T \\ p(\mathtt{z}|\,\mathrm{tail}(\mathtt{P})) \times \mathtt{P.b} & \mathtt{Pz} \notin T \text{ and } \mathtt{P} \in T \\ \mathtt{Pz.p} & \mathtt{Pz} \in T \end{cases} \tag{4}$$

The optimistic version (or "max-backoff") q of p is defined as $q(\mathtt{z}|\mathtt{P}) \equiv \max_H p(\mathtt{z}|\mathtt{HP})$, where $\mathtt{H}$ varies over all possible contexts extending the prefix $\mathtt{P}$ to the left. The Max-ARPA table allows to compute $q(\mathtt{z}|\mathtt{P})$ for arbitrary values of $\mathtt{z}$ and $\mathtt{P}$. It is constructed on the basis of the ARPA table T by adding two columns to T: a column $\mathtt{Z.q}$ that stores the value $q(\mathtt{z}|\mathtt{P})$ and a column $\mathtt{Z.m}$ that stores an optimistic version of the backoff weight.

These columns are computed offline in two passes by first sorting T in descending order of n-gram length.[12] In the first pass (Algorithm 1), we compute for every entry in the table an optimistic backoff weight $\mathtt{m}$. In the second pass (Algorithm 2), we compute for every entry an optimistic conditional probability $\mathtt{q}$ by maximising over 1-word history extensions (whose $\mathtt{.q}$ fields are already known due to the sorting of T).

The following **Theorem** holds (see proof below): *For an arbitrary n-gram $Z = Pz$, the probability $q(z|P)$ can be recovered through the procedure shown in Equation 5.*

$$q(\mathtt{z}|\mathtt{P}) = \begin{cases} p(\mathtt{z}|\mathtt{P}) & \mathtt{Pz} \notin T \text{ and } \mathtt{P} \notin T \\ p(\mathtt{z}|\mathtt{P}) \times \mathtt{P.m} & \mathtt{Pz} \notin T \text{ and } \mathtt{P} \in T \\ \mathtt{Pz.q} & \mathtt{Pz} \in T \end{cases} \tag{5}$$

Note that, if $\mathtt{Z}$ is listed in the table, we return its upperbound probability $\mathtt{q}$ directly. When the n-gram is unknown, but its prefix is known, we take into account the optimistic backoff weight $\mathtt{m}$ of the prefix. On the other hand, if both the n-gram and its prefix are unknown, then no additional context could change the score of the n-gram, in which case $q(\mathtt{z}|\mathtt{P}) = p(\mathtt{z}|\mathtt{P})$.

In the sequel, we will need the following definitions. Suppose $\alpha = y_I^J$ is a substring of $\mathbf{y} = y_1^M$. Then $p_{\text{LM}}(\alpha) \equiv \prod_{k=I}^{J} p(y_k|y_1^{k-1})$ is the contribution of α to the true LM score of $\mathbf{y}$. We then obtain an upperbound $q_{\text{LM}}(\alpha)$ to this contribution by defining $q_{\text{LM}}(\alpha) \equiv q(y_I|\epsilon) \prod_{k=I+1}^{J} q(y_k|y_I^{k-1})$.

[12] If an n-gram is listed in T, then all its substrings must also be listed. Certain pruning strategies may corrupt this property, in which case we make missing substrings explicit.

Proof of **Theorem**. Let us first suppose that the length of $\mathtt{P}$ is strictly larger than the order n of the language model. Then for any $\mathtt{H}$, $p(\mathtt{z}|\mathtt{HP}) = p(\mathtt{z}|\mathtt{P})$; this is because $\mathtt{HP} \notin T$ and $\mathtt{P} \notin T$, along with all intermediary strings, hence, by (4), $p(\mathtt{z}|\mathtt{HP}) = p(\mathtt{z}|\,\mathrm{tail}(\mathtt{HP})) = p(\mathtt{z}|\,\mathrm{tail}(\mathrm{tail}(\mathtt{HP}))) = \ldots = p(\mathtt{z}|\mathtt{P})$. Hence $q(\mathtt{z}|\mathtt{P}) = p(\mathtt{z}|\mathtt{P})$, and, because $\mathtt{Pz} \notin T$ and $\mathtt{P} \notin T$, the theorem is satisfied in this case.

Having established the theorem for $|\mathtt{P}| > n$, we now assume that it is true for $|\mathtt{P}| > m$ and prove by induction that it is true for $|\mathtt{P}| = m$. We use the fact that, by the definition of $\mathtt{q}$, we have $q(\mathtt{z}|\mathtt{P}) = \max_{\mathtt{x}\in\Delta} q(\mathtt{z}|\mathtt{xP})$. We have three cases to consider.

First, suppose that $\mathtt{Pz} \notin T$ and $\mathtt{P} \notin T$. Then $\mathtt{xPz} \notin T$ and $\mathtt{xP} \notin T$, hence by induction $q(\mathtt{z}|\mathtt{xP}) = p(\mathtt{z}|\mathtt{xP}) = p(\mathtt{z}|\mathtt{P})$ for any $\mathtt{x}$, therefore $q(\mathtt{z}|\mathtt{P}) = p(\mathtt{z}|\mathtt{P})$. We have thus proven the first case.

Second, suppose that $\mathtt{Pz} \notin T$ and $\mathtt{P} \in T$. Then, for any $\mathtt{x}$, we have $\mathtt{xPz} \notin T$, and:

$$\begin{aligned} q(\mathtt{z}|\mathtt{P}) &= \max_{\mathtt{x}\in\Delta} q(\mathtt{z}|\mathtt{xP}) \\ &= \max\Big(\max_{\mathtt{x}\in\Delta,\, \mathtt{xP}\notin T} q(\mathtt{z}|\mathtt{xP}), \max_{\mathtt{x}\in\Delta,\, \mathtt{xP}\in T} q(\mathtt{z}|\mathtt{xP})\Big). \end{aligned}$$

For $\mathtt{xP} \notin T$, by induction, $q(\mathtt{z}|\mathtt{xP}) = p(\mathtt{z}|\mathtt{xP}) = p(\mathtt{z}|\mathtt{P})$, and therefore $\max_{\mathtt{x}\in\Delta,\, \mathtt{xP}\notin T} q(\mathtt{z}|\mathtt{xP}) = p(\mathtt{z}|\mathtt{P})$. For $\mathtt{xP} \in T$, we have $q(\mathtt{z}|\mathtt{xP}) = p(\mathtt{z}|\mathtt{xP}) \times \mathtt{xP.m} = p(\mathtt{z}|\mathtt{P}) \times \mathtt{xP.b} \times \mathtt{xP.m}$. Thus, we have:

$$\max_{\mathtt{x}\in\Delta,\, \mathtt{xP}\in T} q(\mathtt{z}|\mathtt{xP}) = p(\mathtt{z}|\mathtt{P}) \times \max_{\mathtt{x}\in\Delta,\, \mathtt{xP}\in T} \mathtt{xP.b} \times \mathtt{xP.m}.$$

But now, because of lines 3 and 4 of Algorithm 1, $\mathtt{P.m} = \max_{\mathtt{x}\in\Delta,\, \mathtt{xP}\in T} \mathtt{xP.b} \times \mathtt{xP.m}$, hence $\max_{\mathtt{x}\in\Delta,\, \mathtt{xP}\in T} q(\mathtt{z}|\mathtt{xP}) = p(\mathtt{z}|\mathtt{P}) \times \mathtt{P.m}$. Therefore, $q(\mathtt{z}|\mathtt{P}) = \max(p(\mathtt{z}|\mathtt{P}), p(\mathtt{z}|\mathtt{P}) \times \mathtt{P.m}) = p(\mathtt{z}|\mathtt{P}) \times \mathtt{P.m}$, where we have used the fact that $\mathtt{P.m} \geq 1$ due to line 1 of Algorithm 1. We have thus proven the second case.

Finally, suppose that $\mathtt{Pz} \in T$. Then, again,

$$\begin{aligned} q(\mathtt{z}|\mathtt{P}) &= \max_{\mathtt{x}\in\Delta} q(\mathtt{z}|\mathtt{xP}) \\ &= \max\Big(\\ &\qquad \max_{\mathtt{x}\in\Delta,\, \mathtt{xPz}\notin T,\, \mathtt{xP}\notin T} q(\mathtt{z}|\mathtt{xP}), \\ &\qquad \max_{\mathtt{x}\in\Delta,\, \mathtt{xPz}\notin T,\, \mathtt{xP}\in T} q(\mathtt{z}|\mathtt{xP}), \\ &\qquad \max_{\mathtt{x}\in\Delta,\, \mathtt{xPz}\in T} q(\mathtt{z}|\mathtt{xP})\ \Big). \end{aligned}$$

For $\mathtt{xPz} \notin T$, $\mathtt{xP} \notin T$, we have $q(\mathtt{z}|\mathtt{xP}) = p(\mathtt{z}|\mathtt{xP}) = p(\mathtt{z}|\mathtt{P}) = \mathtt{Pz.p}$, where the last equality is due to the fact that $\mathtt{Pz} \in T$. For $\mathtt{xPz} \notin T$, $\mathtt{xP} \in T$, we have $q(\mathtt{z}|\mathtt{xP}) = p(\mathtt{z}|\mathtt{xP}) \times \mathtt{xP.m} = p(\mathtt{z}|\mathtt{P}) \times \mathtt{xP.b} \times \mathtt{xP.m} = \mathtt{Pz.p} \times \mathtt{xP.b} \times \mathtt{xP.m}$. For $\mathtt{xPz} \in T$, we have $q(\mathtt{z}|\mathtt{xP}) = \mathtt{xPz.q}$. Overall, we thus have:

$$\begin{aligned} q(\mathtt{z}|\mathtt{P}) = \max\Big(& \mathtt{Pz.p}, \\ & \max_{\mathtt{x}\in\Delta,\, \mathtt{xPz}\notin T,\, \mathtt{xP}\in T} \mathtt{Pz.p} \times \mathtt{xP.b} \times \mathtt{xP.m}, \\ & \max_{\mathtt{x}\in\Delta,\, \mathtt{xPz}\in T} \mathtt{xPz.q}\ \Big). \end{aligned}$$

Note that $\mathtt{xPz} \in T \Rightarrow \mathtt{xP} \in T$, and then one can check that Algorithm 2 exactly computes $\mathtt{Pz.q}$ as this maximum over three maxima, hence $\mathtt{Pz.q} = q(\mathtt{z}|\mathtt{P})$.

Algorithm 1 Max-ARPA: first pass

```
1: for Z ∈ T do
2:     Z.m ← 1
3:     for x ∈ Δ s.t xZ ∈ T do
4:         Z.m ← max(Z.m, xZ.b × xZ.m)
5:     end for
6: end for
```

Algorithm 2 Max-ARPA: second pass

```
1: for Z = Pz ∈ T do
2:     Pz.q ← Pz.p
3:     for x ∈ Δ s.t xP ∈ T do
4:         if xPz ∈ T then
5:             Pz.q ← max(Pz.q, xPz.q)
6:         else
7:             Pz.q ← max(Pz.q, Pz.p × xP.b × xP.m)
8:         end if
9:     end for
10: end for
```

4.4 Search

The search for the true optimum derivation is illustrated in Algorithm 3. The algorithm takes as input the initial proposal distribution $g^{(0)}(\mathbf{d})$ (see §4.2, Figure 3) and a maximum error ϵ (which we set to a small constant 0.001 rather than zero, to avoid problems with floating point precision). In line 3 we find the optimum derivation $\mathbf{d}$ in $g^{(0)}$ (see §4.5). The variable g^* stores the maximum score w.r.t. the current proposal, while the variable f^* stores the maximum score observed thus far w.r.t. the true model (note that in line 5 we assess the true score of $\mathbf{d}$). In line 6 we start a loop that runs until the error falls below ϵ. This *error* is the difference (in $\log$-domain) between the proxy maximum g^* and the best true score observed thus far f^*.[13] In line 7, we refine the current proposal using evidence from $\mathbf{d}$ (see §4.6). In line 9, we update the maximum derivation searching through the refined proposal. In line 11, we keep track of the best score so far according to the true model, in order to compute the updated gap in line 6.

4.5 Dynamic Programming

Finding the best derivation in a proposal hypergraph is straightforward with standard dynamic programming. We can compute *inside weights* in the *max-times* semiring in time proportional to $O(|V| + |E|)$ (Goodman, 1999). Once inside weights have been computed, finding the Viterbi-derivation starting from the root is straightforward. A simple, though important, optimisation concerns the computation of inside weights. The inside algorithm (Baker, 1979) requires a bottom-up traverse of the nodes in V. To do that, we topologically sort the nodes in V at time $t = 0$ and maintain a sorted list of nodes as we refine g throughout the search – thus avoiding having to recompute the partial ordering of the nodes at every iteration.

[13] Because $g^{(t)}$ upperbounds f everywhere, in optimisation we have a guarantee that the maximum of f must lie in the interval $[f^*, g^*)$ (see Figure 2) and the quantity $g^* - f^*$ is an upperbound on the error that we incur if we early-stop the search at any given time t. This bound provides a principled criterion in trading accuracy for performance (a direction that we leave for future work). Note that most algorithms for approximate search produce solutions with unbounded error.

Algorithm 3 Exact decoding

```
1: function OPTIMISE(g^(0), ε)
2:     t ← 0                                          ▷ step
3:     d ← argmax_d g^(t)(d)
4:     g* ← g^(t)(d)
5:     f* ← f(d)
6:     while (g* − f* ≥ ε) do      ▷ ε is the maximum error
7:         g^(t+1) ← refine(g^(t), d)        ▷ update proposal
8:         t ← t + 1
9:         d ← argmax_d g^(t)(d)             ▷ update argmax
10:        g* ← g^(t)(d)
11:        f* ← max(f*, f(d))        ▷ update "best so far"
12:    end while
13:    return g^(t), d
14: end function
```

4.6 Refinement

If a derivation $\mathbf{d} = \operatorname{argmax}_{\mathbf{d}} g^{(t)}(\mathbf{d})$ is such that $g^{(t)}(\mathbf{d}) \ll f(\mathbf{d})$, there must be in $\mathbf{d}$ at least one n-gram whose upperbound LM weight is far above its true LM weight. We then lower $g^{(t)}$ locally by refining only nonterminal nodes that participate in $\mathbf{d}$. Nonterminal nodes are refined by having their LM states extended one word at a time.[14]

For an illustration, assume we are performing optimisation with a bigram LM. Suppose that in the first iteration a derivation $\mathbf{d}_0 = \operatorname{argmax}_{\mathbf{d}} g^{(0)}(\mathbf{d})$ is obtained. Now consider an edge in $\mathbf{d}_0$

$$[l, C, r, \epsilon]\ \ \alpha y_1 \xrightarrow{w} [l_0, C_0, r_0, \epsilon]$$

where an empty LM state is made explicit (with an empty string ϵ) and αy_1 represents a target phrase. We refine the edge's head $[l_0, C_0, r_0, \epsilon]$ by creating a node based on it, however, with an extended LM state, i.e., $[l_0, C_0, r_0, y_1]$. This motivates a split of the set of incoming edges to the original node, such that, if the target projection of an incoming

[14] The refinement operation is a special case of a general finite-state intersection. However, keeping its effect local to derivations going through a specific node is non-trivial using the general mechanism and justifies a tailored operation.

edge ends in y_1, that edge is reconnected to the new node as below.

$$[l, C, r, \epsilon]\ \ \alpha y_1 \xrightarrow{w} [l_0, C_0, r_0, y_1]$$

The outgoing edges from the new node are reweighted copies of those leaving the original node. That is, outgoing edges such as

$$[l_0, C_0, r_0, \epsilon]\ \ y_2\beta \xrightarrow{w} \left[l', C', r', \gamma'\right]$$

motivate edges such as

$$[l_0, C_0, r_0, y_1]\ \ y_2\beta \xrightarrow{w \otimes w'} \left[l', C', r', \gamma'\right]$$

where $w' = \lambda_\psi {}^{q_{\text{LM}}(y_1 y_2)}/_{q_{\text{LM}}(y_2)}$ is a change in LM probability due to an extended context.

Figure 4 is the logic program that constructs the refined hypergraph in the general case. In comparison to Figure 3, items are now extended to store an LM state. The input is the original hypergraph $G = \langle V, E\rangle$ and a node $\mathbf{v}_0 \in V$ to be refined by left-extending its LM state γ_0 with the word y. In the program, $\left\langle \mathbf{u}\sigma \xrightarrow{w} \mathbf{v} \right\rangle$ with $\mathbf{u}, \mathbf{v} \in V$ and $\sigma \in \Delta^*$ represents an edge in E. An item $[l, C, r, \gamma]_{\mathbf{v}}$ (annotated with a state $\mathbf{v} \in V$) represents a node (in the refined hypergraph) whose signature is equivalent to $\mathbf{v}$ (in the input hypergraph). We start with AXIOMS by copying the nodes in G. In COPY, edges from G are copied unless they are headed by $\mathbf{v}_0$ and their target projections end in $y\gamma_0$ (the extended context). Such edges are processed by REFINE, which instead of copying them, creates new ones headed by a refined version of $\mathbf{v}_0$. Finally, REWEIGHT continues from the refined node with reweighted copies of the edges leaving $\mathbf{v}_0$. The weight update represents a change in LM probability (w.r.t. the upper-bound distribution) due to an extended context.

INPUT
$G = \langle V, E\rangle$
$\mathbf{v}_0 = [l_0, C_0, r_0, \gamma_0] \in V$ where $\gamma_0 \in \Delta^*$
$y \in \Delta$
ITEM $[l, C, r, \gamma \in \Delta^*]$
AXIOMS

$$\frac{}{[l, C, r, \gamma]_{\mathbf{v}}}\ \ \mathbf{v} \in V$$

COPY

$$\frac{[l, C, r, \alpha]_{\mathbf{u}} \quad \left\langle \mathbf{u}\beta \xrightarrow{w} \mathbf{v} \right\rangle}{[l', C', r', \alpha']_{\mathbf{v}} : w} \quad \begin{array}{l} \mathbf{v} \neq \mathbf{v}_0 \vee \alpha\beta \neq \sigma y \gamma_0 \\ \alpha, \alpha', \beta, \sigma \in \Delta^* \end{array}$$

REFINE

$$\frac{[l, C, R, \alpha]_{\mathbf{u}} \quad \left\langle \mathbf{u}\beta \xrightarrow{w} \mathbf{v}_0 \right\rangle}{[l_0, C_0, r_0, y\gamma_0] : w} \quad \begin{array}{l} \alpha\beta = \sigma y \gamma_0 \\ \alpha, \beta, \sigma \in \Delta^* \end{array}$$

REWEIGHT

$$\frac{[l_0, C_0, r_0, y\gamma_0] \quad \left\langle \mathbf{v}_0\sigma \xrightarrow{w} \mathbf{v} \right\rangle}{[l, C, r, \gamma]_{\mathbf{v}} : w \otimes w'}\ \ \sigma, \gamma \in \Delta^*$$

where $w' = \lambda_\psi \dfrac{q_{\text{LM}}(y\gamma_0)}{q_{\text{LM}}(\gamma_0)}$

Figure 4: Local intersection via LM right state refinement. The input is a hypergraph $G = \langle V, E\rangle$, a node $\mathbf{v}_0 \in V$ singly identified by its carry $[l_0, C_0, r_0, \gamma_0]$ and a left-extension y for its LM context γ_0. The program copies most of the edges $\left\langle \mathbf{u}\sigma \xrightarrow{w} \mathbf{v} \right\rangle \in E$. If a derivation goes through $\mathbf{v}_0$ and the string under $\mathbf{v}_0$ ends in $y\gamma_0$, the program refines and reweights it.

5 Experiments

We used the dataset made available by the Workshop on Statistical Machine Translation (WMT) (Bojar et al., 2013) to train a German-English phrase-based system using the Moses toolkit (Koehn et al., 2007) in a standard setup. For phrase extraction, we used both Europarl (Koehn, 2005) and News Commentaries (NC) totalling about 2.2M sentences.[15] For language modelling, in addition to the monolingual parts of Europarl and NC, we added News-2013 totalling about 25M sentences. We performed language model interpolation and *batch-mira* tuning (Cherry and Foster, 2012) using *newstest2010* (2,849 sentence pairs). For tuning we used cube pruning with a large beam size ($k = 5000$) and a distortion limit $d = 4$. Unpruned language models were trained using *lmplz* (Heafield et al., 2013b) which employs modified Kneser-Ney smoothing (Kneser and Ney, 1995). We report results on *newstest2012*.

Our exact decoder produces optimal translation derivations for all the 3,003 sentences in the test set. Table 1 summarises the performance of our novel decoder for language models of order $n = 3$ to $n = 5$. For 3-gram LMs we also varied the distortion limit d (from 4 to 6). We report the average time (in seconds) to build the initial proposal, the total run time of the algorithm, the number of iterations N before convergence, and the size of the hypergraph in the end of the search (in thousands of nodes and thousands of edges).[16]

[15] Pre-processing: tokenisation, truecasing and automatic compound-splitting (German only). Following Durrani et al. (2013), we set the maximum phrase length to 5.

[16] The size of the initial proposal does not depend on LM order, but rather on distortion limit (see Figure 3): on average (in thousands) $|V_0| = 0.6$ and $|E_0| = 27$ with $d = 4$, $|V_0| = 1.3$ and $|E_0| = 70$ with $d = 5$, and $|V_0| = 2.5$ and

n	d	build (s)	total (s)	N	$\|V\|$	$\|E\|$
3	4	1.5	21	190	2.5	159
3	5	3.5	55	303	4.4	343
3	6	10	162	484	8	725
4	4	1.5	50	350	4	288
5	4	1.5	106	555	6.1	450

Table 1: Performance of the exact decoder in terms of: time to *build* $g^{(0)}$, *total* decoding time including *build*, number of *iterations* (N), and number of nodes and edges (in thousands) at the end of the search.

n	Nodes at level m					LM states at level m			
	0	1	2	3	4	1	2	3	4
3	0.4	1.2	0.5	-	-	113	263	-	-
4	0.4	1.6	1.4	0.3	-	132	544	212	-
5	0.4	2.1	2.4	0.7	0.1	142	790	479	103

Table 2: Average number of nodes (in thousands) whose LM state encode an m-gram, and average number of unique LM states of order m in the final hypergraph for different n-gram LMs ($d = 4$ everywhere).

It is insightful to understand how different aspects of the initial proposal impact on performance. Increasing the translation option limit (*tol*) leads to $g^{(0)}$ having more edges (this dependency is linear with *tol*). In this case, the number of nodes is only minimally affected — due to the possibility of a few new segmentations. The maximum phrase length (*mpl*) introduces in $g^{(0)}$ more configurations of reordering constraints ($[l, C]$ in Figure 3). However, not many more, due to C being limited by the distortion limit d. In practice, we observe little impact on time performance. Increasing d introduces many more permutations of the input leading to exponentially many more nodes and edges. Increasing the order n of the LM has no impact on $g^{(0)}$ and its impact on the overall search is expressed in terms of a higher number of nodes being locally intersected.

An increased hypergraph, be it due to additional nodes or additional edges, necessarily leads to slower iterations because at each iteration we must compute inside weights in time $O(|V|+|E|)$. The number of nodes has the larger impact on the number of iterations. OS* is very efficient in ignoring hypotheses (edges) that cannot compete for an optimum. For instance, we observe that running time depends linearly on *tol* only through the computation of inside weights, while the number of iterations is only minimally affected.[17] An increased LM order, for a fixed distortion limit, impacts much more on the number of iterations than on the average running time of a single iteration. Fixing $d = 4$, the average time per iteration is 0.1 ($n = 3$), 0.13 ($n = 4$) and 0.18 ($n = 5$). Fixing a 3-gram LM, we observe 0.1 ($d = 4$), 0.17 ($d = 5$) and 0.31 ($d = 6$). Note the exponential growth of the latter, due to a proposal encoding exponentially many more permutations.

Table 2 shows the average degree of refinement of the nodes in the final proposal. Nodes are shown by level of refinement, where m indicates that they store m words in their carry. The table also shows the number of unique m-grams ever incorporated to the proposal. This table illustrates well how our decoding algorithm moves from a coarse upperbound where every node stores an empty string to a variable-order representation which is sufficient to prove an optimum derivation.

In our approach a complete derivation is optimised from the proxy model at each iteration. We observe that over 99% of these derivations project onto distinct strings. In addition, while the optimum solution may be found early in the search, a certificate of optimality requires refining the proxy until convergence (see §4.1). It turns out that most of the solutions are first encountered as late as in the last 6-10% of the iterations.

We use the optimum derivations obtained with our exact decoder to measure the number of search errors made by beam search and cube pruning with increasing beam sizes (see Table 3). Beam search reaches optimum derivations with beam sizes $k \geq 500$ for all language models tested. Cube pruning, on the other hand, still makes mistakes at $k = 1000$. Table 4 shows translation quality achieved with different beam sizes for cube pruning and compares it to exact decoding. Note that for $k \geq 10^4$ cube pruning converges to optimum

$|E_0| = 178$ with $d = 6$. Observe the exponential dependency on distortion limit, which also leads to exponentially longer running times.

[17] It is possible to reduce the size of the hypergraph throughout the search using the upperbound on the search error $g^* - f^*$ to prune hypotheses that surely do not stand a chance of competing for the optimum (Graehl, 2005). Another direction is to group edges connecting the same nonterminal nodes into one *partial edge* (Heafield et al., 2013a) — this is particularly convenient due to our method only visiting the 1-best derivation from $g(\mathbf{d})$ at each iteration.

k	Beam search			Cube pruning		
	3	4	5	3	4	5
10	938	1294	1475	2168	2347	2377
10^2	19	60	112	613	999	1126
10^3	0	0	0	29	102	167
10^4	0	0	0	0	4	7

Table 3: Beam search and cube pruning search errors (out of 3,003 test samples) by beam size using LMs of order 3 to 5 ($d = 4$).

order	3			4	5
k	$d = 4$	$d = 5$	$d = 6$	$d = 4$	$d = 4$
10	20.47	20.13	19.97	20.71	20.69
10^2	21.14	21.18	21.08	21.73	21.76
10^3	21.27	21.34	21.32	21.89	21.91
10^4	21.29	21.37	21.37	21.92	21.93
OS*	21.29	21.37	21.37	21.92	21.93

Table 4: Translation quality in terms of BLEU as a function of beam size in cube pruning with language models of order 3 to 5. The bottom row shows BLEU for our exact decoder.

derivations in the vast majority of the cases (100% with a 3-gram LM) and translation quality in terms of BLEU is no different from OS*. However, with $k < 10^4$ both model scores and translation quality can be improved. Figure 5 shows a finer view on search errors as a function of beam size for LMs of order 3 to 5 (fixed $d = 4$). In Figure 6, we fix a 3-gram LM and vary the distortion limit (from 4 to 6). Dotted lines correspond to beam search and dashed lines correspond to cube pruning.

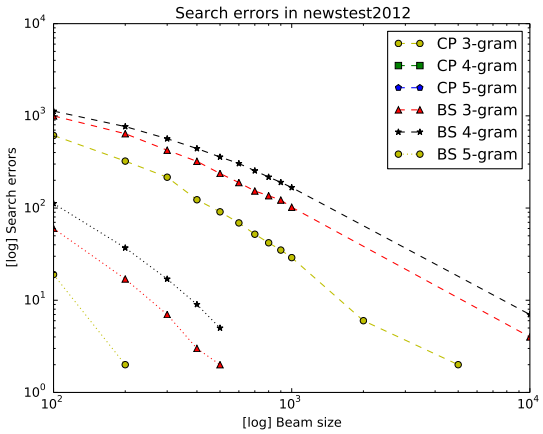


Figure 5: Search errors made by beam search and cube pruning as a function of beam-size.

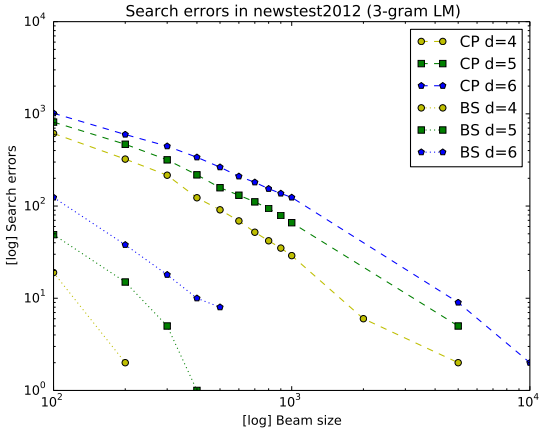


Figure 6: Search errors made by beam search and cube pruning as a function of the distortion limit (decoding with a 3-gram LM).

6 Conclusions and Future Work

We have presented an approach to decoding with unpruned hypergraphs using upperbounds on the language model distribution. The algorithm is an instance of a coarse-to-fine strategy with connections to A* and adaptive rejection sampling known as OS*. We have tested our search algorithm using state-of-the-art phrase-based models employing robust language models. Our algorithm is able to decode all sentences of a standard test set in manageable time consuming very little memory. We have performed an analysis of search errors made by beam search and cube pruning and found that both algorithms perform remarkably well for phrase-based decoding. In the case of cube pruning, we show that model score and translation quality can be improved for beams $k < 10,000$.

There are a number of directions that we intend to investigate to speed up our decoder, such as: (1) error-safe pruning based on search error bounds; (2) use of reinforcement learning to guide the decoder in choosing which n-gram contexts to extend; and (3) grouping edges into partial edges, effectively reducing the size of the hypergraph and ultimately computing inside weights in less time.

Acknowledgments

The work of Wilker Aziz and Lucia Specia was supported by EPSRC (grant EP/K024272/1).

References

Michael Auli, Adam Lopez, Hieu Hoang, and Philipp Koehn. 2009. A systematic analysis of translation model search spaces. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT '09, pages 224–232, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wilker Aziz, Marc Dymetman, and Sriram Venkatapathy. 2013. Investigations in exact inference for hierarchical translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 472–483, Sofia, Bulgaria, August. Association for Computational Linguistics.

James K. Baker. 1979. Trainable grammars for speech recognition. In *Proceedings of the Spring Conference of the Acoustical Society of America*, pages 547–550, Boston, MA, June.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August. Association for Computational Linguistics.

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

Simon Carter, Marc Dymetman, and Guillaume Bouchard. 2012. Exact sampling and decoding in high-order hidden Markov models. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1125–1134, Jeju Island, Korea, July. Association for Computational Linguistics.

Yin-Wen Chang and Michael Collins. 2011. Exact decoding of phrase-based translation models through Lagrangian relaxation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 26–37, Stroudsburg, PA, USA. Association for Computational Linguistics.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 427–436, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 263–270, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33:201–228.

Shay B. Cohen, Robert J. Simmons, and Noah A. Smith. 2008. Dynamic programming algorithms as products of weighted logic programs. In Maria Garcia de la Banda and Enrico Pontelli, editors, *Logic Programming*, volume 5366 of *Lecture Notes in Computer Science*, pages 114–129. Springer Berlin Heidelberg.

G Dantzig, R Fulkerson, and S Johnson. 1954. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410.

Nadir Durrani, Barry Haddow, Kenneth Heafield, and Philipp Koehn. 2013. Edinburgh's machine translation systems for European language pairs. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 114–121, Sofia, Bulgaria, August. Association for Computational Linguistics.

Chris Dyer and Philip Resnik. 2010. Context-free reordering, finite-state translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 858–866, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marc Dymetman, Guillaume Bouchard, and Simon Carter. 2012. Optimization and sampling for NLP from a unified viewpoint. In *Proceedings of the First International Workshop on Optimization Techniques for Human Language Technology*, pages 79–94, Mumbai, India, December. The COLING 2012 Organizing Committee.

Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. 1993. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2-3):177–201, April.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 228–235, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joshua Goodman. 1999. Semiring parsing. *Comput. Linguist.*, 25(4):573–605, December.

Jonathan Graehl. 2005. Relatively useless pruning. Technical report, USC Information Sciences Institute.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions On Systems Science And Cybernetics*, 4(2):100–107.

Kenneth Heafield, Philipp Koehn, and Alon Lavie. 2013a. Grouping language model boundary words

to speed k-best extraction from hypergraphs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 958–968, Atlanta, Georgia, USA, June.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013b. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696, Sofia, Bulgaria, August. Association for Computational Linguistics.

Kenneth Heafield, Michael Kayser, and Christopher D. Manning. 2014. Faster Phrase-Based decoding by refining feature state. In *Proceedings of the Association for Computational Linguistics*, Baltimore, MD, USA, June.

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

Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. 2009. Rule filtering by pattern for efficient hierarchical translation. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 380–388, Athens, Greece, March. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Series in Artificial Intelligence. Prentice Hall, 1 edition.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. *Acoustics, Speech, and Signal Processing*, 1:181–184.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Comput. Linguist.*, 25(4):607–615, December.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit*, pages 79–86.

Shankar Kumar, Yonggang Deng, and William Byrne. 2006. A weighted finite state transducer translation template model for statistical machine translation. *Natural Language Engineering*, 12(1):35–75, March.

Zhifei Li and Sanjeev Khudanpur. 2008. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proceedings of the Second Workshop on Syntax and Structure in Statistical Translation*, SSST '08, pages 10–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Adam Lopez. 2008. Statistical machine translation. *ACM Computing Surveys*, 40(3):8:1–8:49, August.

Adam Lopez. 2009. Translation as weighted deduction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 532–540, Stroudsburg, PA, USA. Association for Computational Linguistics.

Franz Josef Och, Nicola Ueffing, and Hermann Ney. 2001. An efficient A* search algorithm for statistical machine translation. In *Proceedings of the workshop on Data-driven methods in machine translation - Volume 14*, DMMT '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1 of *ACL '03*, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Slav Petrov, Aria Haghighi, and Dan Klein. 2008. Coarse-to-fine syntactic machine translation using language projections. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 108–116, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sebastian Riedel and James Clarke. 2009. Revisiting optimal decoding for machine translation IBM model 4. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '09, pages 5–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christian P. Robert and George Casella. 2004. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Alexander M. Rush and Michael Collins. 2011. Exact decoding of syntactic translation models through Lagrangian relaxation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 72–82, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christoph Tillmann, Stephan Vogel, Hermann Ney, and A. Zubiaga. 1997. A DP based search using monotone alignments in statistical translation. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, EACL '97, pages 289–296, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mikhail Zaslavskiy, Marc Dymetman, and Nicola Cancedda. 2009. Phrase-based statistical machine translation as a traveling salesman problem. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 333–341, Stroudsburg, PA, USA. Association for Computational Linguistics.

Large-scale Expected BLEU Training of Phrase-based Reordering Models

Michael Auli, Michel Galley, Jianfeng Gao
Microsoft Research
Redmond, WA, USA
{michael.auli,mgalley,jfgao}@microsoft.com

Abstract

Recent work by Cherry (2013) has shown that directly optimizing phrase-based reordering models towards BLEU can lead to significant gains. Their approach is limited to small training sets of a few thousand sentences and a similar number of sparse features. We show how the expected BLEU objective allows us to train a simple linear discriminative reordering model with millions of sparse features on hundreds of thousands of sentences resulting in significant improvements. A comparison to likelihood training demonstrates that expected BLEU is vastly more effective. Our best results improve a hierarchical lexicalized reordering baseline by up to 2.0 BLEU in a single-reference setting on a French-English WMT 2012 setup.

1 Introduction

Modeling reordering for phrase-based machine translation has been a long standing problem. Contrary to synchronous context free grammar-based translation models (Wu, 1997; Galley et al., 2004; Galley et al., 2006; Chiang, 2007), phrase-based models (Koehn et al., 2003; Och and Ney, 2004) have no in-built notion of reordering beyond what is captured in a single phrase pair, and the first phrase-based decoders simply scored inter-phrase reorderings using a restricted linear distortion feature, which scores a phrase reordering proportionally to the length of its displacement. While phrase-based models allow in theory completely unrestricted reordering patterns, movements are generally limited to a finite distance for complexity reasons. To address this limitation, extensive prior work focused on richer feature sets, in particular on lexicalized reordering models trained with maximum likelihood-based approaches (Tillmann, 2003; Xiong et al., 2006; Galley and Manning, 2008; Nguyen et al.,2009;§2).

More recently, Cherry (2013) proposed a very effective sparse ordering model relying on a set of only a few thousand indicator features which are trained towards a task-specific metric such as BLEU (Papineni et al., 2002). These features are simply added to the log-linear framework of translation that is trained with the Margin Infused Relaxed Algorithm (MIRA; Chiang et al., 2009) on a small development set of a few thousand sentences. While simple, the approach outperforms the state-of-the-art hierarchical reordering model of Galley and Manning (2008), a maximum likelihood-based model trained on millions of sentences to fit millions of parameters.

Ideally, we would like to scale sparse reordering models to similar dimensions but recent attempts to increase the amount of training data for MIRA was met with little success (Eidelman et al., 2013). In this paper we propose much larger sparse ordering models that combine the scalability of likelihood-based approaches with the higher accuracy of maximum BLEU training (§3). We train on the output of a hierarchical reordering model-based system and scale to millions of features learned on hundreds of thousands of sentences (§4). Specifically, we use the expected BLEU objective function (Rosti et al., 2010; Rosti et al., 2011; He and Deng, 2012; Gao and He, 2013; Gao et al., 2014; Green et al., 2014) which allows us to train models that use training data and feature sets that are two to three orders of magnitudes larger than in previous work (§5).

Our models significantly outperform the state-of-the-art hierarchical lexicalized reordering model on two language pairs and we demonstrate that richer feature sets result in significantly higher accuracy than with a feature set similar to Cherry (2013). We also demonstrate that our

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1250–1260,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

approach greatly benefits from more training data than is typically used for maximum BLEU training. Previous work concluded that sparse reordering models perform better than maximum entropy models, however, the two approaches do not only differ in the objective function but also the type of training data (Cherry, 2013). Our analysis isolates the objective function and shows that expected BLEU optimization is the most important factor to train accurate ordering models. Finally, we compare expected BLEU training to pair-wise ranked optimization (PRO) on a feature set similar to Cherry (2013; §7).

2 Reordering Models

Reordering models for phrase-based translation are typically part of the log-linear framework which forms the basis of many statistical machine translation systems (Och and Ney, 2004).

Formally, we are given K training pairs $\mathcal{D} = (f^{(1)}, e^{(1)})...(f^{(K)}, e^{(K)})$, where each $f^{(i)} \in \mathcal{F}$ is drawn from a set of possible foreign sentences, and each English sentence $e^{(i)} \in \mathcal{E}(f^{(i)})$ is drawn from a set of possible English translations of $f^{(i)}$. The log-linear model is parameterized by m parameters θ where each $\theta_k \in \theta$ is the weight of an associated feature $h_k(f, e)$ such as a language model or a reordering model. Function $h(f, e)$ maps foreign and English sentences to the vector $h_1(f, e)...h_m(f, e)$, and we usually choose translations $\hat{e}$ according to the following decision rule:

$$\hat{e} = \arg\max_{e \in \mathcal{E}(f)} \theta^{\mathsf{T}} h(f, e) \quad (1)$$

In practice, computing $\hat{e}$ exactly is intractable and we resort to an approximate but more efficient beam search (Och and Ney, 2004).

Early phrase-based models simply relied on a linear distortion feature, which measures the distance between the first word of the current source phrase and the last word of the previous source phrase (Koehn et al., 2003; Och and Ney, 2004). Unfortunately, this approach is agnostic to the actual phrases being reordered, and does not take into account that certain phrases are more likely to be reordered than others. This shortcoming led to a range of *lexicalized* reordering models that capture exactly those preferences for individual phrases (Tillmann, 2003; Koehn et al., 2007).

Reordering models generally assume a sequence of English phrases $e = \{\bar{e}_1, \ldots, \bar{e}_n\}$ currently hypothesized by the decoder, a phrase alignment $a = \{a_1, \ldots, a_n\}$ that defines a foreign phrase $\bar{f}_{a_i}$ for each English phrase $\bar{e}_i$, and an *orientation* o_i which describes how a phrase pair should be reordered with respect to the previous phrases. There are typically three orientation types and the exact definition depends on the specific models which we describe below. Orientations can be determined during decoding and from word-aligned training corpora. Most models estimate a probability distribution $p(o_i|pp_i, a_1, \ldots, a_i)$ for the i-th phrase pair $pp_i = \langle \bar{e}_i, \bar{f}_{a_i} \rangle$ and the alignments $a_1, \ldots, a_i$ of the previous target phrases.

Lexicalized Reordering. This model defines the three orientation types based only on the position of the current and previously translated source phrase a_i and a_{i-1}, respectively (Tillmann, 2003; Koehn et al., 2007). The orientation types generally are: monotone (M), indicating that a_{i-1} is directly followed by a_i. swap (S) assumes that a_i precedes a_{i-1}, i.e., the two phrases swap places. Finally, discontinuous (D) indicates that a_i is not adjacent to a_{i-1}. The probability distribution over these reordering events is based on a maximum likelihood estimate:

$$p(o|pp, a_{i-1}, a_i) = \frac{cnt(o, pp)}{cnt(pp)} \quad (2)$$

where $o \in \{M, S, D\}$ and cnt returns smoothed frequency counts over a word-aligned corpus.

Hierarchical Reordering. An extension of the lexicalized reordering model better handles long-distance reordering by conditioning the orientation of the current phrase on a context larger than just the previous phrase (Galley and Manning, 2008). In particular, the hierarchical reordering model does so by building a compact representations of the preceding context using an efficient shift-reduce parser. During translation new phrases get moved on a stack and are then combined with any previous phrase if they are adjacent. Figure 1 shows an illustrative example: when the decoder shifts phrase pp_8 onto the stack, this phrase is then merged with pp_7 (reduce operation), which then can be merged with previous phrases to finally form a hierarchical block h_1. These merge operations stop once we reach a phrase (here, pp_3) that is not contiguous with the current block. Then, as another phrase (pp_9) is hypothesized, the decoder uses the hierarchical block at the top of the stack (h_1) to determine the orientation of the current

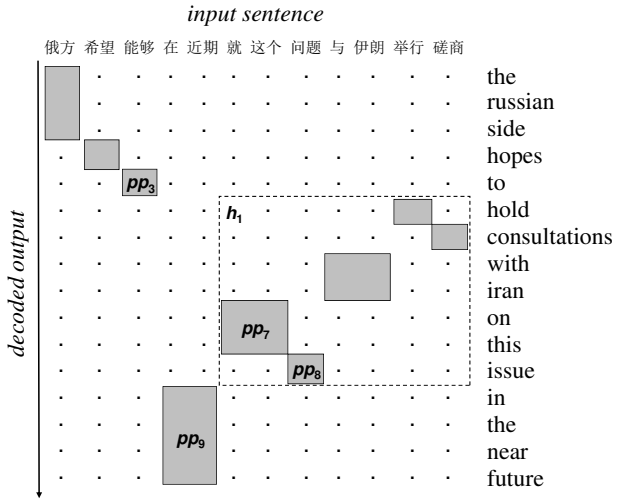


Figure 1: The hierarchical reordering model (HRM) analyzes a non-local context to determine the orientation of the current phrase. For example, the phrase pair pp_9 has a swap orientation ($o_9 = S$) with respect to a hierarchical block (h_1) that comprises the five preceding phrase pairs.

phrase pp_9, which in this case is a swap (S) orientation.[1] The model has the advantage that the orientations computed are more robust to derivational ambiguity of the underlying translation model. A given surface translation may be derived through different phrases but the shift-reduce parser combines them into a single representation which is more consistent with the orientations observed in the word-aligned training data.

Maximum Entropy-based models. The statistics used to estimate the lexicalized and the hierarchical reordering models are based on very sparse estimates, simply because certain phrases are not very frequent. Maximum entropy models address this problem by estimating Eq. 2 through sparse indicator features over phrase pairs instead, but prior work with such models still relies on word aligned corpora for estimation (Xiong et al., 2006; Nguyen et al., 2009). However, recent evaluations of the approach show little gain over the simpler frequency-based estimation method (Cherry, 2013).

Sparse Hierarchical Reordering model. All of the models so far are trained to maximize the likelihood of reordering decisions observed in word aligned corpora. Cherry (2013) argues that it is probably too difficult to learn human reordering patterns through noisy word alignments that were generated by unsupervised methods. Instead, he proposes to learn a discriminative reordering model based on the outputs of the actual machine translation system, adjusting the feature weights to maximize a task-specific objective, which is BLEU in their case. Their model is based on a set of sparse features derived from the hierarchical reordering model which we scale to millions of features (§6).

[1] Galley and Manning (2008) provide a more formal explanation.

3 A Simple Linear Reordering Model

Our reordering model is defined as a simple linear model over the basic orientation types, similar to Cherry (2013). In particular, our model defines score $s_\phi(o, e, f)$ over orientations $o = \{M, S, D\}$, and a sentence pair $\{e, f, a\}$ with alignment a as a linear combination of weighted indicator features:

$$\begin{aligned} s_\phi(o, e, f, a) &= \phi^\mathsf{T} u(o, e, f, a) \\ &= \sum_{i=1}^{I} \phi^\mathsf{T} u(o, pp_i, c_i) \\ &= \sum_{i=1}^{I} s_\phi(o, pp_i, c_i) \end{aligned} \quad (3)$$

where ϕ is a vector of weights, $\{pp_i\}_{i=1}^{I}$ is a set of phrases that decompose the sentence pair $\{e, f, a\}$, and $u(o, pp_i, c_i)$ is a function that maps orientation o, phrase pair pp_i and local context c_i to a sparse vector of indicator features. The local context c_i represents information used by the model that is in addition to the phrase pair. For example, the features of Cherry (2013) condition on the top-stack of the hierarchical shift reduce parser, information that is non-local with respect to the phrase pair. In our experiments, we use features that go beyond the top-stack, in order to condition on various parts of the source and target side contexts (§7).

4 Model Training

Optimization of our model is based on standard stochastic gradient descent (SGD; Bottou, 2004) with an expected BLEU loss $l(\phi)$ which we detail next (§5). The update is:

$$\phi_t = \phi_{t-1} - \mu \frac{\partial l(\phi_{t-1})}{\partial \phi_{t-1}} \quad (4)$$

where ϕ_t and ϕ_{t-1} are model weights at time t and $t-1$ respectively, and μ is a learning rate.

We add the model as a small number of dense features to the log-linear framework of translation

(Eq. 1). Specifically, we extend the m baseline features by a set of new features $h_{m+1}, \ldots, h_{m+j}$, where each represents a linear combination of sparse indicator features corresponding to one of the orientation types. Exposing each orientation as a separate dense feature within the log-linear model is common practice for lexicalized reordering models (Koehn et al., 2005):

$$h_{m+j} = s_\phi(o_j, e, f, a)$$

where $o_j \in \{M, S, D\}$.

The translation model is then parameterized by both θ, the log-linear weights of the baseline features, as well as ϕ, the weights of the reordering model. The reordering model is learned as follows (Gao and He, 2013; Gao et al., 2014):

1. We first train a baseline translation system to learn θ, without the discriminative reordering model, i.e., we set $\theta_{m+1} = 0, \ldots, \theta_{m+j} = 0$.

2. Using these weights, we generate n-best lists for the foreign sentences in the training data using the setup described in the experimental section (§7). The n-best lists serve as an approximation to $\mathcal{E}(f)$, the set of possible translations of f, used in the next step for expected BLEU training of the reordering model (§5).

3. Next, we fix θ, set $\theta_{m+1} = 1, \ldots \theta_{m+j} = 1$ and optimize ϕ with respect to the loss function on the training data using stochastic gradient descent.[2]

4. Finally, we fix ϕ and re-optimize θ in the presence of the discriminative reordering model using Minimum Error Rate Training (MERT; Och 2003; §7).

We found that re-optimizing θ after a few iterations of stochastic gradient descent in step 3 did not improve accuracy.

5 Expected BLEU Objective Function

The expected BLEU objective (Gao and He, 2013; Gao et al., 2014) allows us to efficiently optimize a large scale discriminative reordering model towards the desired task-specific metric, which in our setting is BLEU.

[2]We tuned $\theta_{m+1}, \ldots \theta_{m+j}$ on the development set but found that setting them uniformly to one resulted in faster training and equal accuracy.

Formally, we define our loss function $l(\phi)$ as the negative expected BLEU score, denoted as xBLEU(ϕ), for a given foreign sentence f and a log-linear parameter set θ:

$$\begin{aligned} l(\phi) &= -\text{ xBLEU}(\phi) \\ &= -\sum_{e \in \mathcal{E}(f)} p_{\theta,\phi}(e|f)\, \text{sBLEU}(e, e^{(i)}) \end{aligned} \quad (5)$$

where sBLEU$(e, e^{(i)})$ is a smoothed sentence-level BLEU score with respect to the reference translation $e^{(i)}$, and $\mathcal{E}(f)$ is the generation set approximated by an n-best list. In our experiments we use n-best lists with unique entries and therefore our definitions do not take into account multiple derivations of the same translation. Specifically, our n-best lists are generated by choosing the highest scoring derivation $\hat{e}$ amongst string identical translations e for f. We use a sentence-level BLEU approximation similar to Gao et al. (2014).[3] Finally, $p_{\theta,\phi}(e|f)$ is the normalized probability of translation e given f, defined as:

$$p_{\theta,\phi}(e|f) = \frac{\exp\{\gamma \theta^\mathsf{T} h(f, e)\}}{\sum_{e' \in \mathcal{E}(f)} \exp\{\gamma \theta^\mathsf{T} h(f, e')\}} \quad (6)$$

where $\theta^\mathsf{T} h(f, e)$ includes the discriminative reordering model $h_{m+1}(e, f), \ldots, h_{m+j}(e, f)$ parameterized by ϕ, and $\gamma \in [0, \inf)$ is a tuned scaling factor that flattens the distribution for $\gamma < 1$ and sharpens it for $\gamma > 1$ (Tromble et al., 2008).[4]

Next, we define the gradient of the expected BLEU loss function $l(\phi)$. To simplify our notation we omit the local context c in $s_\phi(o, pp, c)$ (Eq. 3) from now on and assume it to be part of pp. Using the observation that the loss does not explicitly depend on ϕ, we get:

$$\begin{aligned} \frac{\partial l(\phi)}{\partial \phi} &= \sum_{o,pp} \frac{\partial l(\phi)}{\partial s_\phi(o, pp)} \frac{\partial s_\phi(o, pp)}{\partial \phi} \\ &= \sum_{o,pp} -\delta_{o,pp} u(o, pp) \end{aligned}$$

where $\delta_{o,pp}$ is the *error term* for orientation o of phrase pair pp:

$$\delta_{o,pp} = -\frac{\partial l(\phi)}{\partial s_\phi(o, pp)}$$

[3]We found in early experiments that the BLEU+1 approximation used by Liang et al. (2006) and Nakov et. al (2012) worked equally well in our setting.

[4]γ is only used during expected BLEU training.

The error term indicates how the expected BLEU loss changes with the reordering score which we derive in the next section.

Finally, the gradient of the reordering score $s_\phi(o, pp)$ with respect to ϕ is simply given by this:

$$\frac{\partial s_\phi(o, pp)}{\partial \phi} = \frac{\partial \phi^\mathsf{T} u(o, pp)}{\partial \phi} = u(o, pp)$$

5.1 Derivation of the Error Term $\delta_{o,pp}$

We rewrite the loss function (Eq. 5) using Eq. 6 and separate it into two terms $G(\phi)$ and $Z(\phi)$:

$$l(\phi) = -\text{xBLEU}(\phi) = -\frac{G(\phi)}{Z(\phi)} \quad (7)$$
$$= -\frac{\sum_{e \in \mathcal{E}(f)} \exp\{\gamma \theta^\mathsf{T} h(f, e)\} \, \text{sBLEU}(e, e^{(i)})}{\sum_{e' \in \mathcal{E}(f)} \exp\{\gamma \theta^\mathsf{T} h(f, e')\}}$$

Next, we apply the quotient rule of differentiation:

$$\delta_{o,pp} = \frac{\partial \text{xBLEU}(\phi)}{\partial s_\phi(o, pp)} = \frac{\partial (G(\phi)/Z(\phi))}{\partial s_\phi(o, pp)}$$
$$= \frac{1}{Z(\phi)} \left(\frac{\partial G(\phi)}{\partial s_\phi(o, pp)} - \frac{\partial Z(\phi)}{\partial s_\phi(o, pp)} \text{xBLEU}(\phi) \right)$$

The gradients for $G(\phi)$ and $Z(\phi)$ with respect to $s_\phi(o, pp)$ are:

$$\frac{\partial G(\phi)}{\partial s_\phi(o, pp)} = \sum_{e \in \mathcal{E}(f)} \text{sBLEU}(e, e^{(i)}) \frac{\partial \exp\{\gamma \theta^\mathsf{T} h(f, e)\}}{\partial s_\phi(o, pp)}$$
$$\frac{\partial Z(\phi)}{\partial s_\phi(o, pp)} = \sum_{e \in \mathcal{E}(f)} \frac{\partial \exp\{\gamma \theta^\mathsf{T} h(f, e)\}}{\partial s_\phi(o, pp)}$$

By using the following definition:

$$U(\phi, e) = \text{sBLEU}(e, e^{(i)}) - \text{xBLEU}(\phi)$$

together with the chain rule, Eq. 6 and Eq. 7, we can rewrite $\delta_{o,pp}$ as follows:

$$\delta_{o,pp} = \frac{1}{Z(\phi)} \sum_{e \in \mathcal{E}(f)} \left(\frac{\partial \exp\{\gamma \theta^\mathsf{T} h(f, e)\}}{\partial s_\phi(o, pp)} U(\phi, e) \right)$$
$$= \sum_{e \in \mathcal{E}(f)} \left(p_{\theta,\phi}(e|f) \frac{\partial \gamma \theta^\mathsf{T} h(f, e)}{\partial s_\phi(o, pp)} U(\phi, e) \right)$$

Because ϕ is only relevant to the reordering model, represented by $h_{m+1}, \ldots, h_{m+j}$, we have:

$$\frac{\partial \gamma \theta^\mathsf{T} h(f, e)}{\partial s_\phi(o, pp)} = \gamma \lambda_k \frac{\partial h_k(e, f)}{\partial s_\phi(o, pp)}$$
$$= \gamma \lambda_k \mathcal{N}(o, pp, e, f)$$

1: **function** TRAINSGD($\mathcal{D}$, μ)
2: $t \leftarrow 0$
3: **for all** $(f^{(i)}, e^{(i)})$ in $\mathcal{D}$ **do**
4: xBLEU $= 0$ ▷ Compute xBLEU
5: **for all** e in $\mathcal{E}(f^{(i)})$ **do**
6: wBLEU $\leftarrow p_{\theta,\phi_t}(e|f)$ sBLEU$(e, e^{(i)})$
7: xBLEU $\leftarrow$ xBLEU + wBLEU
8: **end for**
9: **for all** e in $\mathcal{E}(f^{(i)})$ **do**
10: $D =$ sBLEU$(e, e^{(i)}) -$ xBLEU
11: **for all** o, pp in $\langle e, f^{(i)} \rangle$ **do**
12: $N = \mathcal{N}(o, pp, e, f)$
13: $\delta_{o,pp} = p_{\theta,\phi_t}(e|f^{(i)}) \gamma \lambda_k N D$
14: $\phi_{t+1} = \phi_t - \mu \delta_{o,pp} u(o, pp))$
15: **end for**
16: **end for**
17: $t \leftarrow t + 1$
18: **end for**
19: **end function**

Figure 2: Algorithm for computing the expected BLEU loss with SGD updates (Eq. 4) based on training data $\mathcal{D}$ and learning rate μ.

where $m + 1 \leq k \leq m + j$ and $\mathcal{N}(o, pp, e, f)$ is the number of times pp with orientation o occurs in the current sentence pair.

This simplifies the error term to:

$$\delta_{o,pp} = \sum_{e \in \mathcal{E}(f)} p_{\theta,\phi}(e|f) \gamma \lambda_k \mathcal{N}(o, pp, e, f) U(\phi, e) \quad (8)$$

where λ_k is the weight of the dense feature summarizing orientation o in the log-linear model. We use Eq. 8 in a simple algorithm to train our model (Figure 2). Our SGD trainer uses a mini-batch size of a single sentence (§7) which entails all hypothesis in the n-best list for this sentence and the parameters are updated after each mini-batch.

6 Feature Sets

Our features are inspired by Cherry (2013) who bases his features on the local phrase-pair $pp = \langle \bar{e}, \bar{f} \rangle$ as well as the top stack of the shift reduce parser of the baseline hierarchical ordering model. We experiment with these variants and extensions:

- SparseHRMLocal: This feature set is exclusively based on the local phrase-pair and

consists of features over the first and last word of both the source and target phrase.[5] We use four different word representations: The word identity itself, but only for the 80 most common source and target language words. The three other word representations are based on Brown clustering with either 20, 50 or 80 classes (Brown et al., 1992). There is one feature for every orientation type.

- SparseHRM: The main feature set of Cherry (2013). This is an extension of SparseHRM-Local adding features based on the first and last word of both the source and the target of the hierarchical block at the top of the stack. There are also features based on the source words *in-between* the current phrase and the hierarchical block at the top of the stack.

- SparseHRM+UncommonWords: This set is identical to SparseHRM, except that word-identity features are not restricted to the 80 most frequent words, but can be instantiated for all words, regardless of frequency.

- SparseHRM+BiPhrases: This augments SparseHRM by phrase-identity features resulting in millions of instances compared to only a few thousand for SparseHRM. We add three features for each possible phrase pair: the source phrase, the target phrase, and the whole phrase pair.

The baseline hierarchical lexicalized reordering model is most similar to SparseHRM+BiPhrases feature set since both have parameters for phrase, orientation pairs.[6] The feature set closest to Cherry (2013) is SparseHRM. However, while Cherry had to severely restrict his features for batch lattice MIRA-based training, our maximum expected BLEU approach can handle millions of features.

7 Experiments

Baseline. We experiment with a phrase-based system similar to Moses (Koehn et al., 2007), scoring translations by a set of common features including maximum likelihood estimates of source given target phrases $p_{MLE}(e|f)$ and vice versa, $p_{MLE}(f|e)$, lexically weighted estimates $p_{LW}(e|f)$ and $p_{LW}(f|e)$, word and phrase-penalties, as well as a linear distortion feature. The baseline uses a hierarchical reordering model with five orientation types, including monotone and swap, described in §2, as well as two discontinuous orientations, distinguishing if the previous phrase is to the left or right of the current phrase. Finally, monotone global indicates that all previous phrases can be combined into a single hierarchical block. The baseline includes a modified Kneser-Ney word-based language model trained on the target-side of the parallel data, which is described below. Log-linear weights are estimated with MERT (Och, 2003). We regard the 1-best output of the phrase-based decoder with the hierarchical reordering model as the baseline accuracy.

Evaluation. We use training and test data from the WMT 2012 campaign and report results on French-English and German-English translation (Callison-Burch et al., 2012). Translation models are estimated on 102M words of parallel data for French-English and 91M words for German-English; between 7.5-8.2M words are newswire, depending on the language pair, and the remainder are parliamentary proceedings. All discriminative reordering models are trained on the newswire subset since we found this portion of the data to be most useful in initial experiments. We evaluate on six newswire domain test sets from 2008, 2010 to 2013 as well as the 2010 system combination test set containing between 2034 to 3003 sentences. Log-linear weights are estimated on the 2009 data set comprising 2525 sentences. We evaluate using BLEU with a single reference.

Discriminative Reordering Model. We use 100-best lists generated by the phrase-based decoder to train the discriminative reordering model. The n-best lists are generated by ten systems, each trained on 90% of the available data in order to decode the remaining 10%. The purpose of this procedure is to avoid a bias introduced by generating n-best lists for sentences on which the translation model was previously trained.[7] Unless otherwise

[5] Phrase-local features allow pre-computation which results in significant speed-ups at run-time. Cherry (2013) shows that local features are responsible for most of his gains.

[6] Although, our model is likely to learn significantly fewer parameters since many phrase, orientation pairs will only be seen in the word-aligned data but not in actual machine translation output.

[7] Later, we found that the bias has only a negligible effect on end-to-end accuracy since we obtained very similar results when decoding with a system trained on all data. This setting increased the training data BLEU score from 27.5 to 37.8. We used a maximum source and target phrase length of 7 words.

	dev	2008	2010	sc2010	2011	2012	2013	AllTest	FeatTypes
noRM	23.37	20.18	24.24	24.18	24.83	24.23	24.85	23.93	-
HRM (baseline)	24.11	20.85	24.92	24.83	25.68	25.11	25.76	24.72	-
SparseHRMLocal	25.24	21.26	25.99	25.93	26.98	26.34	26.77	25.77	4,407
SparseHRM	25.29	21.43	26.17	26.14	26.99	26.63	27.01	25.95	9,463
+UncommonWords	25.32	21.76	26.30	26.29	27.15	26.77	27.18	26.12	897,537
+BiPhrases	25.46	21.67	26.19	26.19	27.55	27.07	27.41	26.26	3,043,053

Table 1: French-English results of expected BLEU trained sparse reordering models compared to no reordering model at all (noRM) and the likelihood trained baseline hierarchical reordering model (HRM) on WMT test sets; sc2010 is the 2010 system combination test set. FeatTypes is the number of different types and AllTest is the average BLEU score over all the test sets, weighted by corpus size. All results for our sparse reordering models include a likelihood-trained hierarchical reordering model.

	dev	2008	2010	sc2010	2011	2012	2013	AllTest	FeatTypes
noRM	18.54	19.28	20.14	20.01	18.90	18.87	21.60	19.81	-
HRM (baseline)	19.35	19.96	20.87	20.66	19.60	19.80	22.48	20.58	-
SparseHRMLocal	19.89	19.86	21.11	20.84	20.04	20.21	22.93	20.88	4,410
SparseHRM	19.83	20.27	21.26	21.05	20.22	20.44	23.17	21.11	9,477
+UncommonWords	20.06	20.35	21.45	21.31	20.28	20.55	23.30	21.24	1,136,248
+BiPhrases	20.09	20.33	21.62	21.47	20.66	20.75	23.27	21.40	3,640,693

Table 2: German-English results of expected BLEU trained sparse reordering models (cf. Table 1).

mentioned, we train our reordering model on the news portion of the parallel data, corresponding to 136K-150K sentences, depending on the language pair. We tuned the various hyper-parameters on a held-out set, including the learning rate, for which we found a simple setting of 0.1 to be useful. To prevent overfitting, we experimented with ℓ_2 regularization, but found that it did not improve test accuracy. We also tuned the probability scaling parameter γ (Eq. 6) but found $\gamma = 1$ to be very good among other settings. We evaluate the performance on a held-out validation set during training and stop whenever the objective changes less than a factor of 0.0003. For our **PRO experiments**, we tuned three hyper-parameters controlling ℓ_2 regularization, sentence-level BLEU smoothing, and length. The latter is important to eliminate PRO's tendency to produce too short translations (Nakov et al., 2012).

7.1 Scaling the Feature Set

We first compare our baseline, a likelihood trained hierarchical reordering model (HRM; Galley & Manning, 2008), to various expected BLEU trained models, starting with SparseHRMLocal, inspired by Cherry (2013) and compare it to SparseHRM+BiPhrases, a set that is three orders of magnitudes larger.

Our results on French-English translation (Table 1) and German-English translation (Table 2) show that the expected BLEU trained models scale to millions of features and that we outperform the baseline by up to 2.0 BLEU on newstest2012 for French-English and by up to 1.1 BLEU on newstest2011 for German-English.[8] Increasing the size of the feature set improves accuracy across the board: The average accuracy over all test sets improves from 1.0 BLEU for the most basic feature set to 1.5 BLEU for the largest feature set on French-English and from 0.3 BLEU to 0.8 BLEU on German-English.[9] The most comparable setting to Cherry (2013) is the feature set SparseHRM, which we outperform by up to 0.5 BLEU on French-English and by 0.3 BLEU on average on both language pairs, demonstrating the benefit of being able to effectively train large feature sets. Furthermore, the increase in the number of features does not affect runtime, since most

[8] Different to the setups of Galley & Manning (2008) and Cherry (2013) our WMT evaluation framework uses only one instead of four references, which makes our BLEU score improvements not directly comparable.

[9] We attribute smaller improvements on German-English to the low distortion limit of only six words of our system and the more difficult reordering patterns when translating from German which may require more elaborate features.

features can be pre-computed and stored in the phrase-table, only requiring a constant time table-lookup, similar to traditional reordering models.

Another appeal of our approach is that training is very fast given a set of n-best lists for the training data. The SparseHRM model with 4,407 features is trained in only 26 minutes, while the SparseHRM+BiPhrases model with over three million parameters can be trained in just over two hours (136K sentences and 100 epochs in both cases). We attribute this to the training regime (§4), which does not iteratively re-decode the training data for expected BLEU training.[10]

7.2 Varying Training Set Size

Previous work on sparse reordering models was restricted to small data sets (Cherry, 2013) due to the limited ability of standard machine translation optimizers to handle more than a few thousand sentences. In particular, recent attempts to scale the margin-infused relaxation algorithm, a variation which was also used by Cherry (2013), to larger data sets showed that more data does not necessarily help to improve test set accuracy for large feature sets (Eidelman et al., 2013).

In the next set of experiments, we shed light on the advantage of training discriminative reordering models with expected BLEU on large training sets. Specifically, we start off by estimating a reordering model on only 2,000 sentences, similar to the size of the development set used by Cherry (2013), and incrementally increase the amount of training data to nearly three hundred thousand sentences. To avoid overfitting to small data sets we experiment with our most basic feature set SparseHRM-Local, comprising of just over 4,400 types.

For this experiment only, we measure accuracy in a *re-ranking* framework for faster experimentation where we use the 100-best output of the baseline system relying on a likelihood-based hierarchical reordering model. We re-estimate the log-linear weights by running a further iteration of MERT on the n-best list of the development set which is augmented by scores corresponding to the discriminative reordering model. The weights of those features are initially set to one and we use 20 random restarts for MERT. At test time we rescore the 100-best list of the test set using the new set of log-linear weights learned previously.

[10] We would expect better accuracy when iteratively decoding the training data but did not do so in this study for efficiency reasons.

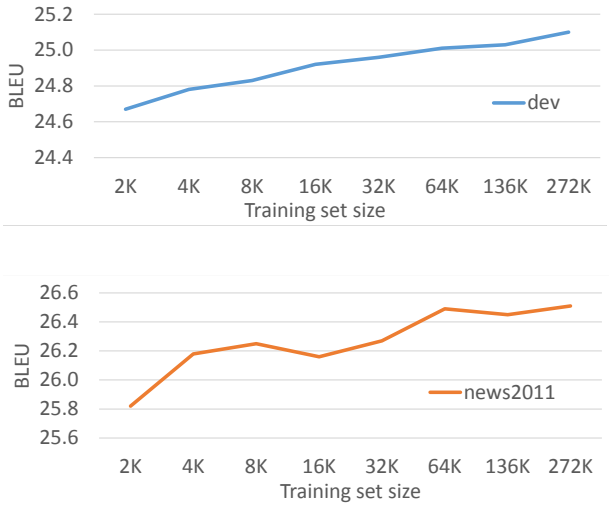


Figure 3: Effect of increasing the training set size from 2,000 to 272,000 sentences measured on the dev set (top) and news2011 (bottom) in an n-best list rescoring setting.

Figure 3 confirms that more training data increases accuracy and that the best model requires a substantially larger amount of training data than what is typically used for maximum BLEU training. We expect an even steeper curve for larger feature sets where more parameters need to be estimated and where the amount of training data is likely to have an even larger effect.

7.3 Likelihood versus BLEU Optimization

Previous research has shown that directly training a reordering model for BLEU can vastly outperform a likelihood trained maximum entropy reordering model (Cherry, 2013). However, the two approaches do not only differ in the objectives used, but also in the type of training data. The maximum entropy reordering model is trained on a word-aligned corpus, trying to learn human reordering patterns, whereas the sparse reordering model is trained on machine translation output, trying to learn from the mistakes made by the actual system. It is therefore not clear how much either one contributes to good accuracy.

Our next experiment teases those two aspects apart and clearly shows the effect of the objective function. Specifically, we compare the traditionally used conditional log-likelihood (CLL) objective to expected BLEU on the French-English translation task in a small feature condition (SparseHRM) of about 9K features and

	dev	2008	2010	sc2010	2011	2012	2013	AllTest
noRM	23.37	20.18	24.24	24.18	24.83	24.23	24.85	23.93
HRM (baseline)	24.11	20.85	24.92	24.83	25.68	25.11	25.76	24.72
SparseHRM (CLL)	24.28	21.02	25.11	25.10	25.92	25.24	25.76	24.88
SparseHRM (xBLEU)	25.29	21.43	26.17	26.14	26.99	26.63	27.01	25.95
SparseHRM+BiPhrases (CLL)	24.42	21.17	25.12	25.00	25.86	25.36	26.18	24.98
SparseHRM+BiPhrases (xBLEU)	25.46	21.67	26.19	26.19	27.55	27.07	27.41	26.26

Table 3: French-English results comparing the baseline hierarchical reordering model (HRM) to sparse reordering model trained towards conditional log-likelihood (CLL) and expected BLEU (xBLEU).

	dev	2008	2010	sc2010	2011	2012	2013	AllTest
PRO	24.05	20.90	25.42	25.28	25.79	25.09	26.07	24.94
xBLEU	25.24	21.26	25.99	25.93	26.98	26.34	26.77	25.77

Table 4: French-English results on the SparseHRMLocal feature set when when trained with pair-wise ranked optimization (PRO) and expected BLEU (xBLEU).

a large feature setting of over 3M features (SparseHRM+BiPhrases). In the CLL setting, we maximize the likelihood of the hypothesis with the highest BLEU score in the n-best list of each training sentence.

Our results (Table 3) show that CLL training achieves only a fraction of the gains yielded by the expected BLEU objective. For SparseHRM, CLL improves the baseline by less than 0.2 BLEU on average across all test sets, whereas expected BLEU achieves 1.2 BLEU. Increasing the number of features to 3M (SparseHRM+BiPhrases) results in a slightly better average gain of 0.3 BLEU for CLL but but expected BLEU still achieves a much higher improvement of 1.5 BLEU. Because our gains with likelihood training are similar to what Cherry (2013) reported for his maximum entropy model, we conclude that the objective function is the most important factor to achieving good accuracy.

7.4 Comparison to PRO

In our final experiment we compare expected BLEU training to pair-wise ranked optimization (PRO), a popular off the shelf trainer for machine translation models with large feature sets (Hopkins and May, 2011).[11] Previous work has shown that PRO does not scale to truly large feature sets with millions of types (Yu et al., 2013) and we therefore restrict ourselves to our smallest set (SparseHRMLocal) of just over 4.4K features. We train PRO on the development set comprising of 2,525 sentences, a setup that is commonly used by standard machine translation optimizers. In this setting, PRO directly learns weights for the baseline features (§7) as well as the 4.4K indicator features corresponding to the sparse reordering model. For expected BLEU training we use the full 136K sentences from the training data. The results (Table 4) demonstrate that expected BLEU outperforms a typical setup commonly used to train large feature sets.

8 Conclusion and Future Work

The expected BLEU objective is a simple and effective approach to train large-scale discriminative reordering models. We have demonstrated that it scales to millions of features, which is orders of magnitudes larger than other modern machine translation optimizers can currently handle.

Empirically, our sparse reordering model improves machine translation accuracy across the board, outperforming a strong hierarchical lexicalized reordering model by up to 2.0 BLEU on a French to English WMT2012 setup, where the baseline was trained on over two million sentence pairs. We have shown that scaling to large training sets is crucial to good performance and that the best performance is reached when hundreds of thousands of training sentences are used. Furthermore, we demonstrate that task-specific training towards expected BLEU is much more effective than optimizing conditional log-likelihood as

[11] MIRA is another popular optimizer but as previously mentioned, even the best publicly available implementation does not scale to large training sets (Eidelman et al., 2013).

is usually done. We attribute this to the fact that likelihood is a strict zero-one loss that does not assign credit to partially correct solutions, whereas expected BLEU does.

In future work we plan to extend expected BLEU training to lattices and to evaluate the effect of estimating weights for the dense baseline features as well. Our current training procedure (Gao and He, 2013; Gao et al., 2014) decodes the training data only once. In future work, we would like to compare this to repeated decoding as done by conventional optimization methods as well as other large-scale discriminative training approaches (Yu et al., 2013). We expect this to yield additional accuracy gains.

Acknowledgements

We would like to thank Arul Menezes and Xiaodong He for helpful discussion related to this work and the three anonymous reviewers for their comments.

References

Léon Bottou. 2004. Stochastic learning. In Olivier Bousquet and Ulrike von Luxburg, editors, *Advanced Lectures in Machine Learning*, Lecture Notes in Artificial Intelligence, pages 146–168. Springer Verlag, Berlin.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, Dec.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proc. of WMT*, pages 10–51. Association for Computational Linguistics, June.

Colin Cherry. 2013. Improved Reordering for Phrase-Based Translation using Sparse Features. In *Proc. of NAACL*, pages 9–14. Association for Computational Linguistics, June.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 New Features for Statistical Machine Translation. In *Proc. of NAACL*, pages 218–226. Association for Computational Linguistics, June.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Vladimir Eidelman, Ke Wu, Ferhan Ture1, Philip Resnik, and Jimmy Lin. 2013. Mr. MIRA: Open-Source Large-Margin Structured Learning on MapReduce. In *Proc. of ACL*, pages 199–204. Association for Computational Linguistics, August.

Michel Galley and Christopher D. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. In *Proc. of EMNLP*, pages 848–856.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. of HLT-NAACL*, pages 273–280, Boston, MA, USA, May.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable Inference and Training of Context-Rich Syntactic Translation Models. In *Proc. of ACL*, pages 961–968, Sydney, Australia, June.

Jianfeng Gao and Xiaodong He. 2013. Training MRF-Based Phrase Translation Models using Gradient Ascent. In *Proc. of NAACL-HLT*, pages 450–459. Association for Computational Linguistics, June.

Jianfeng Gao, Xiaodong He, Scott Wen tau Yih, and Li Deng. 2014. Learning Continuous Phrase Representations for Translation Modeling. In *Proc. of ACL*. Association for Computational Linguistics, June.

Spence Green, Daniel Cer, and Christopher Manning. 2014. An Empirical Comparison of Features and Tuning for Phrase-based Machine Translation. In *Proc. of WMT*. Association for Computational Linguistics, June.

Xiaodong He and Li Deng. 2012. Maximum Expected BLEU Training of Phrase and Lexicon Translation Models. In *Proc. of ACL*, pages 8–14. Association for Computational Linguistics, July.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proc. of EMNLP*. Association for Computational Linguistics, July.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proc. of HLT-NAACL*, pages 127–133, Edmonton, Canada, May.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *Proc. of IWSLT*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proc. of ACL Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, Jun.

Percy Liang, Alexandre Bouchard-Côté, Ben Taskar, and Dan Klein. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of ACL-COLING*, pages 761–768, Jul.

Preslav Nakov, Francisco Guzman, and Stephan Vogel. 2012. Optimizing for Sentence-Level BLEU+1 Yields Short Translations. In *Proc. of COLING*. Association for Computational Linguistics.

Vinh Van Nguyen, Akira Shimazu, Minh Le Nguyen, and Thai Phuong Nguyen. 2009. Improving A Lexicalized Hierarchical Reordering Model Using Maximum Entropy. In *MT Summit XII*. Association for Computational Linguistics, August.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to machine translation. *Computational Linguistics*, 30(4):417–449, June.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of ACL*, pages 160–167, Sapporo, Japan, July.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*, pages 311–318, Philadelphia, PA, USA, Jul.

Antti-Veikko I Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2010. BBN System Description for WMT10 System Combination Task. In *Proc. of WMT*, pages 321–326. Association for Computational Linguistics, July.

Antti-Veikko I Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2011. Expected BLEU Training for Graphs: BBN System Description for WMT11 System Combination Task. In *Proc. of WMT*, pages 159–165. Association for Computational Linguistics, July.

Christoph Tillmann. 2003. A Unigram Orientation Model for Statistical Machine Translation. In *Proc. of NAACL*, pages 106–108. Association for Computational Linguistics, June.

Roy W. Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *Proc. of EMNLP*, pages 620–629. Association for Computational Linguistics, October.

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

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proc. of ACL-COLING*, pages 521–528, Sydney, Jul.

Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. 2013. Max-Violation Perceptron and Forced Decoding for Scalable MT Training. In *Proc. of EMNLP*, pages 1112–1123. Association for Computational Linguistics, October.

Confidence-based Rewriting of Machine Translation Output

Benjamin Marie
LIMSI-CNRS, Orsay, France
Lingua et Machina, Le Chesnay, France
benjamin.marie@limsi.fr

Aurélien Max
LIMSI-CNRS, Orsay, France
Univ. Paris Sud, Orsay, France
aurelien.max@limsi.fr

Abstract

Numerous works in Statistical Machine Translation (SMT) have attempted to identify better translation hypotheses obtained by an initial decoding using an improved, but more costly scoring function. In this work, we introduce an approach that takes the hypotheses produced by a state-of-the-art, reranked phrase-based SMT system, and explores new parts of the search space by applying rewriting rules selected on the basis of posterior phrase-level confidence. In the medical domain, we obtain a 1.9 BLEU improvement over a reranked baseline exploiting the same scoring function, corresponding to a 5.4 BLEU improvement over the original `Moses` baseline. We show that if an indication of which phrases require rewriting is provided, our automatic rewriting procedure yields an additional improvement of 1.5 BLEU. Various analyses, including a manual error analysis, further illustrate the good performance and potential for improvement of our approach in spite of its simplicity.

1 Introduction

The standard configuration of modern phrase-based Statistical Machine Translation (SMT) (Koehn et al., 2003) systems can produce very acceptable results on some tasks. However, early integration of better features to guide the search for the best hypothesis can result in significant improvements, an expression of the complexity of modeling translation quality. For instance, improvements have been obtained by integrating features into decoding that better model semantic coherence at the sentence level (Hasan and Ney, 2009) or syntactic well-formedness (Schwartz et al., 2011). However, early use of such complex features typically comes at a high computational cost. Moreover, some informative features require or are better computed when complete translation hypotheses are available. This is addressed in numerous works on reranking of the highest scored sub-space of hypotheses, on so-called n-best lists (Och et al., 2004; Zhang et al., 2006; Carter and Monz, 2011) or output lattices (Schwenk et al., 2006; Blackwood et al., 2010), where many works specifically target the inclusion of better language modelling capabilities, a well-known weakness of current automatic generation approaches (Knight, 2007).

Another way to improve translation *a posteriori* can be done by rewriting initial hypotheses, for instance in a greedy fashion by including new models (Langlais et al., 2007; Hardmeier et al., 2012), or by specifically modeling a task of automatic post-editing targeting a specific system (Simard et al., 2007; Dugast et al., 2007). While such automatic post-editing may seem to be too limited, notably because of the limited initial diversity considered and the fact that it may be in some instances agnostic to the internals of the initial system, it has been shown to potentially improve accuracy of the new translation hypotheses (Parton et al., 2012) and to offer very high oracle performance (Marie and Max, 2013).

However, an important issue for such approaches is their capacity to only rewrite *incorrect* parts of the translation hypotheses and to use appropriate replacement candidates. Many works have tackled the issue of word to *n*-gram confidence estimation in SMT output (Zens and Ney, 2006; Ueffing and Ney, 2007; Bach et al., 2011; de Gispert et al., 2013), and some attempts have been made to exploit confidence estimates for lattice rescoring (Blackwood et al., 2010) or *n*-best reranking (Bach et al., 2011; Luong et al., 2014b).

In this work, we present an approach in which

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1261–1272,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

new complete hypotheses are produced by rewriting existing hypotheses, and are scored using complex models that could not be used during the initial decoding. We will use as competitive baselines systems that rerank the output of an initial decoder using the complete set of available features, and will show that we manage to improve their translation. The difference between our approach and the reranking baseline lies in the manner in which we expand our training data, as well as in our use of high-confidence rewritings to obtain new translation hypotheses. Importantly, this work will only exploit simple confidence estimates corresponding to phrase-based posteriors, which do not require that large sets of human-annotated data be available as in other works (Bach et al., 2011; Luong et al., 2014b).

The remainder of this paper is organized as follows. Section 2 is devoted to the description of our approach, with details on our rewriting approach (2.1), additional features (2.2), rewriting phrase table (2.3), and training examples (2.4). Section 3 presents experiments. We first describe our experimental setup (3.1) and our baseline systems (3.2). We then report results when naive rewriting is performed and then with confidence-based rewriting (3.3). We next devote a significant part of the paper in section 4 to report further results and analyses: an analysis of the performance of our system depending on the quality of initial hypotheses (4.1); a semi-oracle experiment where correct phrases are known (4.2); an oracle experiment where only correct rewriting decisions are made (4.3); a manual error analysis of the main configurations studied in this work (4.4); and, finally, a study of the performance of our approach on a more difficult translation task (4.5). Related work is discussed in section 5 and we conclude and introduce our future work in section 6.

2 Description of the approach

2.1 Rewriting of translation hypotheses

Langlais *et al* (2007) proposed a greedy search procedure to improve translations by reusing the same translation table and scoring function that were used during an initial phrase-based decoding. In our approach, we rewrite hypotheses by using the same greedy search algorithm, adding more complex models and using the most-confident bi-phrases according to the initial decoder's search space. To select the hypothesis to rewrite for each sentence, we produce a n-best list of the initial decoder and rerank this list with a new, better informed scoring function (see section 2.2). The one-best hypothesis obtained after reranking is then rewritten by our system (denoted as `rewriter`). In this way, we ensure that the hypothesis that was rewritten had been so far the best one according to the initial decoding best subspace and the new models used.

At each iteration, new hypotheses are obtained from a current hypothesis by applying one rewriting operation on bi-phrases. The set of all new hypotheses is called the neighborhood of the current hypothesis. Focusing in this work on *local rewriting*, we used the following set of operations (N denotes the number of bi-phrases, T the maximum number of entries per source phrase in a rewriting phrase table (see 2.3), and S the average number of tokens per source phrase)[1]:

1. `replace` ($\mathcal{O}(N.T)$): replaces the translation of a source phrase with another translation from the rewriting phrase table;

2. `split` ($\mathcal{O}(N.S.T^2)$): splits a source phrase into all possible sets of two (contiguous) phrases, and uses `replace` on each of the resulting phrases;

3. `merge` ($\mathcal{O}(T.N)$): merges two contiguous source phrases and uses `replace` on the resulting new phrase.

This rewriting algorithm is described in pseudo-code in Algorithm 1.

Algorithm 1 `rewriter` Algorithm

Require: *source* a sentence to translate

```
nbestList ← TRANSLATE(source)
oneBest ← RERANK(nbestList)
sCurrent ← GET_SCORE(oneBest)
loop
   hypothesesSet ← NEIGHBORHOOD(oneBest)
   newOneBest ← RANK(hypothesesSet)
   s ← GET_SCORE(newOneBest)
   if s ≤ sCurrent then
      return oneBest
   else
      oneBest ← newOneBest
      sCurrent ← s
   end if
end loop
```

[1] Complexity is expressed in terms of the maximum number of hypotheses that will be considered given some hypothesis to rewrite.

The produced hypotheses are then ranked according to a new, better informed scoring function (see 2.2). At the next iteration, the hypothesis now ranked at the top of the list is rewritten, and search terminates when no better hypothesis is found.

Such a greedy search has several obvious limitations, in particular it can only perform a limited exploration of the search space, a situation that can be improved by using a beam (see Section 3.3). However, associated with a small and precise rewriting phrase table, this approach only visits small numbers of more-confident hypotheses, which is a critical property given the cost of computing the new scoring function used.

2.2 Reranking and features

The rerankings of the hypotheses sets describe in this work are all performed with `kb-mira` (Cherry and Foster, 2012) using the initial features set of the decoder in conjunction with the following additional features:[2]

- **`SOUL` models**: `SOUL` models are structured output layer neural network language models (LMs) which have been shown to be useful in reranking tasks, for instance for WMT evaluations (Allauzen et al., 2013; Pécheux et al., 2014). `SOUL` scoring being too costly to be integrated during decoding, it fits perfectly the `reranker` scenario, which furthermore enables to use larger contexts for *n*-grams. We used both monolingual (Le et al., 2011) and bilingual (Le et al., 2012) `SOUL` 10-gram models, which were trained on the WMT'12 data.

- **POS language model**: part-of-speech (POS) LMs have been shown to yield improvements in *n*-best list reranking (Carter and Monz, 2011). In this work, we trained a 6-gram POS LM using Witten-Bell smoothing.

- **IBM1** : the IBM1 scores ($p(e|f)$ and $p(f|e)$) of the complete hypothesis (Och et al., 2004).

- **phrase-based confidence score** : bi-phrases are associated to a posterior probability, inspired from *n*-gram posterior probability estimation as defined in (de Gispert et al., 2013). Let E be the set of all hypotheses in the space of translation hypotheses defined by the n-best list used for source sentence f, and E_α be the subset of E such that word alignments in sentence pairs (e', f), $\forall e' \in E_\alpha$, allow us to extract bi-phrase α. Let also $H(e, f)$ be the score assigned by a baseline decoder (denoted as `1-pass Moses` henceforth) to sentence pair (e, f). We use the following posterior probability for α:

$$P(\alpha|F) = \frac{\sum_{e' \in E_\alpha} exp(H(e', f))}{\sum_{e'' \in E} exp(H(e'', f))} \quad (1)$$

Then, the logarithms of each phrase's confidence score are summed to use as a confidence score for the complete hypothesis.

[2]Note that we did not try to explore the independant contribution of each feature in this work.

2.3 Rewriting phrase table

Taking the whole translation table of the decoder as a rewriting phrase table to perform the greedy search produces very large neighborhoods that `rewriter` cannot handle due to the cost of the models that have to be computed. We tried two different approaches to extract a rewriting phrase table from the translation table of the system.

We first tried a naive approach where the rewriting phrase table of `rewriter` for the test set uses the phrase table of `1-pass Moses`, filtered to keep the k best entries according to the direct translation model. We denote such a configuration `rpt`k`pef`.

Our second approach consists in extracting the rewriting phrase table containing bi-phrases that were the most probable according to the set of all models used in `1-pass Moses`. Selection of bi-phrases for each sentence is done in a binary fashion, depending on their presence in k-best lists of `1-pass Moses` for a given value of k. This configuration will be denoted `conf`k.

2.4 Training examples

We tried several sets of examples to train the ranker of `rewriter`. We used the 1,000-best list of the development set produced by `1-pass Moses` during its tuning. In other configurations we mixed *a*) the neighborhood of the `reranker` n-best hypotheses computed by our system on the development set using a rewriting phrase table containing the bi-phrases found in the k-best list produced by `1-pass Moses`; and *b*) the neighborhood of the one-best hypotheses of `reranker` using a rewriting phrase table containing the 10-best translations from the `1-pass Moses` translation table according to the direct translation

model. Both neighborhoods are produced by a single iteration of rewriter. We denote respectively these sets of hypotheses n-bestNeigh and 10PefNeigh. Our intuition behind the constitution of these training sets is that the ranker of rewriter needs, in order to perform well, training examples that will be similar to hypotheses that it actually generates.

3 Experiments

3.1 Experimental setup

We used two datasets from two different domains: the data provided for the WMT'14 medical translation task[3] (Medical) and a smaller task using the TED talks[4] (TED Talks) data of the IWSLT evaluation campaigns. For the Medical task we used only the English to French translation direction, and both translation directions, English to French and French to English, for the TED Talks task. In this work, the main part of our experiments uses Medical, and TED Talks will be used at a later stage to study a lower-quality situation (cf. 4.5). For the Medical task, initial decodings were produced using a LM trained on all WMT'14 monolingual and bilingual medical data, while for the TED Talks task we used a much larger LM trained on all the data provided for WMT'13[5]. Both are 4-gram LMs estimated with Kneser-Ney smoothing (Chen and Goodman, 1998). For the 6-gram POS LMs used (see 2.2), we used the same data as used for the token-based LM for Medical, and the concatenation of the News Commentaries and Europarl sub-parts of the WMT'13 data for TED Talks. Table 1 provides relevant statistics about the data used.

Tasks	Corpus	Sentences	Tokens (en-fr)
Medical	train	4.9M	78M - 91M
	dev	500	10k - 12k
	test	1,000	21k - 26k
	LM		- 146M
TED Talks	train	107 758	2M - 2.2M
	dev	934	20k - 20k
	test	1,664	31k - 34k
	LM		6B - 2.5B

Table 1: Corpora used in this work.

[3] http://www.statmt.org/wmt14/medical-task/

[4] https://wit3.fbk.eu/mt.php?release=2013-01

[5] http://www.statmt.org/wmt13

We first built a state-of-the-art phrase-based SMT system using Moses (Koehn et al., 2003) with standard settings. We tuned its parameters towards BLEU (Papineni et al., 2002) on the tuning dataset using the kb-mira implementation available in Moses with default parameters.

Our results will be compared using BLEU and TER (Snover et al., 2006) to *a*) the initial best translation produced by the Moses decoder (1-pass Moses) and *b*) the best translation obtained by reranking the 1,000-best list of 1-pass Moses (reranker). Since reranker implements a well-documented approach and uses types of features commonly used in reranking tasks we will consider it as our main baseline. It was trained using kb-mira on the 1,000-best of the development data decoded by 1-pass Moses.

In our experiments, rewriter rewrites the one-best hypothesis[6] produced by reranker using the operators Replace, Split and Merge as described in section 2.1.

3.2 Baseline results

Table 2 gives the results of the 1-pass Moses decoding for the Medical task and the reranking results of reranker applied to the 1-pass Moses 1,000-best list.

1-pass Moses obtains a score of 38.2 BLEU on the test set, which can be considered as a good baseline system.[7] reranker outperforms 1-pass Moses by 3.5 BLEU, indicating a strong performance of the features used on this task. In particular, SOUL is known to be a useful feature for reranking n-best lists on highly-inflected languages such as French. Note also that the SOUL models we used were trained on the WMT'12 monolingual and bilingual data and so were better informed than the models used during the 1-pass Moses decoding.[8] Moreover, as can be seen on Figure 1, the 1,000-best oracle reveals a large potential for improvement over the one-best (+12.4 BLEU). We further observe that the reranked list of reranker shows a much faster potential for translation improvement.

[6] Note that we will also provide results where a beam of *k*-best hypotheses are rewritten.

[7] Distribution of error types on a sub-part of the test set will be provided in section 4.4.

[8] However, SOUL considers only a small sample of the training data for training. For instance, the training of the French monolingual model used roughly only 1% (895K sentences) of all the WMT'12 data.

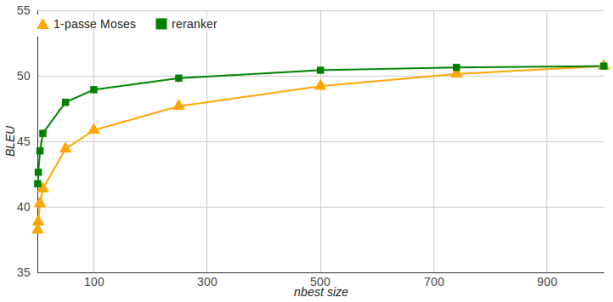


Figure 1: n-best list oracle for `1-pass Moses` and `reranker`

3.3 `rewriter` results

Results for the different rewriting phrase tables and training examples are given in Table 2. First, concerning the rewriting phrase table, for the k=5 (`rpt5pef`) and k=10 (`rpt10pef`) configurations[9] a decrease of 0.7-0.8 BLEU over `reranker` is obtained. This illustrates that naive rewritings applied on the test set cannot be used with our training regime to improve translation quality.

In the next experiments, we used a `conf`k rewriting table. Table 2[10] shows the results of `rewriter` when rewriting the one-best hypothesis from `reranker` for various values of k to define the k-best list from which the rewriting table is built. Various training sets are also considered in the table.

The `1-pass Moses` 1,000-best configuration reused the same set of hypotheses used to train `reranker`. For this configuration, `rewriter` loses 2.6 BLEU over `reranker` on the test set with `conf10k`. Of course, this training data set is of a quite different nature compared to the hypotheses built by `rewriter`.

In the 10pefNeigh training, the ranker is trained with the neighborhoods produced by the first iteration of `rewriter` on the development set with a rewriting phrase table containing only the k-best translations for each source phrase according to the direct translation model. This configuration improves over the previous one by 1.7 BLEU, but is still 0.9 BLEU below `reranker`. Adding the neighborhoods of the `reranker` n-best hypotheses produced with a `conf10k` rewriting phrase table to the training data does not improve over the previous situation for $n = 10$, but increasing n to 30 and then 50 produces strong improvements on the test set (resp. +1.4 and +1.6 BLEU). Considering a larger neighborhood obtained by rewriting the best $n = 90$ hypotheses does not yield further gains. We denote from now on `opti` our best configuration thus far, considering the performance on the development set and having the largest confidence-based rewriting phrase table considered.

Letting `rewriter` perform a beam search on the 10-best hypotheses of the test set, further gains are obtained, corresponding now to an improvement of +1.9 BLEU over our `reranker` baseline, or +5.4 BLEU over `1-pass Moses`.[11] Furthermore, although taking the bi-phrases from the 10,000-best is our best configuration, it is interesting to note that taking bi-phrases from the 10-best only already yields a moderate improvement of +0.6 BLEU over `reranker`. Figure 2a shows that up to $k = 10,000$ higher value of k to extract the rewriting phrase table increase the BLEU score on the test set. [12] We did not experiment with higher values of k, but plan to use the output lattice produced by `1-pass Moses` to compute efficiently posteriors for larger sets of bi-phrases (de Gispert et al., 2013).

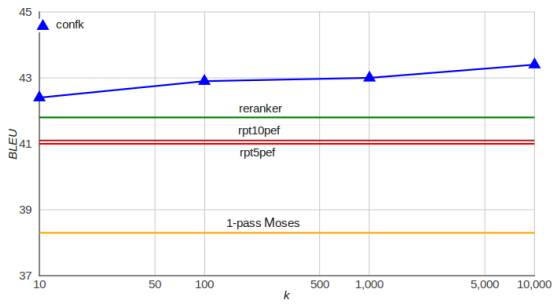
As illustrated on Figure 2b, `rewriter` mostly improves the BLEU score during the three first iterations and then converges at the ninth iteration. However, it is important to note that not all sentences are actually improved by our system. As illustrated on Figure 3a, `opti` improves 40.8% of the sentences of the test set but degrades 29.2% of them according to sentence-BLEU (Lin and Och, 2004). It is certainly the case that more informative confidence features may help idenfity more precisely which fragments of the translations should really undergo rewriting. We will investigate the exploitation of an oracle phrase-based confidence measure in Section 4.2.

[9] We did not experiment with higher values of k because of the computationnal cost of the features used by `reranker`. Indeed, adding more phrase translations increases the size of the neighborhoods corresponding to many additional n-grams to score by `SOUL`, the most expensive model.

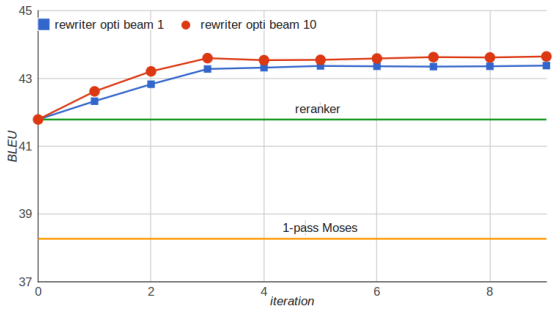
[10] In Table 2 the number of unique bi-phrases for the `rpt` rewriting phrase tables is computed by considering only source phrases appearing in the test set, for the n-best Neighborhood configurations we merged the phrase tables of each sentence into one and count just as one unique entry bi-phrases appearing several times.

[11] Using a beam becomes quickly prohibitive: using 12 threads, 25 mn vs. 3h were needed for the test set for the configurations of size 1 and 10, respectively.

[12] Note that even for $k = 10,000$ the computed neighborhoods are still quite small with an average of 116 hypotheses for each hypothesis to rewrite per iteration, against an average of 788 hypotheses for the `rpt10pef` configuration.



(a) Results of `rewriter` with `rpt5pef`, `rpt10pef` and different values of k for `confk`



(b) Iterations of `rewriter` on test with `opti` and two beam sizes : 1 and 10.

Figure 2: Performance of `rewriter` depending on the type of the rewriting phrase table and the number of iterations and beam sizes.

baseline				dev	test		
				BLEU	BLEU	TER	GOS BLEU
1-pass Moses				40.9	38.3	44.6	
reranker				44.1	41.8	41.6	
training data	rewriting phrase table	unique bi-phrases	beam size				
1-pass Moses 1 000-best	conf10k	38 455	1	44.1	39.2$_{(-2.6)}$	43.8$_{(+2.2)}$	58.7
10pefNeigh	conf10k	38 455	1	43.9	40.9$_{(-0.9)}$	41.2$_{(-0.4)}$	58.7
10-bestNeigh + 10pefNeigh	conf10k	38 455	1	43.8	40.9$_{(-0.9)}$	41.2$_{(-0.4)}$	58.7
30-bestNeigh + 10pefNeigh	conf10k	38 455	1	44.2	43.2$_{(+1.4)}$	40.6$_{(-1.0)}$	58.7
50-bestNeigh + 10pefNeigh	rpt5pef	85 530	1	44.5	41.0$_{(-0.8)}$	42.0$_{(+0.4)}$	50.6
=	rpt10pef	149 887	1	44.5	41.1$_{(-0.7)}$	42.1$_{(+0.5)}$	54.5
=	conf10	21 398	1	44.5	42.4$_{(+0.6)}$	41.0$_{(-0.6)}$	45.9
=	conf100	28 730	1	44.5	42.9$_{(+1.1)}$	40.8$_{(-0.8)}$	50.2
=	conf1k	33 929	1	44.5	43.0$_{(+1.2)}$	40.6$_{(-1.0)}$	53.3
= (opti)	conf10k	38 455	1	44.5	**43.4**$_{(+1.6)}$	**40.4**$_{(-1.2)}$	58.7
=	conf10k	38 455	10	44.5	**43.7**$_{(+1.9)}$	**40.1**$_{(-1.5)}$	59.6
90-bestNeigh + 10pefNeigh	conf10k	38 455	1	44.4	43.4$_{(+1.6)}$	40.4$_{(-1.2)}$	58.7

Table 2: Results on `Medical` for different training configurations, rewriting phrase tables and beam sizes. `opti` denotes our optimal configuration for `rewriter`.

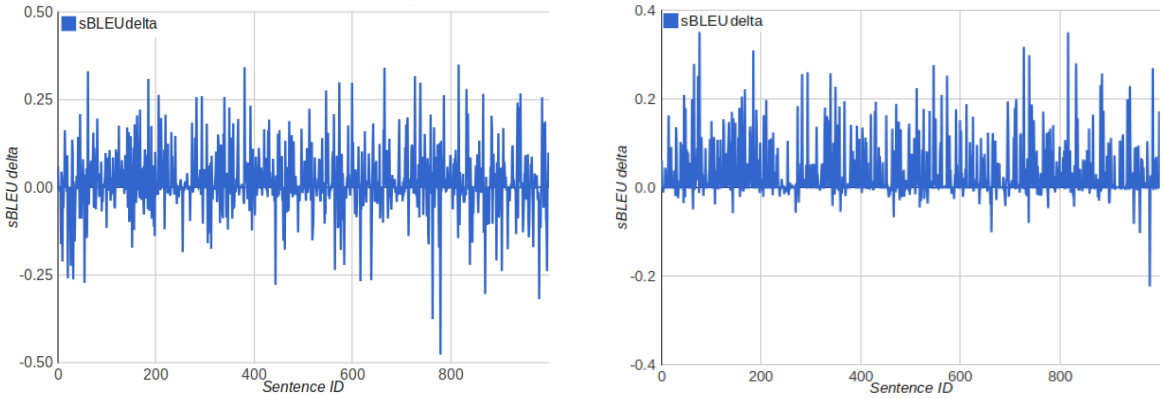
4 Analysis of confidence-based rewriting

4.1 Performance of `rewriter` depending on the quality of initial hypotheses

The first question we address in our analysis of `rewriter` is whether its performance depends on the difficulty of each individual sentence. As a proxy of sentence difficulty we used sentence-BLEU of `1-pass Moses`, and used it to divide the sentences of the test set into quartiles. Figure 4 shows that `reranker` improves more over `1-pass Moses` and that at the same time `rewriter` improves more over `reranker` as the sentences are more difficult. In particular, `rewriter` obtains a 8.6 BLEU improvement over `1-pass Moses` on the more difficult quartile, but only a 1.3 BLEU improvement on the least difficult quartile. We hypothesize that better performance may be achieved if adapting the training and rewriting of `rewriter` to sentences of varying quality, which may, for instance, be estimated with off-the-shelf estimators (Specia et al., 2013).

4.2 Semi-oracle experiments: rewriting only incorrect fragments

We observed in section 3.3 that our `opti` configuration, which obtains strong improvements in translation quality (as given by corpus-BLEU), in fact degrades (as given by sentence-BLEU) a significant proportion of sentences. To further analyze these results, we simulate a situation where oracle confidence information is available at the phrase-level: in particular, `rewriter` is prevented from rewriting bi-phrases whose target phrase appears exactly in the reference transla-



(a) automatic (b) semi-oracle

Figure 3: sBLEU delta, for each sentence, between the `reranker` one-best to rewrite and its automatic (3a) or semi-oracle (3b) rewriting computed by `rewriter` with the `opti` configuration.

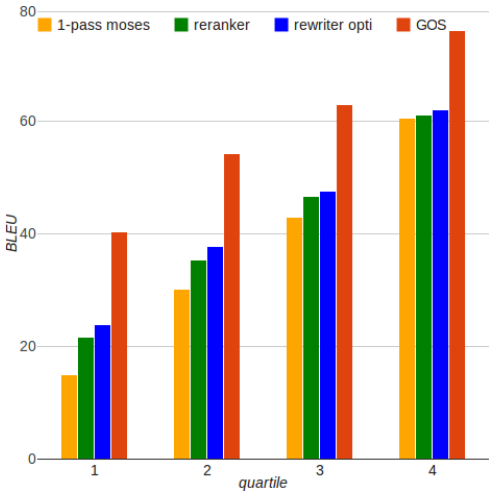


Figure 4: Source sentences were divided into quartiles according to sBLEU of the `1-pass Moses` system. For each quartile we reported the performance of `1-pass Moses`, `reranker`, `rewriter`, `GOS`.

tion.[13] Furthermore, this "freezing" of bi-phrases can be repeated after each iteration of `rewriter`.

Thus, we now have an oracle situation for choosing which source phrases may be rewritten, but the rest of the rewriting procedure is still fully automatic. Moreover, we purposefully did not adapt the training procedure to this new configuration, and reused `opti` as is. Results, reported in Table 3, indicate that an additional 1.5 BLEU is obtained from `opti`, or 3.1 BLEU from `reranker` and 6.6 BLEU from `1-pass Moses`. The use of a larger beam of size 10 did not improve those results any further. At the first iteration, `rewriter` "froze" approximatively 65.6% of the bi-phrases, and 70.5% at the last iteration, demonstrating the ability of `rewriter` to find good rewritings that match the reference translation. Looking at Figure 3b, we now find that, as expected, only a limited number of sentences are now degraded by `rewriter`. The large improvements obtained clearly underlines the important role that better confidence estimates could play in our framework.

System	test	
	BLEU	TER
`reranker`	41.8	41.6
`opti`	43.4	40.4
`semi-oracle`, beam 1	$44.9_{(+1.5)}$	$39.2_{(-1.2)}$
`semi-oracle`, beam 10	$44.9_{(+1.5)}$	$39.0_{(-1.4)}$

Table 3: Results for the semi-oracle using `opti`.

4.3 Oracle experiments: making only the correct decisions

We now turn to the situation where only rewritings that actually improve translation performance would be made. In practice, we use a simple solution: we resort to greedy oracle search (`GOS`) (Marie and Max, 2013), where sentence-BLEU is maximized using rewritings from the `opti` phrase table. At each iteration the rewriting in the neighborhood that maximizes sentence-BLEU is selected until convergence.

Results for this greedy search oracle appear in the last column of Table 2 and allow us to put in perspective the individual potential of the var-

[13] This is obviously not an optimal solution.

ious tested configurations. We can first notice that the rpt5pef phrase table allows the oracle to reach 50.6 BLEU, 8.1 BLEU below the oracle value obtained with conf10k, although rpt5pef contains twice as many bi-phrases. The same conclusion can be made about rpt10pef, which is 3.9 BLEU higher than rpt5pef but contains nearly twice as many bi-phrases. Finally, although conf10k contains approximatively four times fewer bi-phrases than rpt10pef, its oracle value is 4.2 BLEU higher. This points out the fact that conf10k is a lot more precise rewriting phrase table for the translations to rewrite, as well as the fact that rpt5pef and rpt10pef are much noisier and consequently difficult to use efficiently by our automatic rewriting procedure.

4.4 Manual error analysis

In the previous sections, we have shown that our automatic rewriting procedure can improve translation quality over both an initial Moses baseline, and a reranked baseline using the same features as our procedure. We have further shown in section 4.3 that much larger improvements could be obtained by using an oracle procedure.

We now focus on the four following configurations: 1-pass Moses, reranker, rewriter and GOS. Although this four configurations are well separated both in terms of BLEU and TER scores, it is informative to look more precisely into what makes their results different. We performed a small-scale manual error analysis of these four configurations. A French native speaker annotated 70 translation hypotheses using an error typology adapted from (Vilar et al., 2006).

Results of the manual error analysis are reported in Table 4. The most significant results are for the *disamb(iguation)* and *form* error types, the former being more related to translation accuracy, and the later to fluency. In both cases, we first observe a strong reduction of errors between 1-pass Moses and reranker, which demonstrates the positive impact of the features used on these levels. Then, another, similar reduction is obtained between reranker and rewriter, demonstrating that our reranking procedure manages to identify more precise and fluent hypotheses. Finally, a further reduction is found between rewriter and GOS, indicating that our proposed local, greedy rewriting can still be improved, notably by using more informative features and better confidence estimates.

The other types of error categories are less informative. We find no clear differences in error types attributable to style issues, which seem to be irrecoverable even for GOS. reranker and rewriter both improve on order-related errors over 1-pass Moses, but our local rewriting unsurprisingly did not fix any of these errors. Finally, reranker and rewriter decreased slightly the number of extra words from 1-pass Moses, while GOS sometimes artificially introduces extra words.

4.5 Lower-quality SMT experiments

We now turn to the question of how our rewriting system fares on a more difficult task, and used TED Talks, 6 BLEU below Medical for the English to French direction, for this purpose. In the same way as we did for Medical, we first tried to find the best training configuration for the ranker of the rewriting system. For this task, mixing the n-best neighborhood and 10pefNeigh with n=10 seemed to be sufficient to have no more improvement on the development set by increasing n for both language directions, so we used this training configuration. As for the rewriting phrase table used on the test set, we simply selected conf10k as in the Medical task. Results are reported in Table 5 for French to English and English to French.

We first observe that reranker performed similarly for the two translation directions, by improving 1-pass Moses by 0.5 BLEU. The smaller improvements may be partly attributed to the better LM used in 1-pass Moses, implying a better early modeling of grammaticality, but also by the fact that models such as SOUL and POS LMs rely on accurate contexts and are therefore more apt to help in choosing translations among generally better candidates.

Finally, rewriter obtains smaller but consistent improvements over reranker: +0.4 BLEU for translation into English, and +0.9 BLEU for translation into French. The smaller improvement in the former situation may be attributed to the nature of the target language which has a simpler agreement system. Consequently, the form-related errors discussed in Section 4.4 are possibly less subject to improvement here.

	extra	*missing*	*incorrect*				*unknown*	
	word	word	disamb	form	style	order	word	**all**
`1-pass Moses`	11	1	57	91	13	31	10	214
`reranker`	5	3	47	73	11	19	10	168
`rewriter`	4	4	40	55	12	19	10	144
`rewriter` oracle	19	2	26	44	14	22	10	137

Table 4: Results for manual error analysis for the first 70 test sentences.

System	fr-en		en-fr	
	BLEU	TER	BLEU	TER
`1-pass Moses`	32.5	47.7	32.3	49.9
`reranker`	33.0	47.3	32.8	49.4
`rewriter`	$33.4_{(+0.4)}$	$47.4_{(+0.1)}$	$33.7_{(+0.9)}$	$49.3_{(-0.1)}$
`semi-oracle`	$34.1_{(+1.1)}$	$46.6_{(-0.7)}$	$34.2_{(+1.4)}$	$48.6_{(-0.8)}$

Table 5: Results for the baselines, our best configuration and the semi-oracle for the `TED Talks`.

5 Related work

Reranking of translation hypotheses n-best list reranking was extensively studied in (Och et al., 2004), using features not used in the initial decoder such as IBM1 scores (which also proved useful for word-level confidence estimation (Blatz et al., 2004)) and generative syntactic models. While the experiments in (Och et al., 2004) did not show any clear contribution of syntactic information used in this manner, the later work by Carter and Monz (2011) managed to successfully exploit syntactic features using discriminative language modeling for n-best reranking. Gimpel *et al.* (2013) outperformed n-best reranking by generating, with an expensive but simple method, diverse hypotheses used as training data. Recently, Luong *et al.* (2014b) reranked n-best lists using confidence scores at the hypothesis level computed from word-level confidence measures learnt from roughly 10,000 SMT system outputs annotated by humans.

Rewriting of translation hypotheses Langlais *et al.* (2007) described a greedy search decoder, first introduced in (Germann et al., 2001), able to improve translations produced by a dynamic programming decoder using the same scoring function and translation table. However, the more recent work by Arun *et al.* (2010) using a Gibbs sampler for approximating maximum translation decoding showed the adequacy of the approximations made by state-of-the-art decoders for finding the best translation in their search space. Other works were more directly targeted at automatic post-editing of SMT output, and approached the problem as one of second-pass translation between automatic predictions and correct translations (Simard et al., 2007; Dugast et al., 2007). The recent work of Zhu *et al.* (2013) attempts to repair translations by exploiting confidence estimates for examples derived from the similarity between source words in the input text and in training examples. Luong *et al.* (2014a) obtained improvements by computing word confidence estimation, trained on human annotated data, and large sets of lexical, syntactic and semantic features, for the words in the n-best list produced during a first-pass decoding, and performing a second-pass decoding exploiting these new scores.

Confidence estimation of Machine Translation The Word Posterior Probability (WPP) proposed by Ueffing and Ney (2007), derived from information from the n-best list produced by a decoder, proved to be useful for estimating word-level confidence. Bach *et al.* (2011) worked on the issue of predicting sentence-level and word-level MT errors by using WPP and other features derived from the source context, the source-target alignment, and dependency structures, but relied on a significantly large manually annotated corpus of MT errors. De Gispert *et al.* (2013) calculate k-

gram posterior probabilities from n-best lists or word lattices, and demonstrated that they were reasonably accurate indications of whether specific k-grams would be found or not in human reference translations. Finally, the work of Blackwood *et al.* (2010) proposed to segment translation lattices according to confidence measures over the maximum likelihood translation hypothesis to focus on regions with potential translation errors. Hypothesis space constraints based on monolingual coverage are then applied to the low confidence regions to improve translation fluency.

6 Conclusions and perspectives

In this paper, we have described an approach that improves translations *a posteriori* by applying simple local rewritings. We have shown that the quality of phrase-level confidence estimates has a direct impact of the amplitude of the improvements that can be obtained, as well as the initial quality of the rewritten hypotheses. We have used a very simple definition for confidence estimates under the form of phrase posteriors estimated from n-best lists from an initial decoder, which obtained good empirical performance, in spite of not requiring large human-annotated datasets as in other approaches (Bach et al., 2011; Luong et al., 2014b).

Our work could be extended in several directions. First, we could use a larger set of rewriting operations (Langlais et al., 2007), including the `rewrite` (*sic*) operation introduced in (Marie and Max, 2013) that paraphrases source phrases and then translates them.

We could also possibly consider any phrase segmentation compatible with a specific word alignment rather than rely on specific phrase segmentations. This would allow us to attain faster some rewritings that could otherwise require several rewriting iterations and may never be attained by the greedy procedure.

More features could also be used, for instance to model more fine-grained syntax (Post, 2011) or document-level lexical coherence (Hardmeier et al., 2012). However, anticipating that some features might be very expensive to compute, we could adapt our procedure to work in several passes: initial passes would tend to restrict the search space more and more using an initial set of features, before a more expensive pass would concentrate on a limited number of hypotheses. Figure 1 indeed already showed a much faster oracle improvement between `1-pass Moses` and `reranker` for n-best list of small sizes.

Another avenue for improvement lies in the possibility to perform the training of our `rewriter` by providing it with more reference translations. As these are typically not readily available, we could resort to targeted paraphrasing (Madnani and Dorr, 2013) to rewrite reference translations into acceptable paraphrases that reuse n-grams from the best hypotheses of the system so far. Contrarily to (Madnani and Dorr, 2013), we could bias the paraphrasing table so that it only contains paraphrases that correspond to target phrases of high confidence values, which would add new n-grams likely of being produced by `rewriter`.

It is furthermore worth noticing that our work proposes a potential answer to an original question: contrarily to typical works on sub-sentencial MT confidence estimation, which predict whether a word or phrase is correct or not, our `rewriter` system could be used to determine automatically whether a rewriting system *could* (if asked to) attempt to improve locally a translation, or whether a human post-editor should already tackle working on improving it. As we showed in our manual error analysis in section 4.4, there are in fact many instances of errors that could not be recovered by our approach, be it because of its local rewriting strategy or of the bilingual resources or models used, so that some knowledge would have to be provided as hard constraints by a human translator, as hinted in (Crego et al., 2010). We could then finally have our `rewriter` system work in a turn-based fashion in collaboration with a human translator, fixing errors or making improvements that are being made possible by the last edits from the translator.

Acknowledgments

The authors would like to thank the anonymous reviewers and Guillaume Wisniewski for their useful remarks. Additional thanks go to Hai Son Le for "anticipating" the need for a large and efficient cache in his SOUL implementation, Quoc Khanh Do for his assistance on using SOUL, and Li Gong and Nicolas Pécheux for providing the authors with data used in the experiments. The work of the first author is supported by a CIFRE grant from French ANRT.

References

Alexandre Allauzen, Nicolas Pécheux, Quoc Khanh Do, Marco Dinarelli, Thomas Lavergne, Aurélien Max, Hai-son Le, and François Yvon. 2013. LIMSI @ WMT13. In *Proceedings of WMT*, Sofia, Bulgaria.

Abhishek Arun, Phil Blunsom, Chris Dyer, Adam Lopez, Barry Haddow, and Philipp Koehn. 2010. Monte Carlo inference and maximization for phrase-based translation. In *Proceedings of CoNLL*, Boulder, USA.

Nguyen Bach, Fei Huang, and Yaser Al-Onaizan. 2011. Goodness: A Method for Measuring Machine Translation Confidence. In *Proceedings of ACL*, Portland, USA.

Graeme Blackwood, Adrià de Gispert, and William Byrne. 2010. Fluency Constraints for Minimum Bayes-Risk Decoding of Statistical Machine Translation Lattices. In *Proceedings of COLING*, Beijing, China.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence Estimation for Machine Translation. In *Proceedings of COLING*, Geneva, Switzerland.

Simon Carter and Christof Monz. 2011. Syntactic Discriminative Language Model Rerankers for Statistical Machine Translation. *Machine Translation*, 25(4):317–339.

Stanley F. Chen and Joshua T. Goodman. 1998. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University.

Colin Cherry and George Foster. 2012. Batch Tuning Strategies for Statistical Machine Translation. In *Proceedings of NAACL*, Montréal, Canada.

Josep M. Crego, Aurélien Max, and François Yvon. 2010. Local lexical adaptation in Machine Translation through triangulation: SMT helping SMT. In *Proceedings of COLING*, Beijing, China.

Adrià de Gispert, Graeme Blackwood, Gonzalo Iglesias, and William Byrne. 2013. N-gram posterior probability confidence measures for statistical machine translation: an empirical study. *Machine Translation*, 27(2):85–114.

Loïc Dugast, Jean Senellart, and Philipp Koehn. 2007. Statistical Post-Editing on SYSTRANs Rule-Based Translation System. In *Proceedings of WMT*, Prague, Czech Republic.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast Decoding and Optimal Decoding for Machine Translation. In *Proceedings of ACL*, Toulouse, France.

Kevin Gimpel, Dhruv Batra, Chris Dyer, Gregory Shakhnarovich, and Virginia Tech. 2013. A Systematic Exploration of Diversity in Machine Translation. In *Proceedings of EMNLP*, Seatlle, USA.

Christian Hardmeier, Joakim Nivre, and Jorg Tiedeman. 2012. Document-Wide Decoding for Phrase-Based Statistical Machine Translation. In *Proceedings of EMNLP*, Jeju Island, Korea.

Saša Hasan and Hermann Ney. 2009. Comparison of Extended Lexicon Models in Search and Rescoring for SMT. In *Proceedings of NAACL, short papers*, Boulder, USA.

Kevin Knight. 2007. Automatic Language Translation Generation Help Needs Badly. In *MT Summit (invited talk)*, Copenhagen, Denmark.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of NAACL*, Edmonton, Canada.

Philippe Langlais, Alexandre Patry, and Fabrizio Gotti. 2007. A Greedy Decoder for Phrase-Based Statistical Machine Translation. In *Proceedings of Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, Skovde, Sweden.

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon. 2011. Structured Output Layer Neural Network Language Model. In *Proceedings of ICASSP*, Prague, Czech Republic.

Hai-Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous Space Translation Models with Neural Networks. In *Proceedings of NAACL*, Montréal, Canada.

Chin Y. Lin and Franz J. Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of COLING*, Geneva, Switzerland.

Ngoc-Quang Luong, Laurent Besacier, and Benjamin Lecouteux. 2014a. An Efficient Two-Pass Decoder for SMT Using Word Confidence Estimation. In *Proceedings of EAMT*, Dubrovnik, Croatia.

Ngoc-Quang Luong, Laurent Besacier, and Benjamin Lecouteux. 2014b. Word Confidence Estimation for SMT N -best List Re-ranking. In *Proceedings of the Workshop on Humans and Computer-assisted Translation (HaCaT)*, Gothenburg, Sweden.

Nitin Madnani and Bonnie J. Dorr. 2013. Generating Targeted Paraphrases for Improved Translation. *ACM Transactions on Intelligent Systems and Technology, special issue on Paraphrasing*, 4(3).

Benjamin Marie and Aurélien Max. 2013. A Study in Greedy Oracle Improvement of Translation Hypotheses. In *Proceedings of IWSLT*, Heidelberg, Germany.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A Smorgasbord of Features for Statistical Machine Translation. In *Proceedings of NAACL*, Boston, USA.

Kishore Papineni, Salim Roukos, Todd Ward, and Weijing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL*, Philadelphia, USA.

Kristen Parton, Nizar Habash, Kathleen R. McKeown, Gonzalo Iglesias, and Adrià de Gispert. 2012. Can Automatic Post-editing Make MT more Meaningful? In *Proceedings of EAMT*, Trento, Italy.

Nicolas Pécheux, Li Gong, Quoc Khanh Do, Benjamin Marie, Yulia Ivanishcheva, Alexander Allauzen, Thomas Lavergne, Jan Niehues, Aurélien Max, and François Yvon. 2014. LIMSI @ WMT'14 Medical Translation Task. In *Proceedings of WMT*, Baltimore, USA.

Matt Post. 2011. Judging Grammaticality with Tree Substitution Grammar Derivations. In *Proceedings of ACL, short papers*, Portland, USA.

Lane Schwartz, Chris Callison-Burch, William Schuler, and Stephen Wu. 2011. Incremental Syntactic Language Models for Phrase-based Translation. In *Proceedings of ACL*, Portland, USA.

Holger Schwenk, Daniel Déchelotte, and Jean-Luc Gauvain. 2006. Continuous Space Language Models for Statistical Machine Translation. In *Proceedings of COLING-ACL*, Sydney, Australia.

Michel Simard, Cyril Goutte, and Pierre Isabelle. 2007. Statistical Phrase-based Post-editing. In *Proceedings of NAACL*, Rochester, USA.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, , and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of AMTA*, Cambridge, USA.

Lucia Specia, Kashif Shah, Jose G.C. de Souza, and Trevor Cohn. 2013. QuEst - A Translation Quality Estimation Framework. In *Proceedings of ACL, System Demonstrations*, Sofia, Bulgaria.

Nicola Ueffing and Hermann Ney. 2007. Word-Level Confidence Estimation for Machine Translation. *Computational Linguistics*.

David Vilar, Jia Xu, Luis Fernando D'Haro, and Hermann Ney. 2006. Error Analysis of Statistical Machine Translation Output. In *Proceedings of LREC*, Genoa, Italy.

Richard Zens and Hermann Ney. 2006. N -Gram Posterior Probabilities for Statistical Machine Translation. In *Proceedings of WMT*, New York, USA.

Ying Zhang, Almut Silja Hildebrand, and Stephan Vogel. 2006. Distributed Language Modeling for N-best List Re-ranking. In *Proceedings of EMNLP*, Sydney, Australia.

Junguo Zhu, Muyun Yang, Sheng Li, and Tiejun Zhao. 2013. Repairing Incorrect Translation with Examples. In *Proceedings of IJCNLP*, Nagoya, Japan.

Learning Compact Lexicons for CCG Semantic Parsing

Yoav Artzi*
Computer Science & Engineering
University of Washington
Seattle, WA 98195
yoav@cs.washington.edu

Dipanjan Das Slav Petrov
Google Inc.
76 9th Avenue
New York, NY 10011
{dipanjand,slav}@google.com

Abstract

We present methods to control the lexicon size when learning a Combinatory Categorial Grammar semantic parser. Existing methods incrementally expand the lexicon by greedily adding entries, considering a single training datapoint at a time. We propose using corpus-level statistics for lexicon learning decisions. We introduce *voting* to globally consider adding entries to the lexicon, and *pruning* to remove entries no longer required to explain the training data. Our methods result in state-of-the-art performance on the task of executing sequences of natural language instructions, achieving up to 25% error reduction, with lexicons that are up to 70% smaller and are qualitatively less noisy.

1 Introduction

Combinatory Categorial Grammar (Steedman, 1996, 2000, CCG, henceforth) is a commonly used formalism for semantic parsing – the task of mapping natural language sentences to formal meaning representations (Zelle and Mooney, 1996). Recently, CCG semantic parsers have been used for numerous language understanding tasks, including querying databases (Zettlemoyer and Collins, 2005), referring to physical objects (Matuszek et al., 2012), information extraction (Krishnamurthy and Mitchell, 2012), executing instructions (Artzi and Zettlemoyer, 2013b), generating regular expressions (Kushman and Barzilay, 2013), question-answering (Cai and Yates, 2013) and textual entailment (Lewis and Steedman, 2013). In CCG, a lexicon is used to map words to formal representations of their meaning, which are then combined using bottom-up operations. In this paper we present learning techniques to explicitly control the size of the CCG lexicon, and show that this results in improved task performance and more compact models.

$$
\begin{array}{l}
\textbf{\textit{chair}} \vdash \boldsymbol{N} : \boldsymbol{\lambda x.\mathrm{chair(x)}} \\
\textbf{\textit{chair}} \vdash \boldsymbol{N} : \boldsymbol{\lambda x.\mathrm{sofa(x)}} \\
\textit{chair} \vdash AP : \lambda a.\mathrm{len(a,3)} \\
\textit{chair} \vdash NP : \mathcal{A}(\lambda x.\mathrm{corner(x)}) \\
\textit{chair} \vdash ADJ : \lambda x.\mathrm{hall(x)}
\end{array}
$$

Figure 1: Lexical entries for the word *chair* as learned with no corpus-level statistics. Our approach is able to correctly learn only the top two bolded entries.

In most approaches for inducing CCGs for semantic parsing, lexicon learning and parameter estimation are performed jointly in an online algorithm, as introduced by Zettlemoyer and Collins (2007). To induce the lexicon, words extracted from the training data are paired with CCG categories one sample at a time (for an overview of CCG, see §2). Joint approaches have the potential advantage that only entries participating in successful parses are added to the lexicon. However, new entries are added greedily and these decisions are never revisited at later stages. In practice, this often results in a large and noisy lexicon.

Figure 1 lists a sample of CCG lexical entries learned for the word *chair* with a greedy joint algorithm (Artzi and Zettlemoyer, 2013b). In the studied navigation domain, the word *chair* is often used to refer to chairs and sofas, as captured by the first two entries. However, the system also learns several spurious meanings: the third shows an erroneous usage of *chair* as an adverbial phrase describing action length, while the fourth treats it as a noun phrase and the fifth as an adjective. In contrast, our approach is able to correctly learn only the top two lexical entries.

We present a *batch* algorithm focused on controlling the size of the lexicon when learning CCG semantic parsers (§3). Because we make updates only after processing the entire training set, we

*This research was carried out at Google.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1273–1283,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

can take corpus-wide statistics into account before each lexicon update. To explicitly control the size of the lexicon, we adopt two complementary strategies: *voting* and *pruning*. First, we consider the lexical evidence each sample provides as a vote towards potential entries. We describe two voting strategies for deciding which entries to add to the model lexicon (§4). Second, even though we use voting to only conservatively add new lexicon entries, we also prune existing entries if they are no longer necessary for parsing the training data. These steps are incorporated into the learning framework, allowing us to apply stricter criteria for lexicon expansion while maintaining a single learning algorithm.

We evaluate our approach on the robot navigation semantic parsing task (Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013b). Our experimental results show that we outperform previous state of the art on executing sequences of instructions, while learning significantly more compact lexicons (§6 and Table 3).

2 Task and Inference

To present our lexicon learning techniques, we focus on the task of executing natural language navigation instructions (Chen and Mooney, 2011). This domain captures some of the fundamental difficulties in recent semantic parsing problems. In particular, it requires learning from weakly-supervised data, rather than data annotated with full logical forms, and parsing sentences in a situated environment. Additionally, successful task completion requires interpreting and executing multiple instructions in sequence, requiring accurate models to avoid cascading errors. Although this overview centers around the aforementioned task, our methods are generalizable to any semantic parsing approach that relies on CCG.

We approach the navigation task as a situated semantic parsing problem, where the meaning of instructions is represented with lambda calculus expressions, which are then deterministically executed. Both the mapping of instructions to logical forms and their execution consider the current state of the world. This problem was recently addressed by Artzi and Zettlemoyer (2013b) and our experimental setup mirrors theirs. In this section, we provide a brief background on CCG and describe the task and our inference method.

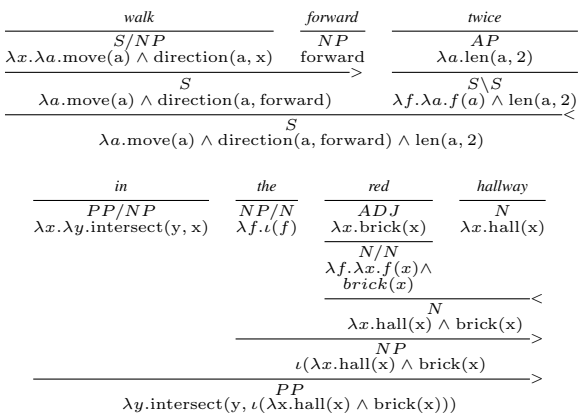


Figure 2: Two CCG parses. The top shows a complete parse with an adverbial phrase (AP), including unary type shifting and forward ($>$) and backward ($<$) application. The bottom fragment shows a prepositional phrase (PP) with an adjective (ADJ).

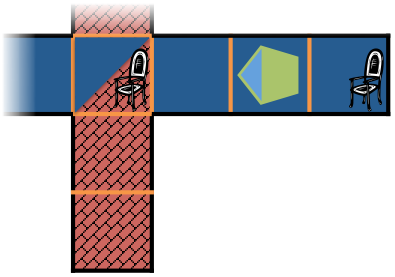
2.1 Combinatory Categorial Grammar

CCG is a linguistically-motivated categorial formalism for modeling a wide range of language phenomena (Steedman, 1996; Steedman, 2000). In CCG, parse tree nodes are categories, which are assigned to strings (single words or n-grams) and combined to create a complete derivation. For example, $S/NP : \lambda x.\lambda a.\text{move(a)} \wedge \text{direction(a, x)}$ is a CCG category describing an imperative verb phrase. The syntactic type S/NP indicates the category is expecting an argument of type NP on its right, and the returned category will have the syntax S. The directionality is indicated by the forward slash $/$, where a backward slash $\backslash$ would specify the argument is expected on the left. The logical form in the category represents its semantic meaning. For example, $\lambda x.\lambda a.\text{move(a)} \wedge \text{direction(a, x)}$ in the category above is a function expecting an argument, the variable x, and returning a function from events to truth-values, the semantic representation of imperatives. In this domain, the conjunction in the logical form specifies conditions on events. Specifically, the event must be a *move* event and have a specified *direction*.

A CCG is defined by a lexicon and a set of combinators. The lexicon provides a mapping from strings to categories. Figure 2 shows two CCG parses in the navigation domain. Parse trees are read top to bottom. Parsing starts by matching categories to strings in the sentence using the lexicon. For example, the lexical entry *walk* $\vdash$ S/NP : $\lambda x.\lambda a.\text{move(a)} \wedge \text{direction(a, x)}$ pairs the string *walk* with the example category above. Each intermediate parse node is constructed by applying

one of a small set of binary CCG combinators or unary operators. For example, in Figure 2 the category of the span *walk forward* is combined with the category of *twice* using backward application ($<$). Parsing concludes with a logical form that captures the meaning of the complete sentence.

We adopt a factored representation for CCG lexicons (Kwiatkowski et al., 2011), where entries are dynamically generated by combining *lexemes* and *templates*. A lexeme is a pair that consists of a natural language string and a set of logical constants, while the template contains the syntactic and semantic components of a CCG category, abstracting over logical constants. For example, consider the lexical entry $\textit{walk} \vdash S/NP : \lambda x.\lambda a.\text{move(a)} \wedge \text{direction(a, x)}$. Under the factored representation, this entry can be constructed by combining the lexeme $\langle \textit{walk}, \{\text{move}, \text{direction}\}\rangle$ and the template $\lambda v_1.\lambda v_2.[S/NP : \lambda x.\lambda a.v_1(a) \wedge v_2(a, x)]$. This representation allows for better generalization over unseen lexical entries at inference time, allowing for pairings of templates and lexemes not seen during training.



facing the chair in the intersection move forward twice
$\lambda a.\text{pre(a, front(you, } \iota(\lambda x.\text{chair(x)} \wedge$
$\text{intersect(x, } \iota(\lambda y.\text{intersection(y)}))))) \wedge$
$\text{move(a)} \wedge \text{len(a, 2)}$
$\langle$FORWARD, FORWARD$\rangle$

turn left
$\lambda a.\text{turn(a)} \wedge \text{direction(a, left)}$
$\langle$LEFT$\rangle$

go to the end of the hall
$\lambda x.move(a) \wedge \text{to(a, } \iota(\lambda x.\text{end(x, } \iota(\lambda y.\text{hall(y)}))))$
$\langle$FORWARD, FORWARD$\rangle$

Figure 3: Fragment of a map and instructions for the navigation domain. The fragment includes two intersecting hallways (red and blue), two chairs and an agent facing left (green pentagon), which follows instructions such as these listed below. Each instruction is paired with a logical form representing its meaning and its execution in the map.

2.2 Situated Log-Linear CCGs

We use a CCG to parse sentences to logical forms, which are then executed. Let $\mathcal{S}$ be a set of states, $\mathcal{X}$ be the set of all possible sentences, and $\mathcal{E}$ be the space of executions, which are $\mathcal{S} \rightarrow \mathcal{S}$ functions. For example, in the navigation task from Artzi and Zettlemoyer (2013b), $\mathcal{S}$ is a set of positions on a map, as illustrated in Figure 3. The map includes an agent that can perform four actions: LEFT, RIGHT, MOVE, and NULL. An execution e is a sequence of actions taken consecutively. Given a state $s \in \mathcal{S}$ and a sentence $x \in \mathcal{X}$, we aim to find the execution $e \in \mathcal{E}$ described in x. Let $\mathcal{Y}$ be the space of CCG parse trees and $\mathcal{Z}$ the space of all possible logical forms. Given a sentence x we generate a CCG parse $y \in \mathcal{Y}$, which includes a logical form $z \in \mathcal{Z}$. An execution e is then generated from z using a deterministic process.

Parsing with a CCG requires choosing appropriate lexical entries from an often ambiguous lexicon and the order in which operations are applied. In a situated scenario such choices must account for the current state of the world. In general, given a CCG, there are many parses for each sentence-state pair. To discriminate between competing parses, we use a situated log-linear CCG, inspired by Clark and Curran (2007).

Let $\textsc{Gen}(x, s; \Lambda) \subset \mathcal{Y}$ be the set of all possible CCG parses given the sentence x, the current state s and the lexicon Λ. In $\textsc{Gen}(x, s; \Lambda)$, multiple parse trees may have the same logical form; let $\mathcal{Y}(z) \subset \textsc{Gen}(x, s; \Lambda)$ be the subset of such parses with the logical form z at the root. Also, let $\theta \in \mathbb{R}^d$ be a d-dimensional parameter vector. We define the probability of the logical form z as:

$$p(z|x, s; \theta, \Lambda) = \sum_{y \in \mathcal{Y}(z)} p(y|x, s; \theta, \Lambda) \quad (1)$$

Above, we marginalize out the probabilities of all parse trees with the same logical form z at the root. The probability of a parse tree y is defined as:

$$p(y|x, s; \theta, \Lambda) = \frac{e^{\theta \cdot \phi(x,s,y)}}{\sum_{y' \in \textsc{Gen}(x,s;\Lambda)} e^{\theta \cdot \phi(x,s,y')}} \quad (2)$$

Where $\phi(x, s, y) \in \mathbb{R}^d$ is a feature vector. Given a logical form z, we deterministically map it to an execution $e \in \mathcal{E}$. At inference time, given a sentence x and state s, we find the best logical form z^* (and its corresponding execution) by solving:

$$z^* = \arg\max_z \; p(z|x, s; \theta, \Lambda) \quad (3)$$

The above $\arg\max$ operation sums over all trees $y \in \mathcal{Y}(z)$, as described in Equation 1. We use a CKY chart for this computation. The chart signature in each span is a CCG category. Since exact inference is prohibitively expensive, we follow previous work and perform bottom-up beam search, maintaining only the k-best categories for each span in the chart. The logical form z^* is taken from the k-best categories at the root of the chart. The partition function in Equation 2 is approximated by summing the inside scores of all categories at the root. We describe the choices of hyperparameters and details of our feature set in §5.

3 Learning

Learning a CCG semantic parser requires inducing the entries of the lexicon Λ and estimating parsing parameters θ. We describe a batch learning algorithm (Figure 4), which explicitly attempts to induce a compact lexicon, while fully explaining the training data. At training time, we assume access to a set of N examples $\mathcal{D} = \left\{d^{(i)}\right\}_1^N$, where each datapoint $d^{(i)} = \langle x^{(i)}, s^{(i)}, e^{(i)} \rangle$, consists of an instruction $x^{(i)}$, the state $s^{(i)}$ where the instruction is issued and its execution demonstration $e^{(i)}$. In particular, we know the correct execution for each state and instruction, but we do not know the correct CCG parse and logical form. We treat the choices that determine them, including selection of lexical entries and parsing operators, as latent. Since there can be many logical forms $z \in \mathcal{Z}$ that yield the same execution $e^{(i)}$, we marginalize over the logical forms (using Equation 1) when maximizing the following regularized log-likelihood:

$$\mathscr{L}(\theta, \Lambda, \mathcal{D}) = \sum_{d^{(i)} \in \mathcal{D}} \sum_{z \in \mathcal{Z}(e^{(i)})} p(z|x^{(i)}, s^{(i)}; \theta, \Lambda) - \frac{\gamma}{2}\|\theta\|_2^2 \quad (4)$$

Where $\mathcal{Z}(e^{(i)})$ is the set of logical forms that result in the execution $e^{(i)}$ and the hyperparameter γ is a regularization constant. Due to the large number of potential combinations,[1] it is impractical to consider the complete set of lexical entries, where all strings (single words and n-grams) are associated with all possible CCG categories. Therefore, similar to prior work, we gradually expand the lexicon during learning. As a result, the parameter space changes throughout training whenever the lexicon is modified. The learning problem involves jointly finding the best set of parameters and lexicon entries. In the remainder of this section, we describe how we optimize Equation 4, while explicitly controlling the lexicon size.

[1]For the navigation task, given the set of CCG category templates (see §2.1) and parameters used there would be between 7.5-10.2M lexical entries to consider, depending on the corpus used (§5).

Algorithm 1 Batch algorithm for maximizing $\mathscr{L}(\theta, \Lambda, \mathcal{D})$. See §3.1 for details.

Input: Training dataset $\mathcal{D} = \left\{d^{(i)}\right\}_1^N$, number of learning iterations T, seed lexicon Λ_0, a regularization constant γ, and a learning rate μ. VOTE is defined in §4.
Output: Lexicon Λ and model parameters θ

$\Lambda \leftarrow \Lambda_0$
for $t = 1$ to T **do**
  » Generate lexical entries for all datapoints.
  for $i = 1$ to N **do**
    $\lambda^{(i)} \leftarrow \textsc{GenEntries}(d^{(i)}, \theta, \Lambda)$
  » Add corpus-wide voted entries to model lexicon.
  $\Lambda \leftarrow \Lambda \cup \textsc{Vote}(\Lambda, \{\lambda^{(1)}, \ldots, \lambda^{(N)}\})$
  » Compute gradient and entries to prune.
  for $i = 1$ to N **do**
    $\langle \lambda_-^{(i)}, \Delta^{(i)} \rangle \leftarrow \textsc{ComputeUpdate}(d^{(i)}, \theta, \Lambda)$
  » Prune lexicon.
  $\Lambda \leftarrow \Lambda \setminus \bigcap_{i=1}^{N} \lambda_-^{(i)}$
  » Update model parameters.
  $\theta \leftarrow \theta + \mu \sum_{i=1}^{N} \Delta^{(i)} - \gamma\theta$
return Λ and θ

Algorithm 2 GENENTRIES: Algorithm to generate lexical entries from one training datapoint. See §3.2 for details.

Input: Single datapoint $d = \langle x, s, e \rangle$, current model parameters θ and lexicon Λ.
Output: Datapoint-specific lexicon entries λ.

» Augment lexicon with sentence-specific entries.
$\Lambda_+ \leftarrow \Lambda \cup \textsc{GenLex}(d, \Lambda, \theta)$
» Get max-scoring parses producing correct execution.
$\boldsymbol{y}_+ \leftarrow \textsc{GenMax}(x, s, e; \Lambda_+, \theta)$
» Extract lexicon entries from max-scoring parses.
$\lambda \leftarrow \bigcup_{y \in \boldsymbol{y}_+} \textsc{Lex}(y)$
return λ

Algorithm 3 COMPUTEUPDATE: Algorithm to compute the gradient and the set of lexical entries to prune for one datapoint. See §3.3 for details.

Input: Single datapoint $d = \langle x, s, e \rangle$, current model parameters θ and lexicon Λ.
Output: $\langle \lambda_-, \Delta \rangle$, lexical entries to prune for d and gradient.

» Get max-scoring correct parses given Λ and θ.
$\boldsymbol{y}_+ \leftarrow \textsc{GenMax}(x, s, e; \Lambda, \theta)$
» Create the set of entries to prune.
$\lambda_- \leftarrow \Lambda \setminus \bigcup_{y \in \boldsymbol{y}_+} \textsc{Lex}(y)$
» Compute gradient.
$\Delta \leftarrow \mathbb{E}(y \mid x, s, e; \theta, \Lambda) - \mathbb{E}(y \mid x, s; \theta, \Lambda)$
return $\langle \lambda_-, \Delta \rangle$

Figure 4: Our learning algorithm and its subroutines.

3.1 Optimization Algorithm

We present a learning algorithm to optimize the data log-likelihood, where both lexicon learning and parameter updates are performed in *batch*, i.e., after observing all the training corpus. The batch formulation enables us to use information from the entire training set when updating the model lexicon. Algorithm 1 presents the outline of our optimization procedure. It takes as input a training dataset $\mathcal{D}$, number of iterations T, seed lexicon Λ_0, learning rate μ and regularization constant γ.

Learning starts with initializing the model lexicon Λ using Λ_0 (line 1). In lines 2-9, we run T iterations; in each, we make two passes over the corpus, first to generate lexical entries, and second to compute gradient updates and lexical entries to prune. To generate lexical entries (lines 3-4) we use the subroutine GENENTRIES to independently generate entries for each datapoint, as described in §3.2. Given the entries for each datapoint, we vote on which to add to the model lexicon. The subroutine VOTE (line 5) chooses a subset of the proposed entries using a particular voting strategy (see §4). Given the updated lexicon, we process the corpus a second time (lines 6-7). The subroutine COMPUTEUPDATE, as described in §3.3, computes the gradient update for each datapoint $d^{(i)}$, and also generates the set of lexical entries not included in the max-scoring parses of $d^{(i)}$, which are candidates for pruning. We prune from the model lexicon all lexical entries not used in any correct parse (line 8). During this pruning step, we ensure that no entries from Λ_0 are removed from Λ. Finally, the gradient updates are accumulated to update the model parameters (line 9).

3.2 Lexical Entries Generation

For each datapoint $d = \langle x, s, e \rangle$, the subroutine GENENTRIES, as described in Algorithm 2, generates a set of potential entries. The subroutine uses the function GENLEX, originally proposed by Zettlemoyer and Collins (2005), to generate lexical entries from sentences paired with logical forms. We use the weakly-supervised variant of Artzi and Zettlemoyer (2013b). Briefly, GENLEX uses the sentence and expected execution to generate new lexemes, which are then paired with a set of templates factored from Λ_0 to generate new lexical entries. For more details, see §8 of Artzi and Zettlemoyer (2013b).

Since GENLEX over-generates entries, we need to determine the set of entries that participate in max-scoring parses that lead to the correct execution e. We therefore create a sentence-specific lexicon Λ_+ by taking the union of the GENLEX-generated entries for the current sentence and the model lexicon (line 1). We define $\textsc{GenMax}(x, s, e; \Lambda_+, \theta)$ to be the set of all max-scoring parses according to the parameters θ that are in $\textsc{Gen}(x, s; \Lambda_+)$ and result in the correct execution e (line 2). In line 3 we use the function $\textsc{Lex}(y)$, which returns the lexical entries used in the parse y, to compute the set of all lexical entries used in these parses. This final set contains all newly generated entries for this datapoint and is returned to the optimization algorithm.

3.3 Pruning and Gradient Computation

Algorithm 3 describes the subroutine COMPUTEUPDATE that, given a datapoint d, the current model lexicon Λ and model parameters θ, returns the gradient update and the set of lexical entries to prune for d. First, similar to GENENTRIES we compute the set of correct max-scoring parses using GENMAX (line 1). This time, however, we do not use a sentence-specific lexicon, but instead use the model lexicon that has been expanded with all voted entries. As a result, the set of max-scoring parses producing the correct execution may be different compared to GENENTRIES. $\textsc{Lex}(y)$ is then used to extract the lexical entries from these parses, and the set difference (λ_-) between the model lexicon and these entries is set to be pruned (line 2). Finally, the partial derivative for the datapoint is computed using the difference of two expected feature vectors, according to two distributions (line 3): (a) parses conditioned on the correct execution e, the sentence x, state s and the model, and (b) all parses not conditioned on the execution e. The derivatives are approximate due to the use of beam search, as described in §2.2.

4 Global Voting for Lexicon Learning

Our goal is to learn compact and accurate CCG lexicons. To this end, we globally reason about adding new entries to the lexicon by *voting* (VOTE, Algorithm 1, line 5), and remove entries by *pruning* the ones no longer required for explaining the training data (Algorithm 1, line 8). In voting, each datapoint can be considered as attempting to influence the learning algorithm to update the model lexicon with the entries required to parse it. In this

	Round 1	Round 2	Round 3	Round 4
$d^{(1)}$	⟨*chair*, {chair}⟩ 1/3 ⟨*chair*, {hatrack}⟩ 1/3 ⟨*chair*, {turn, direction}⟩ 1/3	⟨*chair*, {chair}⟩ 1/2 ⟨*chair*, {hatrack}⟩ 1/2	⟨*chair*, {chair}⟩ 1	⟨*chair*, {chair}⟩ 1
$d^{(2)}$	⟨*chair*, {chair}⟩ 1/2 ⟨*chair*, {hatrack}⟩ 1/2	⟨*chair*, {chair}⟩ 1/2 ⟨*chair*, {hatrack}⟩ 1/2	⟨*chair*, {chair}⟩ 1	⟨*chair*, {chair}⟩ 1
$d^{(3)}$	⟨*chair*, {chair}⟩ 1/2 ⟨*chair*, {easel}⟩ 1/2	⟨*chair*, {chair}⟩ 1/2 ⟨*chair*, {easel}⟩ 1/2	⟨*chair*, {chair}⟩ 1/2 ⟨*chair*, {easel}⟩ 1/2	⟨*chair*, {chair}⟩ 1
$d^{(4)}$	⟨*chair*, {easel}⟩ 1	⟨*chair*, {easel}⟩ 1	⟨*chair*, {easel}⟩ 1	⟨*chair*, {easel}⟩ 1
Votes	⟨*chair*, {chair}⟩ 1 1/3 ⟨*chair*, {easel}⟩ 1 1/2 ⟨*chair*, {hatrack}⟩ 5/6 ⟨*chair*, {turn, direction}⟩ 1/3	⟨*chair*, {chair}⟩ 1 1/2 ⟨*chair*, {easel}⟩ 1 1/2 ⟨*chair*, {hatrack}⟩ 1	⟨*chair*, {chair}⟩ 2 1/2 ⟨*chair*, {easel}⟩ 1 1/2	**⟨*chair*, {chair}⟩** 3 ⟨*chair*, {easel}⟩ 1
Discard	⟨*chair*, {turn, direction}⟩	⟨*chair*, {hatrack}⟩	⟨*chair*, {easel}⟩	

Figure 5: Four rounds of CONSENSUSVOTE for the string *chair* for four training datapoints. For each datapoint, we specify the set of lexemes generated in the Round 1 column, and update this set after each round. At the end, the highest voted new lexeme according to the final votes is returned. In this example, MAXVOTE and CONSENSUSVOTE lead to different outcomes. MAXVOTE, based on the initial sets only, will select ⟨*chair*, {easel}⟩.

section we describe two alternative voting strategies. Both strategies ensure that new entries are only added when they have wide support in the training data, but count this support in different ways. For reproducibility, we also provide step-by-step pseudocode for both methods in the supplementary material.

Since we only have access to executions and treat parse trees as latent, we consider as correct all parses that produce correct executions. Frequently, however, incorrect parses spuriously lead to correct executions. Lexical entries extracted from such spurious parses generalize poorly. The goal of voting is to eliminate such entries.

Voting is formulated on the factored lexicon representation, where each lexical entry is factored into a lexeme and a template, as described in §2.1. Each lexeme is a pair containing a natural language string and a set of logical constants.[2] A lexeme is combined with a template to create a lexical entry. In our lexicon learning approach only new lexemes are generated, while the set of templates is fixed; hence, our voting strategies reason over lexemes and only create complete lexicon entries at the end. Decisions are made for each string independently of all other strings, but considering all occurrences of that string in the training data.

In lines 3-4 of Algorithm 1 GENENTRIES is used to propose new lexical entries for each training datapoint $d^{(i)}$. For each $d^{(i)}$ a set $\lambda^{(i)}$, that includes all lexical entries participating in parses that lead to the correct execution, is generated. In these sets, the same string can appear in multiple lexemes. To normalize its influence, each datapoint is given a vote of 1.0 for each string, which is distributed uniformly among all lexemes containing the same string.

For example, a specific $\lambda^{(i)}$ may consist of the following three lexemes: $\langle$*chair*, {chair}$\rangle$, $\langle$*chair*, {hatrack}$\rangle$, $\langle$*face*, {post, front, you}$\rangle$. In this set, the phrase *chair* has two possible meanings, which will therefore each receive a vote of 0.5, while the third lexeme will be given a vote of 1.0. Such ambiguity is common and occurs when the available supervision is insufficient to discriminate between different parses, for example, if they lead to identical executions.

Each of the two following strategies reasons over these votes to globally select the best lexemes. To avoid polluting the model lexicon, both strategies adopt a conservative approach and only select at most one lexeme for each string in each training iteration.

4.1 Strategy 1: MAXVOTE

The first strategy for selecting voted lexical entries is straightforward. For each string it simply aggregates all votes and selects the new lexeme with the most votes. A lexeme is considered new if it is not already in the model lexicon. If no such single lexeme exists (e.g., no new entries were used in correctly executing parses or in the case of a tie) no lexeme is selected in this iteration.

A potential limitation of MAXVOTE is that the votes for all rejected lexemes are lost. However, it is often reasonable to re-allocate these votes to other lexemes. For example, consider the sets of lexemes for the word *chair* in the Round 1 col-

[2] Recall, for example, that in one lexeme the string *walk* may be paired with the set of constants {move, direction}.

umn of Figure 5. Using MAXVOTE on these sets will select the lexeme $\langle \textit{chair}, \{\text{easel}\}\rangle$, rather than the correct lexeme $\langle \textit{chair}, \{\text{chair}\}\rangle$. This occurs when the datapoints supporting the correct lexeme distribute their votes over many spurious lexemes.

4.2 Strategy 2: CONSENSUSVOTE

Our second strategy CONSENSUSVOTE aims to capture the votes that are lost in MAXVOTE. Instead of discarding votes that do not go to the maximum scoring lexeme, voting is done in several rounds. In each round the lowest scoring lexeme is discarded and votes are re-assigned uniformly to the remaining lexemes. This procedure is continued until convergence. Finally, given the sets of lexemes in the last round, the votes are computed and the new lexeme with most votes is selected.

Figure 5 shows a complete voting process for four training datapoints. In each round, votes are aggregated over the four sets of lexemes, and the lexeme with the fewest votes is discarded. For each set of lexemes, the discarded lexeme is removed, unless it will lead to an empty set.[3] In the example, while $\langle \textit{chair}, \{\text{easel}\}\rangle$ is discarded in Round 3, it remains in the set of $d^{(4)}$. The process converges in the fourth round, when there are no more lexemes to discard. The final sets include two entries: $\langle \textit{chair}, \{\text{chair}\}\rangle$ and $\langle \textit{chair}, \{\text{easel}\}\rangle$. By avoiding wasting votes on lexemes that have no chance of being selected, the more widely supported lexeme $\langle \textit{chair}, \{\text{chair}\}\rangle$ receives the most votes, in contrast to Round 1, where $\langle \textit{chair}, \{\text{easel}\}\rangle$ was the highest voted one.

5 Experimental Setup

To isolate the effect of our lexicon learning techniques we closely follow the experimental setup of previous work (Artzi and Zettlemoyer, 2013b, §9) and use its publicly available code.[4] This includes the provided beam-search CKY parser, two-pass parsing for testing, beam search for executing sequences of instructions and the same seed lexicon, weight initialization and features. Finally, except the optimization parameters specified below, we use the same parameter settings.

Data For evaluation we use two related corpora: SAIL (Chen and Mooney, 2011) and ORACLE (Artzi and Zettlemoyer, 2013b). Due to how the original data was collected (MacMahon et al., 2006), SAIL includes many wrong executions and about 30% of all instruction sequences are infeasible (e.g., instructing the agent to walk into a wall). To better understand system performance and the effect of noise, ORACLE was created with the subset of valid instructions from SAIL paired with their gold executions. Following previous work, we use a held-out set for the ORACLE corpus and cross-validation for the SAIL corpus.

Systems We report two baselines. Our batch baseline uses the same regularized algorithm, but updates the lexicon by adding all entries without voting and skips pruning. Additionally, we added post-hoc pruning to the algorithm of Artzi and Zettlemoyer (2013b) by discarding all learned entries that are not participating in max-scoring correct parses at the end of training. For ablation, we study the influence of the two voting strategies and pruning, while keeping the same regularization setting. Finally, we compare our approach to previous published results on both corpora.

Optimization Parameters We optimized the learning parameters using cross validation on the training data to maximize recall of complete sequence execution and minimize lexicon size. We use 10 training iterations and the learning rate $\mu = 0.1$. For SAIL we set the regularization parameter $\gamma = 1.0$ and for ORACLE $\gamma = 0.5$.

Full Sequence Inference To execute sequences of instructions we use the beam search procedure of Artzi and Zettlemoyer (2013b) with an identical beam size of 10. The beam stores states, and is initialized with the starting state. Instructions are executed in order, each is attempted from all states currently in the beam, the beam is then updated and pruned to keep the 10-best states. At the end, the best scoring state in the beam is returned.

Evaluation Metrics We evaluate the end-to-end task of executing complete sequences of instructions against an oracle final state. In addition, to better understand the results, we also measure task completion for single instructions. We repeated

[3] This restriction is meant to ensure that discarding lexemes will not change the set of sentences that can be parsed. In addition, it means that the total amount of votes given to a string is invariant between rounds. Allowing for empty sets will change the sum of votes, and therefore decrease the number of datapoints contributing to the decision.

[4] Their implementation, based on the University of Washington Semantic Parsing Framework (Artzi and Zettlemoyer, 2013a), is available at `http://yoavartzi.com/navi`.

ORACLE corpus cross-validation	Single sentence			Sequence			Lexicon
	P	R	F1	P	R	F1	size
Artzi and Zettlemoyer (2013b)	84.59	82.74	83.65	68.35	58.95	63.26	5383
w/ post-hoc pruning	84.32	82.89	83.60	66.83	61.23	63.88	3104
Batch baseline	85.14	81.91	83.52	72.64	60.13	65.76	6323
w/ MAXVOTE	84.04	82.25	83.14	72.79	64.86	68.55	2588
w/ CONSENSUSVOTE	84.51	82.23	83.36	72.99	63.45	67.84	2446
w/ pruning	**85.58**	**83.51**	**84.53**	75.15	65.97	70.19	2791
w/ MAXVOTE + pruning	84.50	82.89	83.69	72.91	**66.40**	69.47	2186
w/ CONSENSUSVOTE + pruning	85.22	83.00	84.10	**75.65**	66.15	**70.55**	**2101**

Table 1: Ablation study using cross-validation on the ORACLE corpus training data. We report mean precision (P), recall (R) and harmonic mean (F1) of execution accuracy on single sentences and sequences of instructions and mean lexicon sizes. Bold numbers represent the best performing method on a given metric.

Final results		Single sentence			Sequence			Lexicon
		P	R	F1	P	R	F1	size
SAIL	Chen and Mooney (2011)		54.40			16.18		
	Chen (2012)		57.28			19.18		
	+ additional data		57.62			20.64		
	Kim and Mooney (2012)		57.22			20.17		
	Kim and Mooney (2013)		62.81			26.57		
	Artzi and Zettlemoyer (2013b)	**67.60**	**65.28**	**66.42**	38.06	31.93	34.72	10051
	Our Approach	66.67	64.36	65.49	**41.30**	**35.44**	**38.14**	**2873**
ORACLE	Artzi and Zettlemoyer (2013b)	**81.17** (0.68)	**78.63** (0.84)	**79.88** (0.76)	68.07 (2.72)	58.05 (3.12)	62.65 (2.91)	6213 (217)
	Our Approach	79.86 (0.50)	77.87 (0.41)	78.85 (0.45)	**76.05** (1.79)	**68.53** (1.76)	**72.10** (1.77)	**2365** (57)

Table 2: Our final results compared to previous work on the SAIL and ORACLE corpora. We report mean precision (P), recall (R), harmonic mean (F1) and lexicon size results and standard deviation between runs (in parenthesis) when appropriate. *Our Approach* stands for batch learning with a consensus voting and pruning. Bold numbers represent the best performing method on a given metric.

each experiment five times and report mean precision, recall,[5] harmonic mean (F1) and lexicon size. For held-out test results we also report standard deviation. For the baseline online experiments we shuffled the training data between runs.

6 Results

Table 1 shows ablation results for 5-fold cross-validation on the ORACLE training data. We evaluate against the online learning algorithm of Artzi and Zettlemoyer (2013b), an extension of it to include post-hoc pruning and a batch baseline. Our best sequence execution development result is obtained with CONSENSUSVOTE and pruning. The results provide a few insights. First, simply switching to batch learning provides mixed results: precision increases, but recall drops and the learned lexicon is larger. Second, adding pruning results in a much smaller lexicon, and, especially in batch learning, boosts performance. Adding voting further reduces the lexicon size and provides additional gains for sequence execution. Finally, while MAXVOTE and CONSENSUSVOTE give comparable performance on their own, CONSENSUSVOTE results in more precise and compact models when combined with pruning.

Table 2 lists our test results. We significantly outperform previous state of the art on both corpora when evaluating sequence accuracy. In both scenarios our lexicon is 60-70% smaller. In contrast to the development results, single sentence performance decreases slightly compared to Artzi and Zettlemoyer (2013b). The discrepancy between single sentence and sequence results might be due to the beam search performed when executing sequences of instructions. Models with more compact lexicons generate fewer logical forms for each sentence: we see a decrease of roughly 40% in our models compared to Artzi and Zettlemoyer (2013b). This is especially helpful during sequence execution, where we use a beam size of 10, resulting in better sequences of executions. In general, this shows the potential benefit of using more compact models in scenarios that incorporate reasoning about parsing uncertainty.

To illustrate the types of errors avoided with voting and pruning, Table 3 describes common error classes and shows example lexical entries for batch trained models with CONSENSUSVOTE and pruning and without. Quantitatively, the mean number of entries per string on development folds

[5]Recall is identical to accuracy as reported in prior work.

String	# lexical entries		Example categories
	Batch baseline	With voting and pruning	
The algorithm often treats common bigrams as multiword phrases, and later learns the more general separate entries. Without pruning the initial entries remain in the lexicon and compete with the correct ones during inference.			
octagon carpet	45	0	~~$N : \lambda x.\text{wall(x)}$~~ ~~$N : \lambda x.\text{hall(x)}$~~ ~~$N : \lambda x.\text{honeycomb(x)}$~~
carpet	51	5	$N : \lambda x.\text{hall(x)}$ ~~$N/N : \lambda f.\lambda x.x == \text{argmin(f, }\lambda\text{y.dist(y))}$~~
octagon	21	5	$N : \lambda x.\text{honeycomb(x)}$ ~~$N : \lambda x.\text{cement(x)}$~~ $ADJ : \lambda x.\text{honeycomb(x)}$
We commonly see in the lexicon a long tail of erroneous entries, which compete with correctly learned ones. With voting and pruning we are often able to avoid such noisy entries. However, some noise still exists, e.g., the entry for "intersection".			
intersection	45	7	$N : \lambda x.\text{intersection(x)}$ ~~$S\backslash N : \lambda f.\text{intersect(you, (f))}$~~ $AP : \lambda a.len(a, 1)$ ~~$N/NP : \lambda x.\lambda y.\text{intersect(y, x)}$~~
twice	46	2	$AP : \lambda a.\text{len(a, 2)}$ ~~$AP : \lambda a.\text{pass(a, }\mathcal{A}\text{(}\lambda\text{x.empty(x)))}$~~ ~~$AP : \lambda a.\text{pass(a, }\mathcal{A}\text{(}\lambda\text{x.hall(x)))}$~~
stone	31	5	$ADJ : \lambda x.\text{stone(x)}$ ~~$ADJ : \lambda x.\text{brick(x)}$~~ ~~$ADJ : \lambda x.\text{honeycomb(x)}$~~ ~~$NP/N : \lambda f.\mathcal{A}(f)$~~
Not all concepts mentioned in the corpus are relevant to the task and some of these are not semantically modeled. However, the baseline learner doesn't make this distinction and induces many erroneous entries. With voting the model better handles such cases, either by pairing such words with semantically empty entries or learning no entries for them. During inference the system can then easily skip such words.			
now	28	0	~~$AP : \lambda a.\text{len(a, 3)}$~~ ~~$AP : \lambda a.\text{direction(a, forward)}$~~
only	38	0	~~$N/NP : \lambda x.\lambda y.\text{intersect(y, x)}$~~ ~~$N/NP : \lambda x.\lambda y.\text{front(y, x)}$~~
here	31	8	~~$NP : \text{you}$~~ $S/S : \lambda x.x$ ~~$S\backslash N : \lambda f.\text{intersect(you, }\mathcal{A}\text{(f))}$~~
Without pruning the learner often over-splits multiword phrases and has no way to reverse such decisions.			
coat	25	0	~~$N : \lambda x.\text{intersection(x)}$~~ ~~$ADJ : \lambda x.\text{hatrack(x)}$~~
rack	45	0	~~$N : \lambda x.\text{hatrack(x)}$~~ ~~$N : \lambda x.\text{furniture(x)}$~~
coat rack	55	5	$N : \lambda x.\text{hatrack(x)}$ ~~$N : \lambda x.\text{wall(x)}$~~ ~~$N : \lambda x.\text{furniture(x)}$~~
Voting helps to avoid learning entries for rare words when the learning signal is highly ambiguous.			
orange	20	0	~~$N : \lambda x.\text{cement(x)}$~~ ~~$N : \lambda x.\text{grass(x)}$~~
pics of towers	26	0	~~$N\lambda x.\text{intersection(x)}$~~ ~~$N : \lambda x.\text{hall(x)}$~~

Table 3: Example entries from a learned ORACLE corpus lexicon using batch learning. For each string we report the number of lexical entries without voting (CONSENSUSVOTE) and pruning and with, and provide a few examples. Struck entries were successfully avoided when using voting and pruning.

decreases from 16.77 for online training to 8.11.

Finally, the total computational cost of our approach is roughly equivalent to online approaches. In both approaches, each pass over the data makes the same number of inference calls, and in practice, Artzi and Zettlemoyer (2013b) used 6-8 iterations for online learning while we used 10. A benefit of the batch method is its insensitivity to data ordering, as expressed by the lower standard deviation between randomized runs in Table 2.[6]

7 Related Work

There has been significant work on learning for semantic parsing. The majority of approaches treat grammar induction and parameter estimation separately, e.g. Wong and Mooney (2006), Kate and Mooney (2006), Clarke et al. (2010), Goldwasser et al. (2011), Goldwasser and Roth (2011), Liang et al. (2011), Chen and Mooney (2011), and Chen (2012). In all these approaches the grammar structure is fixed prior to parameter estimation.

Zettlemoyer and Collins (2005) proposed the learning regime most related to ours. Their learner alternates between batch lexical induction and online parameter estimation. Our learning algorithm design combines aspects of previously studied approaches into a batch method, including gradient updates (Kwiatkowski et al., 2010) and using weak supervision (Artzi and Zettlemoyer, 2011). In contrast, Artzi and Zettlemoyer (2013b) use online perceptron-style updates to optimize a margin-based loss. Our work also focuses on CCG lexicon induction but differs in the use of corpus-level statistics through voting and pruning for explicitly controlling the size of the lexicon.

Our approach is also related to the grammar induction algorithm introduced by Carroll and Char-

[6] Results still vary slightly due to multi-threading.

niak (1992). Similar to our method, they process the data using two batch steps: the first proposes grammar rules, analogous to our step that generates lexical entries, and the second estimates parsing parameters. Both methods use pruning after each iteration, to remove unused entries in our approach, and low probability rules in theirs. However, while we use global voting to add entries to the lexicon, they simply introduce all the rules generated by the first step. Their approach also relies on using disjoint subsets of the data for the two steps, while we use the entire corpus.

Using voting to aggregate evidence has been studied for combining decisions from an ensemble of classifiers (Ho et al., 1994; Van Erp and Schomaker, 2000). MAXVOTE is related to approval voting (Brams and Fishburn, 1978), where voters are required to mark if they approve each candidate or not. CONSENSUSVOTE combines ideas from approval voting, Borda counting, and instant-runoff voting. Van Hasselt (2011) described all three systems and applied them to policy summation in reinforcement learning.

8 Conclusion

We considered the problem of learning for semantic parsing, and presented voting and pruning methods based on corpus-level statistics for inducing compact CCG lexicons. We incorporated these techniques into a batch modification of an existing learning approach for joint lexicon induction and parameter estimation. Our evaluation demonstrates that both voting and pruning contribute towards learning a compact lexicon and illustrates the effect of lexicon quality on task performance.

In the future, we wish to study various aspects of learning more robust lexicons. For example, in our current approach, words not appearing in the training set are treated as unknown and ignored at inference time. We would like to study the benefit of using large amounts of unlabeled text to allow the model to better hypothesize the meaning of such previously unseen words. Moreover, our model's performance is currently sensitive to the set of seed lexical templates provided. While we are able to learn the meaning of new words, the model is unable to correctly handle syntactic and semantic structures not covered by the seed templates. To alleviate this problem, we intend to further explore learning novel lexical templates.

Acknowledgements

We thank Kuzman Ganchev, Emily Pitler, Luke Zettlemoyer, Tom Kwiatkowski and Nicholas FitzGerald for their comments on earlier drafts, and the anonymous reviewers for their valuable feedback. We also wish to thank Ryan McDonald and Arturas Rozenas for their valuable input about voting procedures.

References

Yoav Artzi and Luke S. Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yoav Artzi and Luke S. Zettlemoyer. 2013a. UW SPF: The University of Washington Semantic Parsing Framework.

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

Steven J. Brams and Peter C. Fishburn. 1978. Approval voting. *The American Political Science Review*, pages 831–847.

Qingqing Cai and Alexander Yates. 2013. Semantic parsing freebase: Towards open-domain semantic parsing. In *Proceedings of the Joint Conference on Lexical and Computational Semantics*.

Gelnn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. *Working Notes of the Workshop Statistically-Based NLP Techniques*.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

David L. Chen. 2012. Fast online lexicon learning for grounded language acquisition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Conference on Computational Natural Language Learning*.

Dan Goldwasser and Dan Roth. 2011. Learning from natural instructions. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *Proceedings of the Association of Computational Linguistics*.

Tin K. Ho, Jonathan J. Hull, and Sargur N. Srihari. 1994. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 66–75.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Joohyun Kim and Raymond J. Mooney. 2012. Unsupervised pcfg induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Joohyun Kim and Raymond J. Mooney. 2013. Adapting discriminative reranking to grounded language learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Jayant Krishnamurthy and Tom Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceedings of the Human Language Technology Conference of the North American Association for Computational Linguistics*.

Tom Kwiatkowski, Luke S. Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Tom Kwiatkowski, Luke S. Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical Generalization in CCG Grammar Induction for Semantic Parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1(1):179–192.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Matt MacMahon, Brian Stankiewics, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, action in route instructions. In *Proceedings of the National Conference on Artificial Intelligence*.

Cynthia Matuszek, Nicholas FitzGerald, Luke S. Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *Proceedings of the International Conference on Machine Learning*.

Mark Steedman. 1996. *Surface Structure and Interpretation*. The MIT Press.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press.

Merijn Van Erp and Lambert Schomaker. 2000. Variants of the borda count method for combining ranked classifier hypotheses. In *In the International Workshop on Frontiers in Handwriting Recognition*.

Hado Van Hasselt. 2011. *Insights in Reinforcement Learning: formal analysis and empirical evaluation of temporal-difference learning algorithms*. Ph.D. thesis, University of Utrecht.

Yuk W. Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Association for Computational Linguistics*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Morpho-syntactic Lexical Generalization for CCG Semantic Parsing

Adrienne Wang
Computer Science & Engineering
University of Washington
Seattle, WA
axwang@cs.washington.edu

Tom Kwiatkowski
Allen Institute for AI
Seattle, WA
tomk@allenai.org

Luke Zettlemoyer
Computer Science & Engineering
University of Washington
Seattle, WA
lsz@cs.washington.edu

Abstract

In this paper, we demonstrate that significant performance gains can be achieved in CCG semantic parsing by introducing a linguistically motivated grammar induction scheme. We present a new morpho-syntactic factored lexicon that models systematic variations in morphology, syntax, and semantics across word classes. The grammar uses domain-independent facts about the English language to restrict the number of incorrect parses that must be considered, thereby enabling effective learning from less data. Experiments in benchmark domains match previous models with one quarter of the data and provide new state-of-the-art results with all available data, including up to 45% relative test-error reduction.

1 Introduction

Semantic parsers map sentences to formal representations of their meaning (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Liang et al., 2011). One common approach is to induce a probabilistic CCG grammar, which defines the meanings of individual words and phrases and how to best combine them to analyze complete sentences. There has been recent work developing learning algorithms for CCG semantic parsers (Kwiatkowski et al., 2010; Artzi and Zettlemoyer, 2011) and using them for applications ranging from question answering (Cai and Yates, 2013b; Kwiatkowski et al., 2013) to robot control (Matuszek et al., 2012; Krishnamurthy and Kollar, 2013).

One key learning challenge for this style of learning is to induce the CCG lexicon, which lists possible meanings for each phrase and defines a set of possible parses for each sentence. Previous approaches have either hand-engineered a small set of lexical templates (Zettlemoyer and Collins, 2005, 2007) or automatically learned such templates (Kwiatkowski et al., 2010, 2011). These methods are designed to learn grammars that overgenerate; they produce spurious parses that can complicate parameter estimation.

In this paper, we demonstrate that significant gains can instead be achieved by using a more constrained, linguistically motivated grammar induction scheme. The grammar is restricted by introducing detailed syntactic modeling of a wider range of constructions than considered in previous work, for example introducing explicit treatments of relational nouns, Davidsonian events, and verb tense. We also introduce a new lexical generalization model that abstracts over systematic morphological, syntactic, and semantic alternations within word classes. This includes, for example, the facts that verbs in relative clauses and nominal constructions (e.g., "flights departing NYC" and "departing flights") should be infinitival while verbs in phrases (e.g., "What flights depart at noon?") should have tense. These grammar modeling techniques use universal, domain-independent facts about the English language to restrict word usage to appropriate syntactic contexts, and as such are potentially applicable to any semantic parsing application.

More specifically, we introduce a new *morpho-syntactic*, factored CCG lexicon that imposes our grammar restrictions during learning. Each lexical entry has (1) a word stem, automatically constructed from Wiktionary, with part-of-speech and morphological attributes, (2) a lexeme that is learned

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1284–1295,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

and pairs the stem with semantic content that is invariant to syntactic usage, and (3) a lexical template that specifies the remaining syntactic and semantic content. The full set of templates is defined in terms of a small set of base templates and template transformations that model morphological variants such as passivization and nominalization of verbs. This approach allows us to efficiently encode a general grammar for semantic parsing while also eliminating large classes of incorrect analyses considered by previous work.

We perform experiments in two benchmark semantic parsing datasets: GeoQuery (Zelle and Mooney, 1996) and ATIS (Dahl et al., 1994). In both cases, our approach achieves state-of-the-art performance, including a nearly 45% relative error reduction on the ATIS test set. We also show that the gains increase with less data, including matching previous model's performance with less than 25% of the training data. Such gains are particularly practical for semantic parsers; they can greatly reduce the amount of data that is needed for each new application domain.

2 Related Work

Grammar induction methods for CCG semantic parsers have either used hand-engineered lexical templates, e.g. (Zettlemoyer and Collins, 2005, 2007; Artzi and Zettlemoyer, 2011), or algorithms to learn such templates directly from data, e.g. (Kwiatkowski et al., 2010, 2011). Here, we extend the first approach, and show that better lexical generalization provides significant performance gains.

Although CCG is a common choice for semantic parsers, many other formalisms have been studied, including DCS trees (Liang et al., 2011), integer linear programs (Clarke et al., 2010), and synchronous grammars (Wong and Mooney, 2007; Jones et al., 2012; Andreas et al., 2013). All of these approaches build complete meaning representations for individual sentences, but the data we use has also been studied in related work on cross-sentence reasoning (Miller et al., 1996; Zettlemoyer and Collins, 2009) and modeling semantic interpretation as a tagging problem (Tur et al., 2013; Heck et al., 2013). Although we focus on full analysis with CCG, the general idea of using linguistic constraints to improve learning is broadly applicable.

Semantic parsers are also commonly learned from a variety of different types of supervision, including logical forms (Kate and Mooney, 2006; Wong and Mooney, 2007; Muresan, 2011; Kwiatkowski et al., 2012), question-answer pairs (Clarke et al., 2010; Liang et al., 2011), conversational logs (Artzi and Zettlemoyer, 2011), distant supervision (Krishnamurthy and Mitchell, 2012; Cai and Yates, 2013b), sentences paired with system behavior (Goldwasser and Roth, 2011; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013b), and even from database constraints with no explicit semantic supervision (Poon, 2013). We learn from logical forms, but CCG learning algorithms have been developed for each case above, making our techniques applicable.

There has been significant related work that influenced the design of our morpho-syntactic grammars. This includes linguistics studies of relational nouns (Partee and Borschev, 1998; de Bruin and Scha, 1988), Davidsonian events (Davidson, 1967), parsing as abduction (Hobbs et al., 1988), and other more general theories for lexicons (Pustejovsky, 1991) and CCG (Steedman, 2011). It also includes work on using morphology in CCG syntactic parsing (Honnibal et al., 2010) and more broad-coverage semantics in CCG (Bos, 2008; Lewis and Steedman, 2013). However, our work is unique in studying the use of related ideas for semantic parsing.

Finally, there has also been recent progress on semantic parsing against large, open domain databases such as Freebase (Cai and Yates, 2013a; Kwiatkowski et al., 2013; Berant et al., 2013). Unfortuantely, existing Freebase datasets are not a good fit to test our approach because the sentences they include have relatively simple structure and can be interepreted accurately using only factoid lookups with no database joins (Yao and Van Durme, 2014). Our work focuses on learning more syntactically rich models that support compositional reasoning.

3 Background

Lambda Calculus We represent the meanings of sentences, words and phrases with

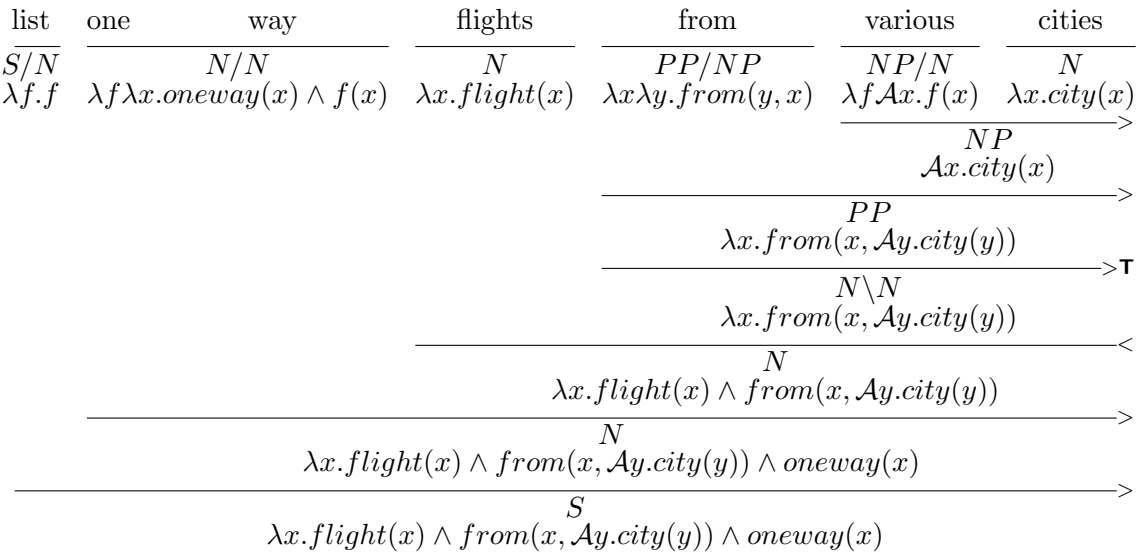


Figure 1: An example CCG parse.

lambda calculus logical expressions. We use a version of the typed lambda calculus (Carpenter, 1997), in which the basic types include entities, events, truth values and numbers. Function types are assigned to lambda expressions. The expression $\lambda x.flight(x)$ with type $\langle e, t\rangle$ takes an entity and returns a truth value, and represents a set of flights.

Combinatory Categorial Grammar CCG (Steedman, 1996, 2000) is a formalism that tightly couples syntax and semantics, and can be used to model a wide range of linguistic phenomena. A traditional CCG grammar includes a lexicon Λ with lexical entries like the following:

$$\begin{aligned}&flights \vdash N : \lambda x.flight(x)\\ &from \vdash PP/NP : \lambda y.\lambda x.from(x, y)\\ &cities \vdash N : \lambda x.city(x)\end{aligned}$$

where a lexical item $w \vdash X : h$ has words w, syntactic category X, and logical expression h.

CCG uses a small set of *combinatory rules* to jointly build syntactic parses and semantic representations. Two common combinatory rules are forward ($>$) and backward ($<$) *application*:

$$\begin{array}{llll} X/Y : f & Y : g & \Rightarrow & X : f(g) \quad (>)\\ Y : g & X\backslash Y : f & \Rightarrow & X : f(g) \quad (<)\end{array}$$

CCG also includes combinatory rules of forward ($>$ **B**) and backward ($<$ **B**) *composition*:

$$\begin{array}{ll} X/Y : f \quad Y/Z : g \Rightarrow X/Z : \lambda x.f(g(x)) & (> \mathbf{B})\\ Y\backslash Z : g \quad X\backslash Y : f \Rightarrow X\backslash Z : \lambda x.f(g(x)) & (< \mathbf{B})\end{array}$$

These rules apply to build syntactic and semantic derivations concurrently.

In this paper, we also implement type raising rules for compact representation of PP (prepositional phrase) and AP (adverbial phrase).

$$\begin{array}{ll} PP : g \Rightarrow N\backslash N : \lambda f\lambda x.f(x) \wedge g(x) & (\mathbf{T})\\ AP : g \Rightarrow S\backslash S : \lambda f\lambda e.f(e) \wedge g(e) & (\mathbf{T})\\ AP : g \Rightarrow S/S : \lambda f\lambda e.f(e) \wedge g(e) & (\mathbf{T})\end{array}$$

Figure 1 shows an example CCG parse (Steedman, 1996, 2000) where the lexical entries are listed across the top and the output lambda-calculus meaning representation is at the bottom. This meaning is a function (denoted by $\lambda x...$) that defines a set of flights with certain properties and includes a generalized Skolem constant (Steedman, 2011) ($\mathcal{A}y...$) that performs existential quantification. Following recent work (Artzi and Zettlemoyer, 2013b), we use meaning representations that model a variety of linguistic constructions, for example including Skolem constants for plurals and Davidson quantifiers for events, which we will introduce briefly throughout this paper as they appear.

Weighted CCGs A weighted CCG grammar is defined as $G = (\Lambda, \Theta)$, where Λ is a CCG lexicon and $\Theta \in \mathbb{R}^d$ is a d-dimensional parameter vector, which will be used to rank the parses allowed under Λ.

For a sentence x, G produces a set of candi-

date parse trees $Y = \mathsf{Y}(x; G)$. Given a feature vector $\Phi \in \mathbb{R}^d$, each parse tree y for sentence x is scored by $\mathsf{S}(y; \Theta) = \theta \cdot \phi(x, y)$. The output logical form $\hat{z}$ is then defined to be at the root of the highest-scoring parse $\hat{y}$:

$$\hat{y} = \arg \max_{y \in \mathsf{Y}(x;G)} \mathsf{S}(y; \Theta) \quad (1)$$

We use existing CKY-style parsing algorithms for this computation, implemented with UW SPF (Artzi and Zettlemoyer, 2013a). Section 7 describes the set of features we use in the learned models.

Learning with GENLEX We will also make use of an existing learning algorithm (Zettlemoyer and Collins, 2007) (ZC07). We first briefly review the ZC07 algorithm, and describe our modifications in Section 7.

Given a set of training examples $D = \{(x_i, z_i) : i = 1...n\}$, x_i being the ith sentence and z_i being its annotated logical form, the algorithm learns a set of parameters Θ for the grammar, while also inducing the lexicon Λ.

The ZC07 learning algorithm uses a function $\mathsf{GENLEX}(x, z)$ to define a set of lexical entries that could be used to parse the sentence x to construct the logical form z. For each training example (x, z), $\mathsf{GENLEX}(x, z)$ maps all substrings x to a set of potential lexical entries, generated by exhaustively pairing the logical constants in z using a set of hand-engineered templates. The example is then parsed with this much bigger lexicon and lexical entries from the highest scoring parses are added to Λ. The parameters Θ used to score parses are updated using a perceptron learning algorithm.

4 Morpho-Syntactic Lexicon

This section defines our morpho-syntactic lexical formalism. Table 1 shows examples of how lexemes, templates, and morphological transformations are used to build lexical entries for example verbs. In this section, we formally define each of these components and show how they are used to specify the space of possible lexical entries that can be built for each input word. In the following two sections, we will provide more discussion of the complete sets of templates (Section 5) and transformations (Section 6).

Verb, Noun, Preposition, Pronoun, Adjective, Adverb, Conjunction, Numeral, Symbol, Proper Noun, Interjection, Expression

Table 2: Part-of-Speech types

POS	**A**ttribute	**V**alues
Noun	Number	singular, plural
Verb	Person	first, second, third
Verb	Voice	active, passive
Verb	Tense	present, past
Verb	Aspect	simple, progressive, perfect
Verb	Participle	present participle, past participle
Adj, Adv, Det	Degree of comparison	comparative, superlative

Table 3: Morphological attributes and values.

We build on the factored CCG lexicon introduced by Kwiatkowski et al. (2011) but (a) further generalize lexemes to represent word stems, (b) constrain the use of templates with widely available syntactic information, and (c) efficiently model common morphological variations between related words.

The first step, given an input word w, is to do morphological and part-of-speech analysis with the **morpho-syntactic function** F. F maps a word to a set of possible morpho-syntactic representations, each containing a triple (s, p, m) of word stem s, part-of-speech p and morphological category m. For example, F maps the word $flies$ to two possible representations:

$$F(flies) = \{(fly, \text{Noun}, (plural)),$$
$$(fly, \text{Verb}, (third, singular, simple, present))\}$$

for the plural noun and present-tense verb senses of the word. F is defined based on the stems, part-of-speech types, and morphological attributes marked for each definition in Wiktionary. [1] The full sets of possible part-of-speech and morphological types required for our domains are shown in Table 2 and Table 3.

Each morpho-syntactic analysis $a \in F(w)$ is then paired with lexemes based on stem match. A **lexeme** $(s, \vec{c})$ pairs a word stem s with a list of logical constants $\vec{c} = [c_1 \dots c_k]$. Table 1 shows the words 'depart', 'departing', 'departure', which are all assigned the lexeme $(depart, [depart])$. In general, there can

[1] www.wiktionary.com

Word	Lexeme : Base Template	Trans	Lexical entry
depart	$(depart, [depart])$:	I	$depart \vdash S\backslash NP : \lambda x\lambda e.depart(e, x)$
departing		I	$departing \vdash S\backslash NP : \lambda x\lambda e.depart(e, x)$
departing	$\xi \vdash S\backslash NP : \lambda x\lambda e.v_1(e, x)$	f_{pres}	$departing \vdash PP : \lambda x\lambda e.depart(e, x)$
departure		f_{nom}	$departure \vdash N : \lambda x\lambda e.depart(e, x)$
use	$(use, [airline])$:	I	$use \vdash S\backslash NP/NP : \lambda x\lambda y\lambda e.airline(e, y, x)$
using		I	$using \vdash S\backslash NP/NP : \lambda x\lambda y\lambda e.airline(e, y, x)$
using	$\xi \vdash S\backslash NP/NP : \lambda x\lambda y\lambda e.v_1(e, y, x)$	f_{pres}	$using \vdash PP/NP : \lambda x\lambda e.airline(e, y, x)$
use		f_{nom}	$use \vdash N/NP : \lambda x\lambda y\lambda e.airline(e, y, x)$

Trans	Template Transformation
f_{pres}	$\xi \vdash S\backslash NP/\mathcal{T} : \lambda x_1..x_n\lambda e.v(e, x_n..x_1) \rightarrow \xi \vdash PP/\mathcal{T} : \lambda x_1..x_n\lambda e.v(e, x_n..x_1)$
f_{nom}	$\xi \vdash S\backslash NP/\mathcal{T} : \lambda x_1..x_n\lambda e.v(e, x_n..x_1) \rightarrow \xi \vdash N/\mathcal{T} : \lambda x_1..x_n\lambda e.v(e, x_n..x_1)$

Table 1: Lexical entries constructed by combining a *lexeme*, *base template*, and *transformation* for the intransitive verb 'depart' and the transitive verb 'use'.

be many different lexemes for each stem, that vary in the selection of which logical constants are included.

Given analysis (s, p, m) and lexeme $(s, \vec{c})$, we can use a **lexical template** to construct a **lexical entry**. Each template has the form:

$$\lambda(\xi, \vec{v}).[\xi \vdash X : h_{\vec{v}}]$$

where ξ and $\vec{v}$ are variables that abstract over the words and logical constants that will be used to define a lexical entry with syntax X and templated logical form $h_{\vec{v}}$.

To instantiate a template, ξ is filled with the original word w and the constants in $\vec{c}$ replace the variables $\vec{v}$. For example, the template $\lambda(\xi, \vec{v}).[\xi \vdash S\backslash NP : \lambda x\lambda e.v_1(e, x)]$ could be used with the word 'departing' and the lexeme $(depart, [depart])$ to produce the lexical entry $departing \vdash S\backslash NP : \lambda x\lambda e.depart(e, x)$. When clear from context, we will omit the function signature $\lambda_p(\xi, \vec{v})$. for all templates, as seen in Table 1.

In general, there can be many applicable templates, which we organize as follows. Each final template is defined by applying a **morphological transformation** to one of a small set of possible **base templates**. The pairing is found based on the morphological analysis (s, p, m), where each base template is associated with part-of-speech p and each transformation is indexed by the morphology m. A transformation f_m is a function:

$$\mathsf{f}_m(\lambda_p(\xi, \vec{v}).[\xi \vdash X : h_{\vec{v}}]) = \lambda_p(\xi, \vec{v}).[\xi \vdash X' : h'_{\vec{v}}]$$

that takes the base template as input and produces a new template to model the inflected form specified by m.

For example, both base templates in Table 1 are for verbs. The template $\xi \vdash S\backslash NP : \lambda x\lambda e.v_1(e, x)$ can be translated into three other templates based on the transformations I, f_{pres}, and f_{nom}, depending on the analysis of the original words. These transformations generalize across word type; they can be used for the transitive verb 'use' as well as the intransitive 'depart.' Each resulting template, potentially including the original input if the identity transformation I is available, can then be used to make an output lexical entry, as we described above.

5 Lexical Templates

The templates in our lexicon, as introduced in Section 4, model the syntactic and semantic aspects of lexical entries that are shared within each word class. Previous approaches have also used hand-engineered lexical templates, as described in Section 2, but we differ by (1) using more templates allowing for more fine grained analysis and (2) using word class information to restrict template use, for example ensuring that words which cannot be verbs are never paired with templates designed for verbs. This section describes the templates used during learning, first presenting those designed to model grammatical sentences and then a small second set designed for more elliptical spoken utterances.

Base Forms Table 4 lists the primary template set, where each row shows an example with a sentence illustrating its use. Templates are also grouped by the word classes, including adjectives, adverbs, prepositions, and several types of nouns and verbs. While there is not enough space to discuss each row, it is worth

word class	example usage	base template
Noun phrase	Boston	$\xi \vdash NP : v$
Noun (regular)	What **flight** is provided by delta?	$\xi \vdash N : \lambda x.v(x)$
Noun (relation)	I need **fares** of flights	$\xi \vdash N/PP : \lambda x\lambda y.v(x,y)$
	delta **schedule**	$\xi \vdash N\backslash(N/N) : \lambda f\lambda x.v(\mathcal{A}y.f(\lambda z.true, y), x)$
Noun (function)	**size** of California	$\xi \vdash NP/NP : \lambda x.v(x)$
$V_{intrans}$	What flights **depart** from New York?	$\xi \vdash S\backslash NP : \lambda x\lambda e.v(e,x)$
V_{trans}	Which airlines **serve** Seattle (active verb)	$\xi \vdash S\backslash NP/NP : \lambda x\lambda y\lambda e.v(e,y,x)$
	What airlines **have** flights (passive verb)	$\xi \vdash S\backslash NP/NP : \lambda x\lambda y\lambda e.v(e,x,y)$
$V_{ditrans}$	They **give** him a book	$\xi \vdash S\backslash NP/NP/NP : \lambda x\lambda y\lambda z\lambda e.v(e,z,y,x)$
$V_{imperson}$	It **costs** $500 to fly to Boston	$\xi \vdash S\backslash NP/NP/NP : \lambda x\lambda y\lambda z\lambda e.v(e,y,x)$
V_{aux}	The flights **have** arrived at Boston	$\xi \vdash S\backslash NP/(S\backslash NP) : \lambda f.f$
		$\xi \vdash S/NP/(S/NP) : \lambda f.f$
	Does delta provide flights from Seattle?	$\xi \vdash S/S : \lambda f.f$
V_{copula}	The flights **are** from Boston	$\xi \vdash S\backslash NP/PP : \lambda f\lambda x.f(x)$
	What flight **is** cheap?	$\xi \vdash S\backslash NP/(N/N) : \lambda f\lambda x.f(\lambda y.true, x)$
	Alaska **is** the state with the most rivers	$\xi \vdash S\backslash NP/NP : \lambda x\lambda y.equals(y,x)$
Adjective	I need a **one way** flight	$\xi \vdash N/N : \lambda f\lambda x.f(x) \wedge v(x)$
	Boston flights **round trip**	$\xi \vdash PP : \lambda x.v(x)$
	How **long** is mississippi?	$\xi \vdash DEG : \lambda x.v(x)$
Preposition	List flights **from** Boston	$\xi \vdash PP/NP : \lambda x\lambda y.v(y,x)$
	List flights that go **to** Dallas	$\xi \vdash AP/NP : \lambda x\lambda e.v(e,x)$
	List flights **between** Dallas and Boston	$\xi \vdash PP/NP/NP : \lambda x\lambda y\lambda z.v_1(z,x) \wedge v_2(z,y)$
	What flights leave **between** 8am and 9am?	$\xi \vdash AP/NP/NP : \lambda x\lambda y\lambda e.v_1(e,x) \wedge v_2(e,y)$
Adverb	Which flight departs **daily**?	$\xi \vdash AP : \lambda e.v(e)$
	How **early** does the flight arrive?	$\xi \vdash DEG : \lambda x.v(x)$
Determiner	Which airline has **a** flight from Boston?	$\xi \vdash NP/N : \lambda f\mathcal{A}x.f(x)$

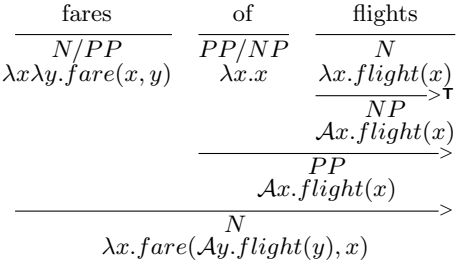
Table 4: Base templates that define different syntactic roles.

type	example usage	base template
$t_{elliptical}$	flights **Newark** to Cleveland	$\xi \vdash PP : \lambda x.P(x,v)$
	flights arriving **2pm**	$\xi \vdash AP : \lambda e.P(e,v)$
	american airline from Denver	$\xi \vdash N : \lambda x.P(x,v)$
$t_{metonymy}$	List **airlines** from Seattle	$\xi \vdash N/PP : \lambda f\lambda x.v(x) \wedge P(\mathcal{A}y.f(y),x))$
	Shat **airlines** depart from Seattle?	$\xi \vdash N/(S\backslash NP) : \lambda f\lambda x.v(x) \wedge P(\mathcal{A}y.f(y),x)$
	fares from miami to New York	$\xi \vdash N/PP : \lambda f\lambda x.v(\mathcal{A}y.f(y),x)$

Table 5: Base templates for ungrammatical linguistic phenomena

considering nouns as an illustrative example.

We model nouns as denoting a set of entities that satisfy a given property. Regular nouns are represented using unary predicates. Relational nouns syntactically function as regular nouns but semantically describe sets of entities that have some relationship with a complement (Partee and Borschev, 1998). For example, the relational noun *fare* describes a binary relationship between flights and their price information, as we see in this parse:



This analysis differs from previous approaches (Zettlemoyer and Collins, 2007), where relational nouns were treated as regular nouns and prepositions introduced the binary relationship. The relational noun model reduces lexical ambiguity for the prepositions, which are otherwise highly polysemous.

Adjectives are nominal modifiers that take a noun or a noun phrase as an argument and add properties through conjunction. Prepositions take nominal objects and function as adjectival modifiers for nouns or adverbial modifiers for verbs. Verbs can be subcategorized by their grammatical structures into transitive (V_{trans}), intransitive ($V_{intrans}$), impersonal ($V_{imperson}$), auxiliary (V_{aux}) and copula (V_{copula}). Adverbs are verb modifiers defining aspects like time, rate and duration. The adoption of event semantics allows adverbial modifiers to be represented by predicates and

linked by the shared events. Determiners precede nouns or noun phrases and distinguish a reference of the noun. Following the generalized Skolem terms, we model determiners, including indefinite and definite articles, as a $\langle\langle e,t\rangle,e\rangle$ function that selects a unique individual from a $\langle e,t\rangle$-typed function defining a singleton set.

Missing Words The templates presented so far model grammatically correct input. However, in dialogue domains such as ATIS, speakers often omit words. For example, speakers can drop the preposition "from" in "flights from Newark to Cleveland" to create the elliptical utterance "flights Newark to Cleveland". We address this issue with the templates $t_{elliptical}$ illustrated in Table 5. Each of these adds a binary relation P to a lexeme with a single entity typed constant. For our example, the word "Newark" could be assigned the lexical item $Newark \vdash PP : \lambda x.from(x, newark)$ by selecting the first template and $P = from$.

Another common problem is the use of metonymy. In the utterance "What airlines depart from New York?", the word "airlines" is used to reference flight services operated by a specific airline. This is problematic because the word "depart" needs to modify an event of type *flight*. We solve this with the $t_{metonymy}$ templates in Table 5. These introduce a binary predicate P that would, in the case of our example, map airlines on to the flights that they operate.

The templates in Table 5 handle the major cases of missing words seen in our data and are more efficient than the approach taken by (Zettlemoyer and Collins, 2007) who introduced complex type shifting rules and relaxed the grammar to allow every word order.

6 Morphological Transformations

Finally, the morpho-syntactic lexicon introduces morphological transformations, which are functions from base lexical templates to lexical templates that model the syntactic and semantic variation as the word is inflected. These transformations allow us to compactly model, for example, the facts that argument order is reversed when moving from active to passive forms of the same verb, and that the subject can be omitted. To the best of our knowledge, we are the first to study such transformations for semantic parsing.

Table 6 shows the transformations. Each row groups a set of transformations by linguistic category, including singular vs. plural number, active vs. passive voice, and so on, and also includes example sentences where the output templates could be used. Again, for space, we do not detail the motivation for every class, but it is worth looking at some of the alternations for verbs and nouns as our prototypical example.

Some verbs can act as noun modifiers. For example, the present participle "using" modifies "flights" in "flights *using* twa". To model this variation, we use the transformation $\mathsf{f}_{present\_part}$, a mapping that changes the root of the verb signature $S\backslash NP$ to PP:

$$\mathsf{f}_{present\_part} : \xi \vdash S\backslash NP/\mathcal{T} : \lambda x_1..x_n\lambda e.v(e, x_n..x_1) \rightarrow \xi \vdash PP/\mathcal{T} : \lambda x_1..x_n\lambda e.v(e, x_n..x_1)$$

where $\mathcal{T} = [\epsilon, NP, NP/NP]$ instantiates this rule for verbs that take different sets of arguments, effectively allowing any verb that is in its finite or -ing form to behave syntactically like a prepositional phrase.

Intransitive present participles can also act as prenominal adjectival modifiers as in "the *departing* flight". We add a second mapping that maps the intransitive category $S\backslash NP$ to the noun modifier N/N.

$$\mathsf{f}_{present\_part} : \quad \xi \vdash S\backslash NP : \lambda x\lambda e.v(e, x) \rightarrow \quad \xi \vdash N/N : \lambda f\lambda x\lambda e.f(x) \wedge v(e, x)$$

Finally, verbal nouns have meanings derived from actions typically described by verbs but syntactically function as nouns. For example, *landing* in the phrase "landing from jfk" is the gerundive use of the verb *land*. We add the following mapping to $\mathsf{f}_{present\_part}$ and $\mathsf{f}_{nominal}$:

$$\xi \vdash S\backslash NP/\mathcal{T} : \lambda x_1..x_n\lambda e.v(e, x_n..x_1) \rightarrow \xi \vdash N/\mathcal{T} : \lambda x_1..x_n\lambda e.v(e, x_n..x_1)$$

with $\mathcal{T}$ from above. This allows for reuse of the same meaning across quite different syntactic constructs, including for example "flights that *depart* from Boston" and "*departure* from Boston."

Template transformations f_m	Example usage
Plural Number (f_{plural})	
I	**flight** → early **flights**
$\xi \vdash N : \lambda x.v(x) \rightarrow \xi \vdash NP : \mathcal{A}x.v(x)$	**city** → flights to **cities**
Singular Number ($\mathsf{f}_{singular}$)	
I	**flight** → **flight**
Possessive ($\mathsf{f}_{possess}$)	
$\xi \vdash NP : v \rightarrow \xi \vdash N/N : \lambda f \lambda x.f(x) \wedge P(x, v)$	**delta** → **delta's** flights
$\xi \vdash N : \lambda x.v(x) \rightarrow \xi \vdash N/N : \lambda f \lambda x.f(x) \wedge P(x, \mathcal{A}y.v(y))$	**airline** → **airline's** flights
Passive Voice ($\mathsf{f}_{passive}$)	
$\xi \vdash \mathcal{Y}/NP : \lambda x_1..x_n \lambda e.v(e, x_1..x_n) \rightarrow \xi \vdash \mathcal{Y}/PP : \lambda x_1..x_n \lambda e.v(e, x_n..x_1)$	**serves** →is **served** by
$\xi \vdash \mathcal{Y}/NP : \lambda x_1..x_n \lambda e.v(e, x_1, .., x_n) \rightarrow \xi \vdash \mathcal{Y} : \lambda x_1..x_{n-1} \lambda e.v(e, x_{n-1}..x_1)$	**name** →city **named** Austin
Present Participle ($\mathsf{f}_{present}$)	
$\xi \vdash S\backslash NP/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1) \rightarrow \xi \vdash PP/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1)$	**use** →flights **using** twa
$\xi \vdash S\backslash NP : \lambda x \lambda e.v(e, x) \rightarrow \xi \vdash N/N : \lambda f \lambda x \lambda e.f(x) \wedge v(e, x)$	**arrive** →**arriving** flights
$\xi \vdash S\backslash NP/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1) \rightarrow \xi \vdash N/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1)$	**land** → **landings** at jfk
Past Participle (f_{past})	
$\xi \vdash S\backslash NP/NP : \lambda x_1..x_n \lambda e.v(e, x_n..x_1) \rightarrow \xi \vdash PP/PP : \lambda x_1..x_n \lambda e.v(e, x_1..x_n)$	**use** → plane **used** by
Nominalization ($\mathsf{f}_{nominal}$)	
$\xi \vdash S\backslash NP/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1) \rightarrow \xi \vdash N/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1)$	**depart** → **departure**
Comparative (f_{comp})	
$\xi \vdash DEG : \lambda x.v(x) \rightarrow \xi \vdash PP/PP : \lambda x \lambda y.v(y) < v(x)$	**short** → **shorter**
$\xi \vdash DEG : \lambda x.v(x) \rightarrow \xi \vdash PP/PP : \lambda x \lambda y.v(y) > v(x)$	**long** → **longer**
Superlative (f_{super})	
$\xi \vdash DEG : \lambda x.v(x) \rightarrow \xi \vdash NP/N : \lambda f.argmin(\lambda x.f(x), \lambda x.v(x))$	**short** → **shortest**
$\xi \vdash DEG : \lambda x.v(x) \rightarrow \xi \vdash NP/N : \lambda f.argmax(\lambda x.f(x), \lambda x.v(x))$	**long** → **longest**

Table 6: Morphological transformations with examples. $\mathcal{T} = [\epsilon, NP, NP/NP]$ and $\mathcal{Y} = [S\backslash NP, S\backslash NP/NP]$ allow a single transformation to generalize across word type.

Nouns can be inflected by number to denote singular and plural forms or by adding an apostrophe to mark a possessive case. The transformation function $\mathsf{f}_{\mathsf{singular}}$ is an identity transformation. Plurals may have different interpretations: one is the generic $\langle e, t \rangle$ set representation, which requires no transformation on the base, or plurals can occur without overt determiners (bare plurals), but semantically imply quantification. We create a plural to singular type shifting rule which implements the $\langle \langle e, t \rangle, e \rangle$ skolem function to select a unique individual from the set. The possessive transformation $\mathsf{f}_{\mathsf{possess}}$ transfers the base template to a noun modifier, and adds a binary predicate P that encodes the relation.

There are also a number of instances of the identity transformation function I, which does not change the base template. Because the semantics we are constructing was designed to answer questions against a static database, it does not need to represent certain phenomena to return the correct answer. This includes more advanced variants of person, tense, aspect, and potentially many others. Ideally, these morphological attributes should add semantic modifiers to the base meaning, for example, tense can constrain the time at which an event occurs. However, none of our domains support such reasoning, so we assign the identity transformation, and leave the exploration of these issues to future work.

7 Learning

One advantage of our morpho-syntactic, factored lexicon is that it can be easily learned with small modifications to existing algorithms (Zettlemoyer and Collins, 2007). We only need to modify the GENLEX procedure that defines the space of possible lexical entries. For each training example (x, z), $\mathsf{GENLEX}(x, z, F)$ first maps each substring in the sentence x into the morphological representation (s, p, c) using F introduced in Section 4. A candidate lexeme set L' is then generated by exhaustively pairing the word stems with all subsets of the logical constants from z. Lexical templates are applied to the lexemes in L' to generate candidate lexical entries for x. Finally, the lexemes that participate in the top scoring correct parse of x are added to the permanent lexicon.

Initialization Following standard practice, we compile an initial lexicon Λ_0, which consists of a list of domain independent lexical

items for function words, such as interrogative words and conjunctions. These lexical items are mostly semantically vacuous and serve particular syntactic functions that are not generalizable to other word classes. We also initialize the lexemes with a list of NP entities complied from the database, e.g., (*Boston*, [*bos*]).

Features We use two types of features in the model for discriminating parses. Four *lexical* features are fired on each lexical item: $\phi_{(s,\vec{c})}$ for the lexeme, ϕ_{t_p} for the base template, ϕ_{t_m} for the morphologically modified template, and ϕ_l for the complete lexical item. We also compute the standard *logical expression* features (Zettlemoyer and Collins, 2007) on the root semantics to track the pairwise predicate-argument relations and the co-occuring predicate-predicate relations in conjunctions and disjunctions.

8 Experimental Setup

Data and Metrics We evaluate performance on two benchmark semantic parsing datasets, Geo880 and ATIS. We use the standard data splits, including 600/280 train/test for Geo880 and 4460/480/450 train/develop/test for ATIS. To support the new representations in Section 5, we systematically convert annotations with existential quantifiers, temporal events and relational nouns to new logical forms with equivalent meanings. All systems are evaluated with exact match accuracy, the percentage of fully correct logical forms.

Initialization We assign positive initial weights to the indicator features for entries in the initial lexicon, as defined in Section 7, to encourage their use. The elliptical template and metonymy template features are initialized with negative weights to initially discourage word skipping.

Comparison Systems We compare performance with all recent CCG grammar induction algorithms that work with our datasets. This includes methods that used a limited set of hand-engineered templates for inducing the lexicon, ZC05 (Zettlemoyer and Collins, 2005) and ZC07 (Zettlemoyer and Collins, 2007), and those that learned grammar structure by automatically splitting the labeled logical forms, UBL (Kwiatkowski et al., 2010) and FUBL (Kwiatkowski et al., 2011). We also compare the state-of-the-art for Geo880 (DCS (Liang et al., 2011) and DCS+ which includes an engineered seed lexicon) and ATIS (which is ZC07). Finally, we include results for GUSP (Poon, 2013), a recent unsupervised approach for ATIS.

System	Test
ZC05	79.3
ZC07	86.1
UBL	87.9
FUBL	88.6
DCS	87.9
FULL	90.4
DCS$^+$	91.1

Table 7: Exact-match Geo880 test accuracy.

System	Dev	Test
ZC07	74.4	84.6
UBL	65.6	71.4
FUBL	81.9	82.8
GUSP	-	83.5
TEMP-ONLY	85.5	87.2
FULL	87.5	91.3

Table 8: Exact-match accuracy on the ATIS development and test sets.

System Variants We report results for a complete approach (Full), and variants which use different aspects of the morpho-syntactic lexicon. The TEMP-ONLY variant learned with the templates from Section 5 but, like ZC07, does not use any word class information to restrict their use. The TEMP-POS removes morphology from the lexemes, but includes the word class information from Wiktionary. Finally, we also include DCS$^+$, which initialize a set of words with POS tag JJ, NN, and NNS.

9 Results

Full Models Tables 7 and 8 report the main learning results. Our approach achieves state-of-the-art accuracies on both datasets, demonstrating that our new grammar induction scheme provides a type of linguistically motivated regularization; restricting the algorithm to consider a much smaller hypothesis space allows to learn better models.

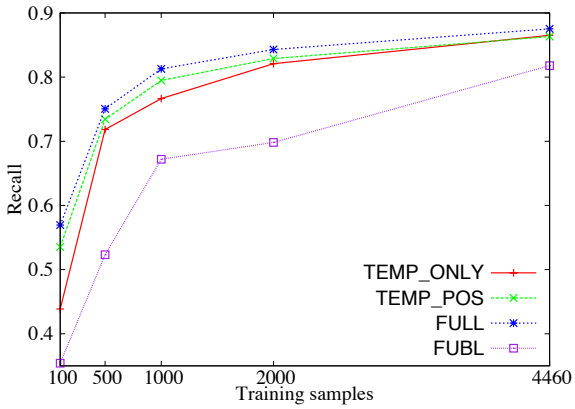


Figure 2: ATIS Learning Curve

On Geo880 the full method edges out the best systems by 2% absolute on the test set, as compared to other systems with no domain-specific lexical initialization. Although DCS requires less supervision, it also uses external signals including a POS tagger.

We see similarly strong results for ATIS, outperforming FUBL on the ATIS development set by 6.8%, and improving the accuracy on the test set by 7.9% over the previous best system ZC07. Unlike FUBL, which excels at the development set but trails ZC07's templated grammar by almost 2 points on the test set, our approach demonstrates consistent improvements on both. Additionally, although the unsupervised model (GUSP) rivals previous approaches, we are able to show that more careful use of supervision open a much wider performance gap.

Learning Curve with Ablations Figure 2 presents a learning curve for the ATIS domain, demonstrating that the learning improvements become even more dramatic for smaller training set sizes. Our model outperforms FUBL by wide margins, matching its final accuracy with only 22% of the total training examples. Our full model also consistently beats the variants with fewer word class restrictions, although by smaller margins. Again, these results further highlight the benefit of importing external syntactic resources and enforcing linguistically motivated constraints during learning.

Learned Lexicon The learned lexicon is also more compact. Table 9 summarizes statistics on unique lexical entries required to parse the ATIS development set. The morpho-syntactic model uses 80.3% of the lexical entries and 63.7% of the lexemes that FUBL needs, while increase performance by nearly 7 points. Upon inspection, our model achieves better lexical decomposition by learning shorter lexical units, for example, the adoption of Davidsonian events allows us to learn unambiguous adverbial modifiers, and the formal modeling of nominalized nouns and relational nouns treats prepositions as syntactic modifiers, instead of being encoded in the semantics. Such restrictions generalize to a much wider variety of syntactic contexts.

System	Lexical Entries	Lexemes
FUBL	1019	721
Our Approach	818	495

Table 9: Lexicon size comparison on the ATIS dev set (460 unique tokens).

10 Summary and Future Work

We demonstrated that significant performance gains can be achieved in CCG semantic parsing by introducing a more constrained, linguistically motivated grammar induction scheme. We introduced a morpho-syntactic factored lexicon that uses domain-independent facts about the English language to restrict the number of incorrect parses that must be considered and demonstrated empirically that it enables effective learning of complete parsers, achieving state-of-the-art performance.

Because our methods are domain independent they should also benefit other semantic parsing applications and other learning algorithms that use different types of supervision, as we hope to verify in future work. We would also like to study how to generalize these gains to languages other than English, by inducing more of the syntactic structure.

Acknowledgements

The research was supported by the NSF (IIS-1115966, IIS-1252835) and the Intel Center for Pervasive Computing at the Univeristy of Washington. The authors thank Robert Gens, Xiao Ling, Xu Miao, Mark Yatskar and the UW NLP group for helpful discussions, and the anonymous reviewers for helpful comments.

References

Andreas, J., Vlachos, A., and Clark, S. (2013). Semantic parsing as machine translation.

Artzi, Y. and Zettlemoyer, L. (2011). Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Artzi, Y. and Zettlemoyer, L. (2013a). UW SPF: The University of Washington Semantic Parsing Framework.

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

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on freebase from question-answer pairs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Bos, J. (2008). Wide-coverage semantic analysis with boxer. In *Proceedings of the Conference on Semantics in Text Processing*.

Cai, Q. and Yates, A. (2013a). Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Cai, Q. and Yates, A. (2013b). Semantic parsing freebase: Towards open-domain semantic parsing. In *Proceedings of the Joint Conference on Lexical and Computational Semantics*.

Carpenter, B. (1997). *Type-Logical Semantics*. The MITPress.

Chen, D. and Mooney, R. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

Clarke, J., Goldwasser, D., Chang, M., and Roth, D. (2010). Driving semantic parsing from the world's response. In *Proceedings of the Conference on Computational Natural Language Learning*.

Dahl, D. A., Bates, M., Brown, M., Fisher, W., Hunicke-Smith, K., Pallett, D., Pao, C., Rudnicky, A., and Shriberg, E. (1994). Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the workshop on Human Language Technology*.

Davidson, D. (1967). The logical form of action sentences. *Essays on actions and events*, pages 105–148.

de Bruin, J. and Scha, R. (1988). The interpretation of relational nouns. In *Proceedings of the Conference of the Association of Computational Linguistics*, pages 25–32. ACL.

Goldwasser, D. and Roth, D. (2011). Learning from natural instructions. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Heck, L., Hakkani-Tür, D., and Tur, G. (2013). Leveraging knowledge graphs for web-scale unsupervised semantic parsing. In *Proc. of the INTERSPEECH*.

Hobbs, J. R., Stickel, M., Martin, P., and Edwards, D. (1988). Interpretation as abduction. In *Proceedings of the Association for Computational Linguistics*.

Honnibal, M., Kummerfeld, J. K., and Curran, J. R. (2010). Morphological analysis can improve a ccg parser for english. In *Proceedings of the International Conference on Computational Linguistics*.

Jones, B. K., Johnson, M., and Goldwater, S. (2012). Semantic parsing with bayesian tree transducers. In *Proceedings of Association of Computational Linguistics*.

Kate, R. and Mooney, R. (2006). Using string-kernels for learning semantic parsers. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Krishnamurthy, J. and Kollar, T. (2013). Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1(2).

Krishnamurthy, J. and Mitchell, T. (2012). Weakly supervised training of semantic parsers. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Kwiatkowski, T., Choi, E., Artzi, Y., and

Zettlemoyer, L. (2013). Scaling semantic parsers with on-the-fly ontology matching.

Kwiatkowski, T., Goldwater, S., Zettlemoyer, L., and Steedman, M. (2012). A probabilistic model of syntactic and semantic acquisition from child-directed utterances and their meanings. *Proceedings of the Conference of the European Chapter of the Association of Computational Linguistics*.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2011). Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Lewis, M. and Steedman, M. (2013). Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Liang, P., Jordan, M., and Klein, D. (2011). Learning dependency-based compositional semantics. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Matuszek, C., FitzGerald, N., Zettlemoyer, L., Bo, L., and Fox, D. (2012). A joint model of language and perception for grounded attribute learning. In *Proceedings of the International Conference on Machine Learning*.

Miller, S., Stallard, D., Bobrow, R., and Schwartz, R. (1996). A fully statistical approach to natural language interfaces. In *Proceedings Association for Computational Linguistics*.

Muresan, S. (2011). Learning for deep language understanding. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Partee, B. H. and Borschev, V. (1998). Integrating lexical and formal sematics: Genitives, relational nouns, and type-shifting. In *Proceedings of the Second Tbilisi Symposium on Language, Logic, and Computation*.

Poon, H. (2013). Grounded unsupervised semantic parsing. In *Association for Computational Linguistics (ACL)*.

Pustejovsky, J. (1991). The generative lexicon. volume 17.

Steedman, M. (1996). *Surface Structure and Interpretation*. The MIT Press.

Steedman, M. (2000). *The Syntactic Process*. The MIT Press.

Steedman, M. (2011). *Taking Scope*. The MIT Press.

Tur, G., Deoras, A., and Hakkani-Tur, D. (2013). Semantic parsing using word confusion networks with conditional random fields. In *Proc. of the INTERSPEECH*.

Wong, Y. and Mooney, R. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Yao, X. and Van Durme, B. (2014). Information extraction over structured data: Question answering with freebase. In *Association for Computational Linguistics (ACL)*.

Zelle, J. and Mooney, R. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*.

Zettlemoyer, L. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Zettlemoyer, L. and Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Zettlemoyer, L. and Collins, M. (2009). Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*.

Semantic Parsing Using Content and Context: A Case Study from Requirements Elicitation

Reut Tsarfaty
Weizmann Institute
Rehovot, Israel

Ilia Pogrebezky
Interdisciplinary Center
Herzliya, Israel

Guy Weiss
Weizmann Institute
Rehovot, Israel

Yaarit Natan
Weizmann Institute
Rehovot, Israel

Smadar Szekely
Weizmann Institute
Rehovot, Israel

David Harel
Weizmann Institute
Rehovot, Israel

Abstract

We present a model for the automatic semantic analysis of requirements elicitation documents. Our target semantic representation employs *live sequence charts*, a multi-modal visual language for scenario-based programming, which can be directly translated into executable code. The architecture we propose integrates sentence-level and discourse-level processing in a generative probabilistic framework for the analysis and disambiguation of individual sentences in context. We show empirically that the discourse-based model consistently outperforms the sentence-based model when constructing a system that reflects all the static (entities, properties) and dynamic (behavioral scenarios) requirements in the document.

1 Introduction

Requirements elicitation is a process whereby a system analyst gathers information from a stakeholder about a desired system (software or hardware) to be implemented. The knowledge collected by the analyst may be *static*, referring to the conceptual model (the entities, their properties, the possible values) or *dynamic*, referring to the behavior that the system should follow (who does what to whom, when, how, etc). A stakeholder interested in the system typically has a specific static and dynamic domain in mind, but he or she cannot necessarily prescribe any formal models or code artifacts. The term *requirements elicitation* we use here refers to a piece of discourse in natural language, by means of which a stakeholder communicates their desiderata to the system analyst.

The role of a system analyst is to understand the different requirements and transform them into formal constructs, formal diagrams or executable code. Moreover, the analyst needs to consolidate the different pieces of information to uncover a single shared domain. Studies in software engineering aim to develop intuitive symbolic systems with which human agents can encode requirements that would then be unambiguously translated into a formal model (Fuchs and Schwitter, 1995; Bryant and Lee, 2002).

More recently, Gordon and Harel (2009) defined a natural fragment of English that can be used for specifying requirements which can be effectively translated into *live sequence charts* (LSC) (Damm and Harel, 2001; Harel and Marelly, 2003), a formal language for specifying the dynamic behavior of reactive systems. However, the grammar that underlies this language fragment is highly ambiguous, and all disambiguation has to be conducted manually by a human agent. Indeed, it is commonly accepted that the more natural a controlled language fragment is, the harder it is to develop an unambiguous translation mechanism (Kuhn, 2014).

In this paper we accept the ambiguity of requirements descriptions as a premise, and aim to answer the following question: can we automatically recover a formal representation of the complete system — one that *best* reflects the human-perceived interpretation of the entire document? Recent advances in natural language processing, with an eye to semantic parsing (Zettlemoyer and Collins, 2005; Liang et al., 2011; Artzi and Zettlemoyer, 2013; Liang and Potts, 2014), use different formalisms and various kinds of learning signals for statistical semantic parsing. In particular, the model of Lei et al. (2013) induces input parsers from format descriptions. However, rarely do these models take into account the entire document's context.

The key idea we promote here is that discourse context provides substantial disambiguating information for sentence analysis. We suggest a novel

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1296–1307,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

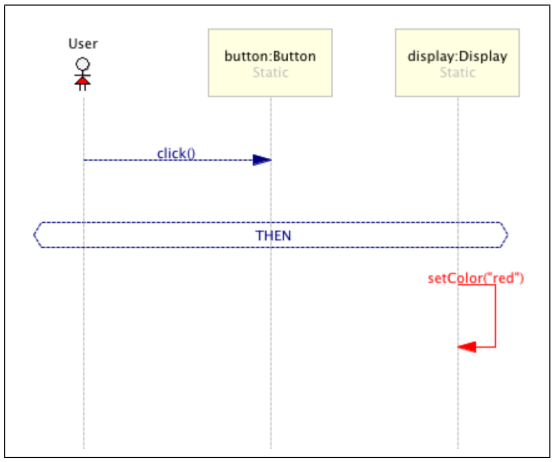


Figure 1: An LSC scenario: "When the user clicks the button, the display color must change to red."

model for integrated sentence-level and discourse-level processing, in a joint generative probabilistic framework. The input for the requirements elicitation task is given in a simplified, yet highly ambiguous, fragment of English, as specified in Gordon and Harel (2009). The output, in contrast, is a sequence of unambiguous and well-formed *live sequence charts* (LSC) (Damm and Harel, 2001; Harel and Marelly, 2003) describing the dynamic behavior of the system, tied to a single shared code-base called a *system model* (SM).

Our solution takes the form of a *hidden markov model* (HMM) where emission probabilities reflect the grammaticality and interpretability of textual requirements via a *probabilistic grammar* and transition probabilities model the overlap between SM snapshots of a single, shared, domain. Using efficient viterbi decoding, we search for the best sequence of domain snapshots that has most likely generated the entire requirements document. We empirically show that such an integrated model consistently outperforms a sentence-based model learned from the same set of data.

The remainder of this document is organized as follows. In Section 2 we describe the task, and spell out our formal assumptions concerning the input and the output. In Section 3 we present our target semantic representation and a specially tailored notion of *grounding* for anchoring the requirements in a code-base. In Section 4 we develop our sentence-based and discourse-based models, and in Section 5 we evaluate the models on various case studies. In Section 6 we discuss applications and future extensions, and in Section 7 we summarize and conclude.

2 Parsing Requirements Elicitation Documents: Task Description

There is an inherent discrepancy between the input and the output of the software engineering process. The input, system requirements, is specified in a natural, informal, language. The output, the system, is ultimately implemented in a formal unambiguous programming language. Can we automatically recover such a formal representation of a complete system from a set of requirements? In this work we explore this challenge empirically.

The Input. We assume a *scenario-based programming* paradigm (a.k.a *behavioral programming* (BP) (Harel et al., 2012)) in which system development is seen as a process whereby humans describe the expected behavior of the system by means of "short-stories", formally called *scenarios* (Harel, 2001). We further assume that a given requirements document describes exactly one system, and that each sentence describes a single, possibly complex, scenario. The requirements we aim to parse are given in a simplified form of English (specifically, the English fragment described in Gordon and Harel (2009)). Contrary to strictly formal specification languages, which are closed and unambiguous, this fragment of English employs an open-ended lexicon and exhibits extensive syntactic and semantic ambiguity.[1]

The Output. Our target semantic representation employs *live sequence charts* (LSC), a diagrammatic formal language for scenario-based programming (Damm and Harel, 2001). Formally, LSCs are an extension of the well-known UML message sequence diagrams (Harel and Maoz, 2006), and they have a direct translation into executable code (Harel and Marelly, 2003).[2] Using LSC diagrams for software modelling enjoys the advantages of being easily learnable (Harel and Gordon, 2009), intuitively interpretable (Eitan et al., 2011) and straightforwardly amenable to execution (Harel et al., 2002) and verification (Harel et al., 2013). The LSC language is particularly suited for representing natural language requirements, since its basic formal constructs, *scenarios*, nicely align with *events*, the primitive objects of Neo-Davidsonian Semantics (Parsons, 1990).

[1] Formally, this variant may be viewed as a CNL of degree P2 E3 N4 S4 with properties C,F,W,A (Kuhn, 2014, pp 6-12).

[2] It can be shown that the execution semantics of the LSC language is embedded in a fragment of a branching temporal logic called CTL* (Kugler et al., 2005).

Live Sequence Charts and Code Artifacts. A *live sequence chart* (LSC) is a diagram that describes a possible or necessary run of a specified system. In a single LSC diagram, entities are represented as vertical lines called *lifelines*, and interactions between entities are represented using horizontal arrows between lifelines called *messages*, connecting a sender to a receiver. Messages may refer to other entities (or properties of entities) as arguments. Time in LSCs proceeds from top to bottom, imposing a partial order on the execution of messages. LSC messages can be *hot* (red, "must happen") or *cold* (blue, "may happen"). A message may have an execution status, which designates it as *monitored* (dashed arrow, "wait for") or *executed* (full arrow, "execute"). The LSC language also encompasses conditions and control structures, and it allows defining requirements in terms of the negation of charts. Figure 1 illustrates the LSC for the scenario "When the user clicks the button, the display color must change to red.". The respective *system model* (SM) is a code-base hierarchy containing the classes USER, BUTTON, DISPLAY, the method BUTTON.CLICK() and the property DISPLAY.COLOR.

3 Formal Settings

In the text-to-code generation task, we aim to implement a prediction function $f : \mathcal{D} \rightarrow \mathcal{M}$, such that $D \in \mathcal{D}$ is a piece of discourse consisting of an ordered set of requirements $D = d_1, d_2...d_n$, and $f(D) = M \in \mathcal{M}$ is a code-base hierarchy that grounds the semantic interpretation of D; we denote this by $M \triangleright \mathbf{sem}(d_1, ..., d_n)$. We now define the objects D, M, and describe how to construct the semantic interpretation function ($\mathbf{sem}(.)$). We then spell out the notion of grounding ($\triangleright$).

Surface Structures: Let Σ be a finite lexicon and let $\mathcal{L}_{req} \subseteq \Sigma^*$ be a language for specifying requirements. We assume the sentences in $\mathcal{L}_{req}$ have been generated by a context-free grammar $G = \langle \mathcal{N}, \Sigma, S \in \mathcal{N}, \mathcal{R} \rangle$, where $\mathcal{N}$ is a set of non-terminals, Σ is the aforementioned lexicon, $S \in \mathcal{N}$ is the start symbol and $\mathcal{R}$ is a set of context-free rules $\{A \rightarrow \alpha | A \in \mathcal{N}, \alpha \in (\mathcal{N} \cup \Sigma)^*\}$. For each utterance $u \in \mathcal{L}_{req}$, we can find a sequential application of rules that generates it: $u = r_1 \circ ... \circ r_k$; $\forall i : r_i \in \mathcal{R}$. We call such a sequence a *derivation* of u. These derivations may be graphically depicted as parse trees, where the utterance u defines the sequence of tree terminals in the leaves.

We define $\mathcal{T}_{req}$ to be the set of trees strongly generated by G, and utilize an auxiliary yield function $yield : \mathcal{T}_{req} \rightarrow \mathcal{L}_{req}$ returning the leaves of the given tree $t \in .\mathcal{L}_{req}$. Different parse-trees can generate the same utterance, so the task of analyzing the structure of an utterance $u \in \mathcal{L}_{req}$ is modeled via a function $syn : \mathcal{L}_{req} \rightarrow \mathcal{T}_{req}$ that returns the correct, human-perceived, parse of u.

Semantic Structures: Our target semantic representation of a requirement $d \in \mathcal{L}_{req}$ is a diagrammatic structure called a *live sequence chart* (LSC). The LSC formal definition we provide here is based on the appendix of Harel and Marelly (2003), but rephrased in set-theoretic, event-based, terms. We defined this alternative formalization in order to make LSCs compatible with Neo-Davidsonian, event-based, semantic theories. As a result, this form of LSC formalization is well-suited for representing the semantics of natural language utterances.

Let us assume that L is a dictionary of entities (lifelines), A is a dictionary of actions, P is a dictionary of attribute names and V a dictionary of attribute values. The set of simple events in the LSC formal system is defined as follows:

$$E_{active} \subset L \times A \times L \times (L \times P \times V)^*$$

$$\times \{hot, cold\} \times \{executed, monitored\}$$

where $e = \langle l_1, a, l_2, \{l_i : p_i : v_i\}_{i=3}^k, temp, exe \rangle$ and $l_i \in L, a \in A, p_i \in P, temp \in \{hot, cold\}$, $exe \in \{executed, monitored\}$. The event e is called a *message* in which an action a is carried over from a sender l_1 to a receiver l_2.[3] The set $\{l_i : p_i : v_i\}_{i=3}^k$ depicts a set of attribute:value pairs provided as arguments to action a. The temperature $temp$ marks the modality of the action (may, must), and the status exe distinguishes actions to be taken from actions to be waited for.

An event e can also refer to a state, where a logical expression is being evaluated over a set of property:value pairs. We call such an event a *condition*, and specify the set of possible conditions as follows:

$$E_{cond} \subset Exp \times (L \times P \times V)^*$$

$$\times \{hot, cold\} \times \{executed, monitored\}$$

[3] The LSC language also distinguishes static lifelines from dynamically-bound lifelines. For brevity, we omit this from the formal description of events, and simply assert that it may be listed as one of the properties of the relevant lifeline.

Specifically, $e = \langle exp, \{l : p : v\}_{i=0}^{k}, temp, exe\rangle$ is a condition to be evaluated, where $l_i \in L, p_i \in P, v_i \in V, temp \in \{hot, cold\}$ and $exe \in \{executed, monitored\}$ are as specified above. The condition $exp \in Exp$ is a first-order logic formula using the usual operators $(\vee, \wedge, \rightarrow, \neg, \exists, \forall)$. The set $\{l : p : v\}_{i=0}^{k}$ depicts a (possibly empty) set of attribute:value pairs that participates as predicates in exp. Executing a condition, that is, evaluating the logical expression specified by exp, also has a modality (may/must) and an execution status (performed/waited for).

The LSC language further allows us to define more complex events by combining partially ordered sets of events with control structures.

$$E_{complex} \subset N \times E_{cond} \times$$

$$\{\langle E_c, <\rangle | \langle E_c, <\rangle \textit{ is a poset }\}$$

N is a set of non-negative integers, E_{cond} is a set of conditions as described above, and each element $\langle E_c, <\rangle$ is a partially ordered set of events. This structure allows us to derive three kinds of control structures:

- $e = \langle \#, \emptyset, \langle E, <\rangle\rangle$ is a loop in which $\langle E, <\rangle$ is executed $\#$ times.
- $e = \langle 0, cond, \langle E, <\rangle\rangle$ is a conditioned execution. If $cond$ holds, $\langle E, <\rangle$ is executed.
- $e = \langle \#, \{cond\}_{i=1}^{\#}, \{\langle E_c, <\rangle\}_{i=1}^{\#}\rangle$ is a switch: in case i, if the condition i holds, $\langle E_c, <\rangle_i$ is executed.

Definition 1 (LSC) An LSC $c = \langle E, <\rangle$ is a partially ordered set of events such that

$$\forall e \in E : e \in E_{active} \vee e \in E_{cond} \vee e \in E_{complex}$$

Grounded Semantics: The information represented in the LSC provides the recipe for a rigorous construction of the code-base that will implement the program. This code-base is said to *ground* the semantic representation. For example, if our target programming language is an Object-Oriented programming language such as Java, then the code-base will include the objects, the methods and the properties that are minimally required for executing the scenario that is represented by the LSC. We refer to this code-base as a *system model* (henceforth, SM), and define it as follows.

Definition 2: (SM) Let L_m be a set of implemented objects, A_m a set of implemented methods, P_m a set of arguments and V_m argument values. We further define the auxiliary functions $methods : A_m \rightarrow L_m$, $props : P_m \rightarrow L_m$ and $values : V_m \rightarrow L_m \times P_m$, for identifying the entity $l \in L_m$ that implements the method $a \in A_m$, the entity $l \in L_m$ that contains the property $p \in P_m$, and the entity property $\langle l, p\rangle \in L_m \times P_m$ that assumes that value $v \in V_m$, respectively. A *system model* (SM) is a tuple m representing the implemented architecture.

$$m = \langle L_m, A_m, P_m, V_m, methods, props, values\rangle$$

Analogously to *interpretation* functions in logic and natural language semantics, we assume here an *implementation* function, denoted $[[.]]$, which maps each formal entity in the LSC semantic representation to its instantiation in the code-base. Using this function we define a notion of grounding that captures the fact that a certain code-base permits the execution of a given LSC c.

Definition 3(a): (Grounding) Let $\mathcal{M}$ be the set of system models and let $\mathcal{C}$ be the set of LSC charts. We say that m *grounds* $c = \langle E, <\rangle$, and write $m \triangleright c$, if $\forall e \in E : m \triangleright e$, where:

- if $e \in E_{active}$ then
  $m \triangleright e \Leftrightarrow$
  $[[l_1]], [[l_2]] \in L$ &
  $[[a]] \in methods([[l_2]])$ &
  $\forall i : \langle l : p : v\rangle_i \Rightarrow [[l]] \in L_m \& [[p]] \in props[[l]] \& v \in values([[l]], [[p]])$
- if $e \in E_{cond}$ then
  $m \triangleright e \Leftrightarrow$
  $\forall i : \langle l : p : v\rangle_i \Rightarrow [[l]] \in L_m \& [[p]] \in props[[l]] \& v \in values([[l]], [[p]])$
- if $e = \langle \#, e_s, \langle E_c, <\rangle \in E_{complex}$ then
  $m \triangleright e \Leftrightarrow m \triangleright e_s$ & $\forall e' \in E_c : m \triangleright e'$

We have thus far defined how the semantics of a single LSC can be grounded in a single SM. In the real world, however, a requirements document typically contains multiple different requirements, but it is interpreted as a complete whole. The desired SM is then one that represents a single domain shared by all the specified requirements. Let us then assume a document $\mathbf{d} = d_1, ..., d_n$ containing n requirements, where $\forall i : d_i \in \mathcal{L}_{req}$, and

let $\sqcup$ be a unification operation that returns the formal unification of two SMs if such exists, and an empty SM otherwise. We define a discourse interpretation function $\mathbf{sem}(\mathbf{d})$ that returns a single SM for the entire document, where different mentions across sentences may share the same reference. The discourse interpretation function $\mathbf{sem}$ can be as simple as unifying all individual SMs for d_i, and asserting that all elements that have the same name in different SMs refer to a single element in the overall SM. Or, it can be as complex as taking into account synonyms ("*clicks* the button" and "*presses* the button"), anaphora ("when the user clicks the *button*, *it* changes colour"), binding ("when the user clicks *any* button, *this* button is highlighted"), and so on. We can now define the grounding of an entire requirements document.

Definition 3(b): (Grounding) Let $\mathbf{d} = d_1...d_n$ be a requirements document and let $\mathbf{m} = m_1...m_n$ be a sequence of system models. $M = \langle \mathbf{m}, \sqcup \rangle$ is a sequence of models and a unification operation, and $M \triangleright \mathbf{sem}(\mathbf{d})$ if and only if $\forall i : m_i \triangleright sem(d_i)$ and $((m_1 \sqcup m_2).... \sqcup m_n) \triangleright \mathbf{sem}(d_1, ..., d_n)$.

In this work we assume that $\mathbf{sem}(\mathbf{d})$ is a simple discourse interpretation function, where entities, methods, properties, etc. that are referred to using the same name in different local SMs refer to a single element in the overall code-base. This simple assumption already carries a substantial amount of disambiguating information concerning individual requirements. For example, assume that we have seen a "click" method over a "button" object in sentence i. This may help us disambiguate future attachment ambiguity, favoring structures where a "button" is attached to "click" over other attachment alternatives. Our goal is then to model discourse-level context for supporting the accurate semantic analysis of individual requirements.

4 Probabilistic Modeling

In this section we set out to explicitly model the requirement's *context*, formally captured as a document-level SM, in order to support the accurate disambiguation of the requirements' *content*. We first specify our probabilistic *content* model, a sentence-level model which is based on a probabilistic grammar augmented with compositional semantic rules. We then specify our probabilistic *context* model, a document-level sequence model that takes into account the content as well as the relation between SMs at different time points.

4.1 Sentence-Based Modeling

The task of our sentence-based model is to learn a function that maps each requirement sentence to its correct LSC diagram and SM snapshot. In a nutshell, we do this via a (partially lexicalized) probabilistic context-free grammar augmented with a semantic interpretation function.

More formally, given a discourse $D = d_1...d_n$ we think of each d_i as having been generated by a *probabilistic context-free grammar* (PCFG) G. The syntactic analysis of d_i may be ambiguous, so we first implement a syntactic analysis function $syn : \mathcal{L}_{req} \rightarrow \mathcal{T}_{req}$ using a probabilistic model that selects the most likely syntax tree t of each d individually. We can simplify $syn(d)$, with d constant with respect to the maximization:

$$\begin{aligned} syn(d) &= \mathit{argmax}_{t \in \mathcal{T}_{req}} P(t|d) \\ &= \mathit{argmax}_{t \in \mathcal{T}_{req}} \frac{P(t,d)}{p(d)} \\ &= \mathit{argmax}_{t \in \mathcal{T}_{req}} P(t,d) \\ &= \mathit{argmax}_{t \in \{t | t \in \mathcal{T}_{req}, yield(t)=d\}} P(t) \end{aligned}$$

Because of the context-freeness assumption, it holds that $P(t) = \prod_{r \in der(t)} P(r)$, where $der(t)$ returns the rules that derive t. The resulting probability distribution $P : \mathcal{T}_{req} \rightarrow [0, 1]$ defines a probabilistic language model over all requirements $d \in \mathcal{L}_{req}$, i.e., $\sum_{d \in \mathcal{L}_{req}} \sum_{t \in \mathcal{T}_{req}, yield(t)=d} P(t) = 1$.

We assume a function $sem : \mathcal{T} \rightarrow \mathcal{C}$ mapping syntactic parse trees to semantic constructs in the LSC language. Syntactic parse trees are complex entities, assigning structures to the flat sequences of words. The principle of compositionality asserts that the meaning of a complex syntactic entity is a function of the meaning of its parts and their mode of combination. Here, the semantics of a tree $t \in \mathcal{T}_{req}$ is derived compositionally from the interpretation of the rules in the grammar G. We overload the sem notation to define $sem : \mathcal{R} \rightarrow \mathcal{C}$ as a function assigning rules to LSC constructs (events or parts of events),[4] with $\hat{\circ}$ merging the resulting sets of events. Our sentence-based compositional semantics is summarized as:

$$sem(u) = sem(syn(u)) = sem(r_1 \circ ... \circ r_n) =$$

$$sem(r_1)\hat{\circ}...\hat{\circ}sem(r_n) = c_1\hat{\circ}...\hat{\circ}c_n = c$$

[4]Here, it suffices to say that sem maps edges in the syntax tree to functions in the API of an existing LSC editor. For example: $sem(NP \rightarrow DET\ NN) = fCreateObject(DET.sem, NN.sem)$. We specify the function sem in the supplementary materials. The code of sem is available as part of *PlayGo* (`www.playgo.co`).

For a single chart c, one can easily construct an implementation for every entity, action and property in the chart. Then, by design, we get an SM m such that $m \triangleright c$. To construct the SM of the entire discourse in the sentence-based model we simply return $f(d_1, ..., d_n) = \sqcup_{i=1}^{n} m_i$ where $\forall i : m_i \triangleright sem(syn(d_i))$ and $\sqcup$ unifies different mentions of the same string to a single element.

4.2 Discourse-Based Modeling

We assume a given document $D \in \mathcal{D}$ and aim to find the most probable system model $M \in \mathcal{M}$ that satisfies the requirements. We assume that M reflects a single domain that the stakeholders have in mind, and we are provided with an ambiguous natural language evidence, an elicited discourse D, in which they convey it. We instantiate this view as a *noisy channel* model (Shannon, 1948), which provides the foundation for many NLP applications, such as speech recognition (Bahl et al., 1983) and machine translation (Brown et al., 1993).

According to the noisy channel model, when a signal is received it does not uniquely identify the message being sent. A probabilistic model is then used to decode the original message. In our case, the signal is the discourse and the message is the overall system model. In formal terms, we want to find a model M that maximises the following:

$$f(D) = argmax_{M \in \mathcal{M}} P(M|D)$$

We can simplify further, using Bayes law, where D is constant with respect to the maximisation.

$$\begin{aligned} f(D) &= argmax_{M \in \mathcal{M}} P(M|D) \\ &= argmax_{M \in \mathcal{M}} \frac{P(D|M) \times P(M)}{P(D)} \\ &= argmax_{M \in \mathcal{M}} P(D|M) \times P(M) \end{aligned}$$

We would thus like to estimate two types of probability distributions, $P(M)$ over the source and $P(D|M)$ over the channel.

Both M and D are structured objects with complex internal structure. In order to assign probabilities to objects involving such complex structures it is customary to break them down into simpler, more basic, events. We know that $D = d_1, d_2, ..., d_n$ is composed of n individual sentences, each representing a certain aspect of the model M. We assume a sequence of *snapshots* of M that correspond to the timestamps $1...n$, that is: $m_1, m_2, ..., m_n \in \mathcal{M}$ where $\forall i : m_i \triangleright sem(d_i)$. The complete SM is given by the union of the different snapshots reflected in different requirements, i.e., $M = \bigsqcup_i m_i$. We then rephrase:

$$\begin{aligned} P(M) &= P(m_1, ..., m_n) \\ P(D|M) &= P(d_1, ..., d_n | m_1, ..., m_n) \end{aligned}$$

These events may be seen as points in a high dimensional space. In actuality, they are too complex and would be too hard to estimate directly. We then define two independence assumptions. First, we assume that a system model snapshot at time i depends only on k previous snapshots (a stationary distribution). Secondly, we assume that each sentence i depends only on the SM snapshot at time i. We now get:

$$\begin{aligned} P(m_1...m_n) &\approx \textstyle\prod_i P(m_i|m_{i-1}...m_{i-k}) \\ P(d_1...d_n|m_1...m_n) &\approx \textstyle\prod_i P(d_i|m_i) \end{aligned}$$

Furthermore, assuming bi-gram transitions, our objective function is now represented as follows:

$$f(D) = argmax_{M \in \mathcal{M}} \prod_{i=1}^{n} P(m_i|m_{i-1}) P(d_i|m_i)$$

Note that m_0 may be empty if the system is implemented from scratch, and non-empty if the requirements assume an existing code-base, which makes $p(m_1|m_0)$ a non-trivial transition.

4.3 Training and Decoding

Our model is in essence a Hidden Markov Model in which states capture SM snapshots, state-transition probabilities model transitions between SM snapshots, and emission probabilities model the verbal description of each state. To implement this, we need to implement a decoding algorithm that searches through all possible state sequences, and a training algorithm that can automatically learn the values of the still rather complex parameters $P(m_i|m_{i-1}), P(d_i|m_i)$ from data.

$$f(D) = \underbrace{argmax_{M \in \mathcal{M}}}_{decoding} \prod_{i=1}^{n} \underbrace{P(m_i|m_{i-1}) P(d_i|m_i)}_{training}$$

Training: We assume a supervised training setting in which we are given a set of examples annotated by a human expert. For instance, these can be requirements an analyst has formulated and encoded using an LSC editor, while manually providing disambiguating information. We are provided with a set of pairs $\{D_i, M_i\}_{i=1}^{n}$ containing n documents, where each of the pairs in $i = 1..n$ is a

tuple set $\{d_{ij}, t_{ij}, c_{ij}, m_{ij}\}_{j=1}^{n_i}$. For all i, j, it holds that $t_{ij} = syn(d_{ij})$, $c_{ij} = sem(t_{ij})$, and $m_{ij} \triangleright sem(syn(d_{ij}))$. The union of the n_i SM snapshots yields the entire model $\sqcup_j m_{i_j} = M_i$, that satisfies the set of requirements $M_i \triangleright \mathbf{sem}(d_{i1}, ..., d_{in_i})$.

(i) Emission Parameters Our emission parameters $P(d_i|m_i)$ represent the probability of a verbal description of a requirement given an SM snapshot which grounds the semantics of the description. A single SM may result from different syntactic derivations. We calculate this probability by marginalizing over the syntactic trees that are grounded in the same SM snapshot.

$$\frac{P(d,m)}{P(m)} = \frac{\sum_{t \in \{t|yield(t)=d, m \triangleright sem(t)\}} P(t)}{\sum_{t \in \{t|t \in \mathcal{T}_{req}, m \triangleright sem(t)\}} P(t)}$$

The probability of $P(t)$ is estimated using a treebank PCFG (Charniak, 1996), based on all pairs $\langle d_{ij}, t_{ij} \rangle$ in the annotated corpus. We estimate rule probabilities using maximum-likelihood estimates, and use simple smoothing for unknown lexical items, using rare-words distributions.

(ii) Transition Parameters Our transition parameters $P(m_i|m_{i-1})$ represent the amount of overlap between the SM snapshots. We look at the current and the previous system model, and aim to estimate how likely the current SM is given the previous one. There are different assumptions that may underly this probability distribution, reflecting different principles of human communication.

We first define a generic estimator as follows:

$$\hat{P}(m_i|m_j) = \frac{gap(m_i, m_j)}{\sum_{m_j} gap(m_i, m_j)}$$

where $gap(.)$ quantifies the information sharing between SM snapshots. Regardless of the implementation of gap, it can be easily shown that $\hat{P}$ is a conditional probability distribution where $\hat{P} : \mathcal{M} \times \mathcal{M} \rightarrow [0, 1]$ and, for all $m_i, m_j, : \sum_{m_j} \hat{P}(m_i|m_j) = 1$. (For efficiency reasons, we consider $\mathcal{M}$ to be a restricted universe that is considered be the decoder, as specified shortly.)

We define different gap implementations, reflecting different assumptions about the discourse. Our first assumption here is that different SM snapshots refer to the same conceptual world, so there should be a large overlap between them. We call this the **max-overlap** assumption. A second assumption is that, in collaborative communication, a new requirement will only be stated if it provides new information, akin to Grice (1975). This is the **max-expansion** assumption. An additional assumption prefers "easy" transitions over "hard" ones, this is the **min-distance** assumption. The different gap calculations are listed in Table 1.

Transition:	$gap(m_{curr}, m_{prev})$
max-overlap	$\frac{\lvert set(m_{curr}) \cap set(m_{prev}) \rvert}{\lvert set(m_{curr}) \rvert}$
max-expansion	$1 - \frac{\lvert set(m_{curr}) \cap set(m_{prev}) \rvert}{\lvert set(m_{prev}) \cup set(m_{curr}) \rvert}$
min-distance	$1 - \frac{ted(m_{prev}, m_{curr})}{\lvert set(m_{prev}) \rvert + \lvert set(m_{curr}) \rvert}$

Table 1: Quantifying the gap between snapshots. $set(m_i)$ is a set of nodes marked by path to root.

Decoding An input document contains n requirements. Our decoding algorithm considers the N-best syntactic analyses for each requirement. At each time step $i = 1...n$ we assume N, states representing the semantics of the N best syntax trees, retrieved via a CKY chart parser. Thus, setting $N = 1$ is equal to a sentence-based model, in which for each sentence we simply select the most likely tree according to a probabilistic grammar, and construct a semantic representation for it.

For each document of length n, we assume that our entire universe of system models $\mathcal{M}$ is composed of $N \times n$ SM snapshots, reflecting the N most-likely analyses of n sentences, as provided by the probabilistic syntactic model. (As shall be seen shortly, even with this restricted[5] universe approximating $\mathcal{M}$, our discourse-based model provides substantial improvements over a sentence-based model).

Our discourse-based model is an HMM where each requirement is an observed signal, and each $i = 1..N$ is a state representing the SM that grounds the i_th best tree. Because of the Markov independence assumption our setup satisfies the *optimal subproblem* and *overlapping problem* properties, and we can use efficient *viterbi* decoding to exhaustively search through the different state sequences, and find the most probable sequence that has generated the sequence of requirements according to our discourse-based probabilistic model.

[5]This restriction is akin to pseudo-likelihood estimation, as in Arnold and Strauss (1991). In pseudo-likelihood estimation, instead of normalizing over the entire set of elements, one uses a subset that reflects only the possible outcomes. Here, instead of summing SM probabilities over all possible sentences in the language, we sum up the SM analyses of the sentences observed in the document only. This estimation could also be addressed via, e.g., sampling methods.

The overall complexity decoding a document with n sentences of which max length is l, using a grammar G of size $|G|$ and a fixed N, is given by:

$$O(n \times l^3 \times |G|^3 + l^2 \times N^2 \times n + n^3 \times N^2)$$

We can break this expression down as follows: (i) In $O(n \times l^3 \times |G|^3)$ we generate N best trees for each one of the n requirements, using a CKY chart (Younger, 1967). (ii) In $O(l^2 \times N^2 \times n)$ we create the universe $\mathcal{M}$ based on the N best trees of the n requirements, and calculate $N \times N$ transitions. (iii) In $O((N \times n)^2 \times n) = O(N^2 \times n^3)$ we decode the $n \times N$ grid using Viterbi (1967) decoding.

5 Experiments

Goal. We set out to evaluate the accuracy of a semantic parser for requirements documents, in the two modes of analysis presented above. Our evaluation methodology is as standardly assumed in machine learning and NLP: given a set of annotated examples — that is, given a set of requirements documents, where each requirement is annotated with its correct LSC representation and each document is associated with a complete SM — we partition this set into a *training set* and a *test set* that are disjoint. We train our statistical model on the examples in the training set and automatically analyze the requirements in the test set. We then compare the predicted semantic analyses of the test set with the human-annotated (henceforth, *gold*) semantic analyses of this test set, and empirically quantify our prediction accuracy.

Metrics. Our semantic LSC objects have the form of a tree (reflecting the sequence of nested events in our scenarios). Therefore, we can use standard tree evaluation metrics, such as ParseEval (Black et al., 1992), to evaluate the accuracy of the prediction. Overall, we define three metrics to evaluate the accuracy of the LSC trees:

> **POS:** the POS metric is the percentage of part-of-speech tags predicted correctly.
> **LSC-F1:** F1 is the harmonic means of the precision and recall of the predicted tree.
> **LSC-EM:** EM is 1 if the predicted tree is an exact match to the gold tree, and 0 otherwise.

In the case of SM trees, as opposed to the LSC trees, we cannot assume identity of the yield between the gold and parse trees for the same sentence,[6] so we cannot use ParseEval. Therefore, we implement a distance-based metrics in the spirit of Tsarfaty et al. (2012). Then, to evaluate the accuracy of the SM, we use two kinds of scores:

> **SM-TED:** TED is the normalized edit distance between the predicted and gold SM trees, subtracted from a unity.
> **SM-EM:** EM is 1 if the predicted SM is an exact match with the gold SM, 0 otherwise.

System	#Scenarios	avg sentence length
Phone	21	24.33
WristWatch	15	29.8
Chess	18	15.83
Baby Monitor	14	20
Total	68	22.395

Table 2: Seed Gold-Annotated Requirements

N=1	POS	LSC-F1	LSC-EM	SM-TED	SM-EM
Gen-Only	85.52	84.40	9.52	84.25	9.52
Gen+Seed	**91.59**	**88.05**	**14.29**	**85.17**	**14.29**

Table 3: Sentence-Based modeling: Accuracy results on the *Phone* development set.

Data. We have a small seed of correctly annotated requirements-specification case studies that describe simple reactive systems in the LSC language. Each document contains a sequence of requirements, each of which is annotated with the correct LSC diagram. The entire program is grounded in a java implementation. As training data, we use the case studies provided by Gordon and Harel (2009). Table 2 lists the case studies and basic statistics concerning these data.

As our annotated seed is quite small, it is hard to generalize from it to unseen examples. In particular, we are not guaranteed to have observed all possible structures that are theoretically permitted by the assumed grammar. To cope with this, we create a synthetic set of examples using the grammar of Gordon and Harel (2009) in generation mode, and randomly generate trees $t \in \mathcal{T}_{req}$.

The grammar we use to generate the synthetic examples clearly *over-generates*. That is to say, it creates many trees that do not have a sound interpretation. In fact, only 3000 our of 10000 generated examples turn out to have a sound semantic interpretation grounded in an SM. Nonetheless, these data allow us to smooth the syntactic distributions that are observed in the seed, and increase the coverage of the grammar learned from it.

[6] This is because the LSC trees are predicted bottom up and the SM trees are predicted top-down.

Results. Table 3 presents the results for parsing the *Phone* document, our development set, with the sentence-based model, varying the training data. We see that despite the small size of the seed, adding it to our set if synthetics examples substantially improves results over a model trained on synthetic examples only.

In our next experiment, we provide empirical upper-bounds and lower-bounds for the discourse-based model. Table 4 presents the results of the discourse-based model for $N > 1$ on the *Phone* example. *Gen-Only* presents the results of the discourse-based model with a PCFG learned from synthetic trees only, incorporating transitions obeying the **max-overlap** assumption. Already here, we see a mild improvement for $N > 1$ relative to the $N = 1$ results, indicating that even a weak signal such as the overlap between neighboring sentences already improves sentence disambiguation in context. We next present the results of an *Oracle* experiment, where every requirement is assigned the highest scoring tree in terms of LSC-F1 with respect to the gold tree, keeping the same transitions. Again we see that results improve with N, indicating that the syntactic model alone does not provide optimal disambiguation. These results provides an upper bound on the parser performance for each N. *Gen+Seed* presents results of the discourse-based model where the PCFG interpolates the seed set and the synthetic train set, with **max-overlap** transitions. Here, we see larger improvements over the synthetic-only PCFG. That is, modeling grammaticality of individual sentences improves the interpretation of the document.

Next we compare the performance for different implementations of the $gap(m_i, m_j)$ function. We estimate probability distributions that reflect each of the assumptions we discussed, and add an additional method called **hybrid**, in which we interpolate the **max-expansion** and **max-overlap** estimates (equal weights). In Table 5, the trend from the previous experiment persists. Notably, the **hybrid** model provides a larger error reduction than its components used separately, indicating that in order to capture discourse context we may need to balance possibly conflicting factors. In **no emissions** we rely solely on the probability of state transitions, and again increasing N leads to improvement. This result confirms that *context* is indispensable for sentence interpretation — even when probabilities for the sentence's semantics (*content*) are entirely absent.

System	N=2	4	8	16	32	64	128
Gen-Only							
POS	85.52	86.30	87.67	88.45	**88.85**	88.85	88.85
LSC-F1	84.40	85.35	86.31	87.51	88.81	89.30	**89.51**
LSC-EM	9.52	9.52	**14.29**	14.29	14.29	14.29	14.29
SM-TED	84.25	85.94	89.14	91.90	92.81	**93.31**	92.70
SM-EM	9.52	19.05	33.33	33.33	33.33	**38.10**	33.33
Gen+Seed							
POS	91.78	92.95	93.54	93.35	94.32	**94.52**	93.93
LSC-F1	88.11	90.18	91.00	90.99	91.81	**92.09**	91.73
LSC-EM	19.05	38.10	**42.86**	42.86	42.86	42.86	42.86
SM-TED	85.49	90.78	93.59	93.02	94.81	**95.69**	93.76
SM-EM	19.05	38.10	**52.38**	52.38	52.38	52.38	52.38
Oracle							
POS	91.98	93.54	94.91	95.30	96.09	96.67	**96.87**
LSC-F1	88.73	91.33	93.19	94.39	95.11	95.91	**96.70**
LSC-EM	23.81	42.86	61.90	61.90	66.67	76.19	**76.19**
SM-TED	86.54	91.28	94.28	94.88	96.24	97.51	**98.80**
SM-EM	23.81	42.86	66.67	71.43	**76.19**	76.19	76.19

Table 4: Discourse-Based Modeling: Accuracy results on the *Phone* dev set. The Oracle selects the highest scoring LSC tree among the N-candidates, providing an upper bound on accuracy. Gen-Only selects the most probable tree, relying on synthetic examples only, providing a lower bound.

We finally perform a cross-fold experiment in which we leave one document out as a test set and take the rest as our seed. The results are provided in Table 6. The discourse-based model outperforms the sentence-based model $N = 1$ in all cases. Moreover, the drop in $N = 128$ for *Phone* seems incidental to this set, and the other cases level off beforehand. Despite our small seed, the persistent improvement on all metrics is consistent with our hypothesis that modeling the interpretation process within the discourse has substantial benefits for automatic understanding of the text.

6 Applications and Discussion

The statistical models we present here are applied in the context of *PlayGo*,[7] a comprehensive tool for behavioral, scenario-based, programming. PlayGo now provides two modes of *playing-in* natural language requirements: interactive play-in, where a user manually disambiguates the analyses of the requirements (Gordon and Harel, 2009), and statistical play-in, where disambiguation decisions are taken using our discourse-based model.

The fragment of English we use is very expressive. It covers not only entities and predicates, but also temporal and aspectual information, modalities, and program flow. Beyond that, we assume an open-ended lexicon. Overall, we are not

[7] `www.playgo.co`.

Transitions	N=2	4	8	16	32	64	128
Min Dist							
POS	91.98	92.76	93.54	93.35	94.32	**94.52**	93.93
LSC-F1	88.39	89.77	91.00	90.99	91.81	**92.09**	91.73
LSC-EM	23.81	42.86	**47.62**	47.62	47.62	47.62	47.62
SM-TED	86.54	91.71	94.38	93.81	95.57	**96.43**	94.53
SM-EM	23.81	42.86	**57.14**	57.14	57.14	57.14	57.14
Max Overlap							
POS	91.78	92.95	93.54	93.35	94.32	**94.52**	93.93
LSC-F1	88.11	90.18	91.00	90.99	91.81	**92.09**	91.73
LSC-EM	19.05	38.10	**42.86**	42.86	42.86	42.86	42.86
SM-TED	85.49	90.78	93.59	93.02	94.81	**95.69**	93.76
SM-EM	19.05	38.10	**52.38**	52.38	52.38	52.38	52.38
Max Expand							
POS	91.98	92.76	93.74	93.54	94.32	**94.52**	93.93
LSC-F1	88.39	89.71	91.00	90.99	91.68	**91.96**	91.60
LSC-EM	23.81	42.86	**47.62**	47.62	47.62	47.62	47.62
SM-TED	86.54	91.93	93.75	93.18	94.79	**95.66**	93.75
SM-EM	23.81	42.86	**57.14**	57.14	57.14	57.14	57.14
Hybrid							
POS	91.78	92.95	93.93	93.74	94.72	**94.91**	94.32
LSC-F1	88.11	90.18	91.34	91.33	92.15	**92.42**	92.07
LSC-EM	19.05	38.10	**47.62**	47.62	47.62	47.62	47.62
SM-TED	85.49	90.78	93.66	93.09	94.87	**95.75**	93.83
SM-EM	19.05	38.10	**57.14**	57.14	57.14	57.14	57.14
No Emissions							
POS	91.78	91.98	92.37	92.37	92.17	92.76	**93.15**
LSC-F1	88.11	88.79	89.12	89.12	89.39	89.67	**89.89**
LSC-EM	19.05	19.05	**23.81**	23.81	23.81	23.81	23.81
SM-TED	85.49	85.74	85.82	85.82	85.87	86.85	**86.92**
SM-EM	19.05	19.05	**23.81**	23.81	23.81	23.81	23.81

Table 5: Discourse-Based modeling: Experiments on the *Phone* development set. Estimation procedure for transition probabilities. All experiments use the Gen+Seed emission probablities.

only translating English sentences into executable LSCs — we provide a fully generative model for translating a complete document (text) into a complete system model (code).

This text-to-code problem may be thought of as a machine translation (MT) problem, where one aims to translate sentences in English to the formal language of LSCs. However, standard statistical MT techniques rely on the assumption that textual requirements and code are aligned at a sentence level. Creating a formal model that aligns text and code on a sentence-by-sentence basis is precisely our technical contribution in Section 3.

To our knowledge, modeling syntax and discourse processing via a fully joint generative model, where a discourse level HMM is interleaved with PCFG sentence-based emissions, is novel. By plugging in different models for $p(d|m)$, different languages may be parsed. This method may further be utilized for relating content and context in other tasks: parsing and document-level NER, parsing and document-level IE, etc. To do so, one only needs to redefine the PCFG (emissions) and state-overlap (transition) parameters, as appropriate for their data.[8]

[8]Our code, annotated data, four case studies, and the LSC visual editor are available via `http://wiki.weizmann.ac.il/playgo/index.php/Download_PlayGo`.

Data Set	N=1	32	64	128
Baby Monitor				
POS	94.29	96.07	96.07	96.07
LSC-F1	91.50	94.96	94.96	94.96
LSC-EM	14.29	21.43	21.43	21.43
SM-TED	88.63	91.11	91.11	91.11
SM-EM	28.57	50.00	50.00	50.00
Chess				
POS	92.63	93.68	93.68	93.68
LSC-F1	95.79	96.16	96.16	96.16
LSC-EM	5.56	11.11	11.11	11.11
SM-TED	94.90	97.10	97.10	97.10
SM-EM	61.11	66.67	66.67	66.67
Phone				
POS	91.59	94.72	94.91	94.32
LSC-F1	88.05	92.15	92.42	92.07
LSC-EM	14.29	47.62	47.62	47.62
SM-TED	85.17	94.87	95.75	93.83
SM-EM	14.29	57.14	57.14	57.14
WristWatch				
POS	34.23	34.45	34.45	34.45
LSC-F1	50.06	51.05	51.05	51.05
LSC-EM	26.67	26.67	26.67	26.67
SM-TED	71.15	72.73	72.73	72.73
SM-EM	26.67	33.33	33.33	33.33

Table 6: Cross-Fold Validation for N=1..128. Seed+Generated emissions, **Hybrid** transitions.

7 Conclusion

The requirements understanding task presents an exciting challenge for CL/NLP. We ought to automatically discover the entities in the discourse, the actions they take, conditions, temporal constraints, and execution modalities. Furthermore, it requires us to extract a single ontology that satisfies all individual requirements. The contributions of this paper are three-fold: we formalize the text-to-code prediction task, propose a semantic representation with well-defined grounding, and empirically evaluate models for this prediction. We show consistent improvement of discourse-based over sentence-based models, in all case studies. In the future, we intend to extend this model for interpreting requirements in un-restricted, or less-restricted, English, endowed with a more sophisticated discourse interpretation function.

Acknowledgements

We thank Shahar Maoz, Rami Marelly, Yoav Goldberg and three anonymous reviewers for their insightful comments on an earlier draft. This research was supported by an Advanced Research Grant to D. Harel from the European Research Council (ERC) under the European Community's Seventh Framework Programme (FP7/2007-2013), and by a grant to D. Harel from the Israel Science Foundation (ISF).

References

B. C. Arnold and D. Strauss. 1991. Pseudolikelihood Estimation: Some Examples. *Sankhyā: The Indian Journal of Statistics, Series B (1960-2002)*, 53(2).

Y. Artzi and L. Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL*, 1:49–62.

L. R. Bahl, F. Jelinek, and R. L. Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 5(2):179–190.

E. Black, J. D. Lafferty, and S. Roukos. 1992. Development and evaluation of a broad-coverage probabilistic grammar of English-language computer manuals. In *Proceedings of ACL*, pages 185–192.

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June.

B. Bryant and B.-S. Lee. 2002. Two-level grammar as an object-oriented requirements specification language. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9 - Volume 9*, HICSS '02, pages 280–, Washington, DC, USA. IEEE Computer Society.

E. Charniak. 1996. Tree-bank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1031–1036.

W. Damm and D. Harel. 2001. LSCs: Breathing life into message sequence charts. *Form. Methods Syst. Des.*, 19(1):45–80, July.

N. Eitan, M. Gordon, D. Harel, A. Marron, and G. Weiss. 2011. On visualization and comprehension of scenario-based programs. In *Proceedings of the 2011 IEEE 19th International Conference on Program Comprehension*, ICPC '11, pages 189–192, Washington, DC, USA. IEEE Computer Society.

N. E. Fuchs and R. Schwitter. 1995. Attempto: Controlled natural language for requirements specifications. In Markus P. J. Fromherz, Marc Kirschenbaum, and Anthony J. Kusalik, editors, *LPE*.

M. Gordon and D. Harel. 2009. Generating executable scenarios from natural language. In *Proceedings of the 10th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing '09, pages 456–467, Berlin, Heidelberg. Springer-Verlag.

H. P. Grice. 1975. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics: Vol. 3: Speech Acts*, pages 41–58. Academic Press, San Diego, CA.

D. Harel and M. Gordon. 2009. On teaching visual formalisms. *IEEE Softw.*, 26(3):87–95, May.

D. Harel and S. Maoz. 2006. Assert and negate revisited: Modal semantics for UML sequence diagrams. In *Proceedings of the 2006 International Workshop on Scenarios and State Machines: Models, Algorithms, and Tools*, SCESM '06, pages 13–20, New York, NY, USA. ACM.

D. Harel and R. Marelly. 2003. *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

D. Harel, H. Kugler, R. Marelly, and A. Pnueli. 2002. Smart play-out of behavioral requirements. In *Proceedings of the 4th International Conference on Formal Methods in Computer-Aided Design*, FMCAD '02, pages 378–398, London, UK. Springer-Verlag.

D. Harel, A. Marron, and G. Weiss. 2012. Behavioral programming. *Commun. ACM*, 55(7):90–100, July.

D. Harel, A. Kantor, G. Katz, A. Marron, L. Mizrahi, and G. Weiss. 2013. On composing and proving the correctness of reactive behavior. In *Embedded Software (EMSOFT), 2013 Proceedings of the International Conference on*, pages 1–10, Sept.

D. Harel. 2001. From play-in scenarios to code: An achievable dream. *Computer*, 34(1):53–60, January.

H. Kugler, D. Harel, A. Pnueli, Y. Lu, and Y. Bontemps. 2005. Temporal logic for scenario-based specifications. In *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'05, pages 445–460, Berlin, Heidelberg. Springer-Verlag.

T. Kuhn. 2014. A survey and classification of controlled natural languages. *Computational Linguistics*, 40(1):121–170.

T. Lei, F. Long, R. Barzilay, and M. C. Rinard. 2013. From natural language specifications to program input parsers. In *ACL (1)*, pages 1294–1303.

P. Liang and C. Potts. 2014. Bringing machine learning and compositional semantics together. *Annual Reviews of Linguistics (submitted)*, 0.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599.

T. Parsons. 1990. *Events in the Semantics of English: A study in subatomic semantics*. MIT Press, Cambridge, MA.

C. Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, July, October.

R. Tsarfaty, J. Nivre, and E. Andersson. 2012. Cross-framework evaluation for statistical parsing. In W. Daelemans, M. Lapata, and L. Màrquez, editors, *Proceedings of EACL*, pages 44–54. The Association for Computer Linguistics.

A. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theor.*

D. H. Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and Control*, 10(2):189–208.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, pages 658–666. AUAI Press.

Semantic Parsing with Relaxed Hybrid Trees

Wei Lu
Information Systems Technology and Design
Singapore University of Technology and Design
luwei@sutd.edu.sg

Abstract

We propose a novel model for parsing natural language sentences into their formal semantic representations. The model is able to perform integrated lexicon acquisition and semantic parsing, mapping each atomic element in a complete semantic representation to a contiguous word sequence in the input sentence in a recursive manner, where certain overlappings amongst such word sequences are allowed. It defines distributions over the novel *relaxed hybrid tree* structures which jointly represent both sentences and semantics. Such structures allow tractable dynamic programming algorithms to be developed for efficient learning and decoding. Trained under a discriminative setting, our model is able to incorporate a rich set of features where certain unbounded long-distance dependencies can be captured in a principled manner. We demonstrate through experiments that by exploiting a large collection of simple features, our model is shown to be competitive to previous works and achieves state-of-the-art performance on standard benchmark data across four different languages. The system and code can be downloaded from http://statnlp.org/research/sp/.

1 Introduction

Semantic parsing, the task of transforming natural language sentences into formal representations of their underlying semantics, is one of the classic goals for natural language processing and artificial intelligence. This area of research recently has received a significant amount of attention. Various models have been proposed over the past few years (Zettlemoyer and Collins, 2005; Kate and Mooney, 2006; Wong and Mooney, 2006; Lu et al., 2008; Jones et al., 2012).

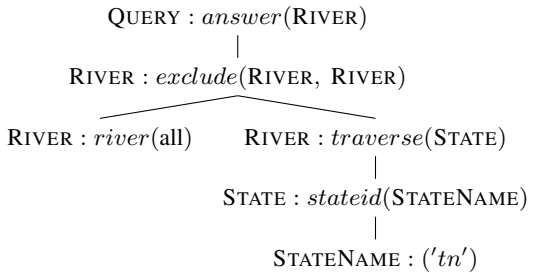


Figure 1: An example tree-structured semantic representation (above) and its corresponding natural language sentence.

Following previous research efforts, we perform semantic parsing under a setting where the semantics for complete sentences are provided as training data, but detailed word-level semantic information is not explicitly given during the training phase. As one example, consider the following natural language sentence paired with its corresponding semantic representation:

What rivers do not run through Tennessee ?
$answer(exclude(river(all), traverse(stateid('tn'))))$

The training data consists of a set of sentences paired with semantic representations. Our goal is to learn from such pairs a model, which can be effectively used for parsing novel sentences into their semantic representations.

Certain assumptions about the semantics are typically made. One common assumption is that the semantics can be represented as certain recursive structures such as trees, which consist of atomic semantic units as tree nodes. For example, the above semantics can be converted into an equivalent tree structure as illustrated in Figure 1. We will provide more details about such tree structured semantic representations in Section 2.1.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1308–1318,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Currently, most state-of-the-art approaches that deal with such tree structured semantic representations either cast the semantic parsing problem as a statistical string-to-string transformation problem (Wong and Mooney, 2006), which ignores the potentially useful structural information of the tree, or employ latent-variable models to capture the correspondences between words and tree nodes using a generative approach (Lu et al., 2008; Jones et al., 2012). While generative models can be used to flexibly model the correspondences between individual words and semantic nodes of the tree, such an approach is limited to modeling local dependencies and is unable to flexibly incorporate a large set of potentially useful features.

In this work, we propose a novel model for parsing natural language into tree structured semantic representations. Specifically, we propose a novel *relaxed hybrid tree* representation which jointly encodes both natural language sentences and semantics; such representations can be effectively learned with a latent-variable discriminative model where long-distance dependencies can be captured. We present dynamic programming algorithms for efficient learning and decoding. With a large collection of simple features, our model reports state-of-the-art results on benchmark data annotated with four different languages.

Furthermore, although we focus our discussions on semantic parsing in this work, our proposed model is a general. Essentially our model is a discriminative string-to-tree model which recursively maps overlapping contiguous word sequences to tree nodes at different levels, where efficient dynamic programming algorithms can be used. Such a model may find applications in other areas of natural language processing, such as statistical machine translation and information extraction.

2 Background

2.1 Semantics

Various semantic formalisms have been considered for semantic parsing. Examples include the tree-structured semantic representations (Wong and Mooney, 2006), the lambda calculus expressions (Zettlemoyer and Collins, 2005; Wong and Mooney, 2007), and dependency-based compositional semantic representations (Liang et al., 2013). In this work, we specifically focus on the tree-structured representations for semantics.

Each semantic representation consists of semantic units as its tree nodes, where each semantic unit is of the following form:

$$m_a \equiv \tau_a : p_\alpha(\tau_b*) \quad (1)$$

Here m_a is used to denote a complete semantic unit, which consists of its semantic type τ_a, its function symbol p_α, as well as an argument list τ_b* (we assume there are at most two arguments for each semantic unit). In other words, each semantic unit can be regarded as a function which takes in other semantics of specific types as arguments, and returns new semantics of a particular type. For example, in Figure 1, the semantic unit at the root has a type QUERY, a function name $answer$, and a single argument type RIVER.

2.2 Joint Representations

Semantic parsing models transform sentences into their corresponding semantics. It is therefore essential to make proper assumptions about joint representations for language and semantics that capture how individual words and atomic semantic units connect to each other. Typically, different existing models employ different assumptions for establishing such connections, leading to very different definitions of joint representations. We survey in this section various representations proposed by previous works.

The WASP semantic parser (Wong and Mooney, 2006) essentially casts the semantic parsing problem as a string-to-string transformation problem by employing a statistical phrase-based machine translation approach with synchronous grammars (Chiang, 2007). Therefore, one can think of the joint representation for both language and semantics as a synchronous derivation tree consisting of those derivation steps for transforming sentences into target semantic representation strings. While this joint representation is flexible, allowing blocks of semantic structures to map to word sequences, it does not fully exploit the structural information (tree) as conveyed by the semantics.

The KRISP semantic parser (Kate and Mooney, 2006) makes use of Support Vector Machines with string kernels (Lodhi et al., 2002) to recursively map contiguous word sequences into semantic units to construct a tree structure. Our *relaxed hybrid tree* structures also allow input word sequences to map to semantic units in a recursive manner. One key distinction, as we will see, is that our structure distinguishes words which are imme-

diately associated with a particular semantic unit, from words which are remotely associated.

The SCISSOR model (Ge and Mooney, 2005) performs integrated semantic and syntactic parsing. The model parses natural language sentences into semantically augmented parse trees whose nodes consist of both semantic and syntactic labels and then builds semantic representations based on such augmented trees. Such a joint representation conveys more information, but requires language-specific syntactic analysis.

The *hybrid tree* model (Lu et al., 2008) is based on the assumption that there exists an underlying generative process which jointly produces both the sentence and the semantic tree in a top-down recursive manner. The generative process results in a hybrid tree structure which consists of words as leaves and semantic units as nodes. An example hybrid tree structure is shown in Figure 2 (a). Such a representation allows each semantic unit to map to a possibly *discontiguous* sequence of words. The model was shown to be effective empirically, but it implicitly assumes that both the sentence and semantics exhibit certain degree of structural similarity that allows the hybrid tree structures to be constructed.

UBL (Kwiatkowski et al., 2010) is a semantic parser based on restricted higher-order unification with CCG (Steedman, 1996). The model can be used to handle both tree structured semantic representations and lambda calculus expressions, and assumes there exist CCG derivations as joint representations in which each semantic unit is associated with a contiguous word sequence where overlappings amongst word sequences are not allowed.

Jones et al. (2012) recently proposed a framework that performs semantic parsing with tree transducers. The model learns representations that are similar to the hybrid tree structures using a generative process under a Bayesian setting. Thus, their representations also potentially present similar issues as the ones mentioned above.

Besides these supervised approaches, recently there are also several works that take alternative learning approaches to (mostly task-dependent) semantic parsing. Poon and Domingos (2009) proposed a model for unsupervised semantic parsing that transforms dependency trees into semantic representations using Markov logic (Richardson and Domingos, 2006). Clarke et al. (2010) proposed a model that learns a semantic parser for answering questions without relying on semantic annotations. Goldwasser et al. (2011) took an unsupervised approach for semantic parsing based on self-training driven by confidence estimation. Liang et al. (2013) proposed a model for learning the dependency-based compositional semantics (DCS) which can be used for optimizing the end-task performance. Artzi and Zettlemoyer (2013) proposed a model for mapping instructions to actions with weak supervision.

Symbol	Description
$\mathbf{n}$	A complete natural language sentence
$\mathbf{m}$	A complete semantic representation
$\mathbf{h}$	A complete latent joint representation (or, a *relaxed hybrid tree* for our work)
$\mathcal{H}(\mathbf{n}, \mathbf{m})$	A complete set of latent joint representations that contain the $(\mathbf{n}, \mathbf{m})$ pair exactly
n, n_a	A contiguous sequence of words
w, w_k	A natural language word
m, m_a	A semantic unit
h, h_a	A node in the relaxed hybrid tree
Φ	The feature vector
ϕ_k	The k-th feature
Λ	The weight vector (model parameters)
λ_k	The weight for the k-th feature ϕ_k

Table 1: Notation Table

3 Approach

We discuss our approach to semantic parsing in this section. The notation that we use in this paper is summarized in Table 1.

3.1 Model

In standard supervised syntactic parsing, one typically has access to a complete syntactic parse tree for each sentence in the training phase, which exactly tells the correct associations between words and syntactic labels. In our problem, however, each sentence is only paired with a complete semantic representation where the correct associations between words and semantic units are unavailable. We thus need to model such information with latent variables.

For a given $\mathbf{n}$-$\mathbf{m}$ pair (where $\mathbf{n}$ is a complete natural language sentence, and $\mathbf{m}$ is a complete semantic representation), we assume there exists a latent joint representation $\mathbf{h}$ that consists of both $\mathbf{n}$ and $\mathbf{m}$ which tells the correct associations between words and semantic units in such a pair. We use $\mathcal{H}(\mathbf{n}, \mathbf{m})$[1] to denote the set of all such possible

[1]We will give a concrete definition of $\mathcal{H}(\mathbf{n}, \mathbf{m})$ used for this work, which is the complete set of all possible *relaxed hybrid tree* structures for the $\mathbf{n}$-$\mathbf{m}$ pair, when we discuss our own joint representations later in Section 3.2.

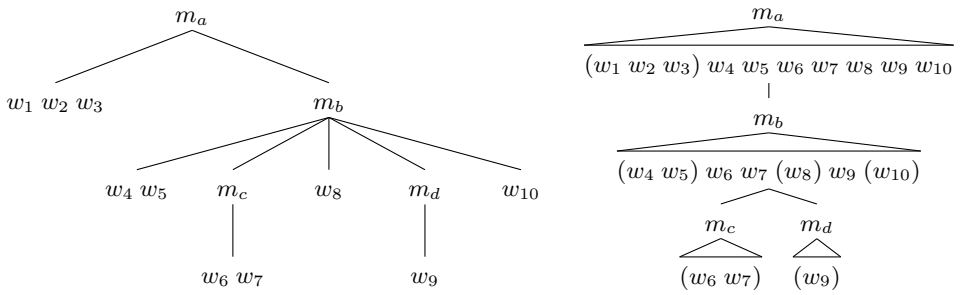


Figure 2: Two different ways of jointly representing sentences and their semantics. The hybrid tree representation of Lu et al. (2008) (left), and our novel *relaxed hybrid tree* representation (right). In our representation, a word w can be either *immediately* associated with its parent m (the words which appear inside the parenthesis), or *remotely* associated with m (the words that do not appear inside the parenthesis, and will also appear under a subtree rooted by one of m's children).

latent joint representations that contain both $\mathbf{n}$ and $\mathbf{m}$ exactly.

Given the joint representations, to model how the data is generated, one can either take a generative approach which models the joint probability distribution over $(\mathbf{n}, \mathbf{m}, \mathbf{h})$ tuples, or a discriminative approach which models the distribution over $(\mathbf{m}, \mathbf{h})$ tuples given the observation $\mathbf{n}$. Following several previous research efforts (Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2010; Liang et al., 2013), in this work we define a discriminative model using a log-linear approach:

$$P(\mathbf{m}, \mathbf{h}|\mathbf{n}; \Lambda) = \frac{e^{\Lambda \cdot \Phi(\mathbf{n}, \mathbf{m}, \mathbf{h})}}{\sum_{\mathbf{m}', \mathbf{h}' \in \mathcal{H}(\mathbf{n}, \mathbf{m}')} e^{\Lambda \cdot \Phi(\mathbf{n}, \mathbf{m}', \mathbf{h}')}} \quad (2)$$

Here $\Phi(\mathbf{n}, \mathbf{m}, \mathbf{h})$ is a function defined over the tuple $(\mathbf{n}, \mathbf{m}, \mathbf{h})$ that returns a vector consisting of counts of features associated with the tuple, and Λ is a vector consisting of feature weights, which are the parameters of the model.

In practice, we are only given the $\mathbf{n}$-$\mathbf{m}$ pairs but the latent structures are not observed. We therefore consider the following marginal probability:

$$P(\mathbf{m}|\mathbf{n}; \Lambda) = \sum_{\mathbf{h} \in \mathcal{H}(\mathbf{n}, \mathbf{m})} P(\mathbf{m}, \mathbf{h}|\mathbf{n}; \Lambda)$$
$$= \frac{\sum_{\mathbf{h} \in \mathcal{H}(\mathbf{n}, \mathbf{m})} e^{\Lambda \cdot \Phi(\mathbf{n}, \mathbf{m}, \mathbf{h})}}{\sum_{\mathbf{m}', \mathbf{h}' \in \mathcal{H}(\mathbf{n}, \mathbf{m}')} e^{\Lambda \cdot \Phi(\mathbf{n}, \mathbf{m}', \mathbf{h}')}} \quad (3)$$

The above probability is defined for a particular $\mathbf{n}$-$\mathbf{m}$ pair. The complete log-likelihood objective for the training set is:

$$\mathcal{L}(\Lambda) = \sum_i \log P(\mathbf{m}_i|\mathbf{n}_i; \Lambda) - \kappa||\Lambda||^2$$
$$= \sum_i \log \sum_{\mathbf{h} \in \mathcal{H}(\mathbf{n}_i, \mathbf{m}_i)} P(\mathbf{m}_i, \mathbf{h}|\mathbf{n}_i; \Lambda) - \kappa||\Lambda||^2 \quad (4)$$

where $(\mathbf{n}_i, \mathbf{m}_i)$ refers to the i-th instance in the training set. Note that here we introduce the additional regularization term $-\kappa \cdot ||\Lambda||^2$ to control over-fitting, where κ is a positive scalar.

Our goal is to maximize this objective function by tuning the model parameters Λ. Let's assume $\Lambda = \langle \lambda_1, \lambda_2, \ldots, \lambda_N \rangle$, where N is the total number of features (or the total number of parameters). Differentiating with respect to λ_k, the weight associated with the k-th feature ϕ_k, yields:

$$\frac{\partial \mathcal{L}(\Lambda)}{\partial \lambda_k} = \sum_i \sum_{\mathbf{h}} \mathbf{E}_{P(\mathbf{h}|\mathbf{n}_i, \mathbf{m}_i; \Lambda)}[\phi_k(\mathbf{n}_i, \mathbf{m}_i, \mathbf{h})]$$
$$- \sum_i \sum_{\mathbf{m}, \mathbf{h}} \mathbf{E}_{P(\mathbf{m}, \mathbf{h}|\mathbf{n}_i; \Lambda)}[\phi_k(\mathbf{n}_i, \mathbf{m}, \mathbf{h})] - 2\kappa\lambda_k \quad (5)$$

where $\phi_k(\mathbf{n}, \mathbf{m}, \mathbf{h})$ refers to the number of occurrences for the k-th feature in the tuple $(\mathbf{n}, \mathbf{m}, \mathbf{h})$.

Given the objective value (4) and gradients (5), standard methods such as stochastic gradient descent or L-BFGS (Liu and Nocedal, 1989) can be employed to optimize the objective function. We will discuss the computation of the objective function and gradients next.

3.2 Relaxed Hybrid Trees

To allow tractable computation of the values for the objective function (4) and the gradients (5),

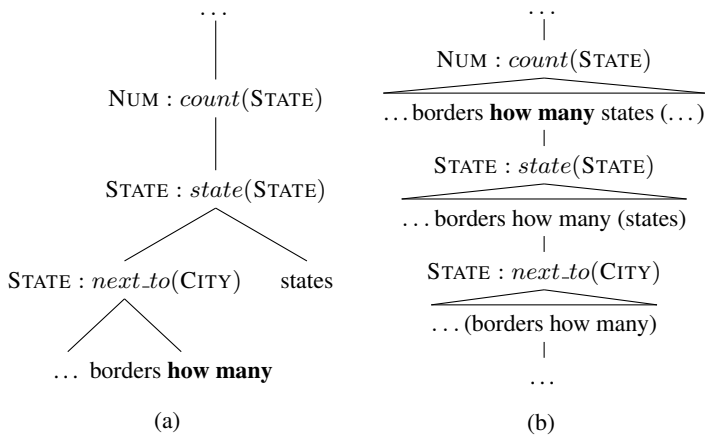


Figure 3: An example hybrid tree and an example *relaxed hybrid tree* representation. When the correct latent structure can not be found, the dependency between the words "how many" and the semantic unit "NUM : $count$(STATE)" can not be captured if the hybrid tree is used, whereas with our relaxed hybrid tree representation, such a dependency can still be captured.

certain restrictions on the latent structures ($\mathbf{h}$) will need to be imposed. We define in this section the set of all valid latent structures $\mathcal{H}(\mathbf{n}, \mathbf{m})$ for the $(\mathbf{n}, \mathbf{m})$ pair so that some efficient dynamic programming algorithms can be deployed.

We introduce our novel *relaxed hybrid tree* representations which jointly encode both natural language sentences and the tree-structured semantics. A *relaxed hybrid tree* $\mathbf{h}$ defined over $(\mathbf{n}, \mathbf{m})$ is a tree whose nodes are (n, m) pairs, where each n is a contiguous sequence of words from $\mathbf{n}$, and each m is a semantic unit (a tree node) from $\mathbf{m}$. For any two nodes $h_a \equiv (n_a, m_a)$ and $h_b \equiv (n_b, m_b)$ that appear in the relaxed hybrid tree $\mathbf{h}$, if h_a is the parent of h_b in $\mathbf{h}$, then m_a must also be the parent of m_b in $\mathbf{m}$, and n_a must contain n_b. If the lowest common ancestor of h_a and h_b in $\mathbf{h}$ is neither h_a nor h_b, then n_a and n_b do not share any common word. Note that words that appear at different positions in $\mathbf{n}$ are regarded as different words, regardless of their string forms.

Figure 2 (b) shows an example *relaxed hybrid tree* structure that we consider. Assume we would like to jointly represent both the natural language sentence $\mathbf{n} \equiv w_1 w_2 \ldots w_{10}$ and its corresponding semantic representation $\mathbf{m} \equiv m_a(m_b(m_c, m_d))$. In the given example, the semantic unit m_a maps to the complete sentence, m_b maps to the sequence $w_4 w_5 \ldots w_{10}$, m_c maps to $w_6 w_7$, and m_d maps to w_9. Certain words such as w_4 and w_{10} that appear directly below the semantic unit m_b but do not map to any of m_b's child semantic units are highlighted with parentheses "()", indicating they are *immediately associated* with m_b. These words play unique roles in the sub-tree rooted by m_b and are expected to be semantically closely related to m_b. Note that each word is immediately associated with *exactly one* semantic unit.

As a comparison, we also show an example hybrid tree representation (Lu et al., 2008) in Figure 2 (a) that has similar words-semantics correspondences. Different from our representation, the hybrid tree representation assumes each natural language word only maps to a single semantic unit (which is its immediate parent), and each semantic unit maps to a possibly discontiguous sequence of words. We believe that such a representation is overly restrictive, which might exhibit problems in cases where natural language sentences are highly non-isomorphic to their semantic tree structures. Under our relaxed hybrid tree representations, words that are immediately associated with a particular semantic unit now can also be *remotely associated* with all its parent semantic units as well. Essentially, our representation allows us to capture certain *unbounded* dependencies – for any word, as long as it appears below a certain semantic unit (in the relaxed hybrid tree), we can always capture the dependency between the two, regardless of which actual semantic unit that word is immediately associated with. Such an important relaxation allows some long-distance dependencies to be captured, which can potentially alleviate the sentence-semantics non-isomorphism issue reported in several earlier semantic parsing works (Kate and Mooney, 2006;

Wong and Mooney, 2007).

To better illustrate the differences, we show a concrete example in Figure 3, where the correct latent structure showing the correspondences between words and semantic units can not be found with the hybrid tree model. As a result, the hybrid tree model will fail to capture the correct dependency between the words "how many" and the semantic unit "NUM : $count$(STATE)". On the other hand, with our relaxed hybrid tree representation, such a dependency can still be captured, since these words will still be (remotely) associated with the semantic unit.

Such a relaxed hybrid tree representation, when further constrained with the word association patterns that we will introduce next, allows both the objective function (4) and the gradients of (5) to be computed through the dynamic programming algorithms to be presented in Section 4.

3.3 Word Association Patterns

As we have mentioned above, in the relaxed hybrid tree structures, each word w under a certain semantic unit m can either appear directly below m only (immediately associated with m), or can also appear in a subtree rooted by one of m's child semantic unit (remotely associated with m).

We allow several different ways for word associations and define the allowable patterns for semantic units with different number of arguments in Table 2. Such patterns are defined so that our model is amendable to dynamic programming algorithms to be discussed in Sec 4. In this table, $\mathbf{w}$ refers to a contiguous sequence of natural language words that are immediately associated with the current semantic unit, while $\mathbf{X}$ and $\mathbf{Y}$ refers to a sequence of natural language words that the first and second child semantic unit will map to, respectively.

For example, in Figure 2 (b), the word sequence directly below the semantic unit m_a follows the pattern $\mathbf{wX}$ (since the word sequence $w_1w_2w_3$ is immediately associated with m_a, and the remaining words are remotely associated with m_a), and the word sequence below m_b follows $\mathbf{wXwYw}$[2].

The word association patterns are similar to those *hybrid patterns* used in hybrid trees. One key difference is that we disallow the unary pattern $\mathbf{X}$. The reason is, when computing the partition function in Equation 3, inclusion of pattern $\mathbf{X}$ will result in relaxed hybrid trees consisting of an infinite number of nodes. However, this issue does not come up in the original *hybrid tree* models due to their generative setting, where the training process does not involve such a partition function.

#Args	Word Association Patterns
0	$\mathbf{w}$
1	$\mathbf{wX}$, $\mathbf{Xw}$, $\mathbf{wXw}$
2	$\mathbf{XY}$, $\mathbf{YX}$, $\mathbf{wXY}$, $\mathbf{wYX}$, $\mathbf{XwY}$, $\mathbf{YwX}$ $\mathbf{XYw}$, $\mathbf{YXw}$, $\mathbf{wXwY}$, $\mathbf{wYwX}$ $\mathbf{wXYw}$, $\mathbf{wYXw}$, $\mathbf{XwYw}$, $\mathbf{YwXw}$ $\mathbf{wXwYw}$, $\mathbf{wYwXw}$

Table 2: The complete list of word association patterns. Here #Args means the number of arguments for a semantic unit.

3.4 Features

The features are defined over the $(\mathbf{n}, \mathbf{m}, \mathbf{h})$ tuples. In practice, we define features at each level of the relaxed hybrid tree structure $\mathbf{h}$. In other words, features are defined over (n, m) tuples where n is a contiguous sequence of natural language words (immediately or remotely) associated with the semantic unit m (recall that $\mathbf{h}$ contains both $\mathbf{n}$ and $\mathbf{m}$, and each level of $\mathbf{h}$ simply consists of a semantic unit and a contiguous sequence of words). Each feature over (n, m) is then further decomposed as a product between two indicator feature functions, defined over the natural language words (n) and semantic unit (m) respectively: $\phi(n, m) = \phi^i(n) \times \phi^o(m)$. For each $\phi^i(n)$ we define two types of features: the *local features*, which are defined over immediately associated words only, and the *span features*, which are defined over all (immediately or remotely) associated words to capture long range dependencies.

The local features include word unigrams and bigrams, the word association patterns, as well as character-level features[3] which perform implicit morphological analysis. The span features include word unigrams, bigrams, as well as trigrams. Although our model allows certain more sophisticated features to be exploited, such as word POS features, word similarity features based on the WordNet (Pedersen et al., 2004), we deliberately choose to only include these simple features so

[2]This is based on the assumption that m_c and m_d are the first and second child of m_b in the semantic representation, respectively. If m_d is the first child in the semantic representation and m_c is the second, the pattern should be $\mathbf{wYwXw}$.

[3]For each word, we used all its prefixes (not necessarily linguistically meaningful ones) whose length are greater than 2 as features, for all languages.

as to make a fair comparison with previous works which also did not make use of external resources. For the features defined on m (i.e., $\phi^o(m)$), we include only the string form of m, as well as m's function name as features.

Finally, we also define features over $\mathbf{m}$ only. Such features are defined over semantic unit pairs such as (m_a, m_b) where m_a is the parent node of m_b as in $\mathbf{m}$. They include: 1) concatenation of the string forms of m_a and m_b, 2) concatenation of the string form of m_a and m_b's type, and 3) concatenation of the function names of m_a and m_b.

4 Algorithms

In this section we describe the efficient algorithms used for learning and decoding. The algorithms are inspired by the inside-outside style algorithms used for the generative hybrid tree models (Lu et al., 2008), but are different in the following ways: 1) we need to handle features, including long-distance features, 2) we need to additionally handle the computation of the partition function of Equation (3).

4.1 Learning

The training process involves the computation of the objective function (4) as well as the gradient terms (5).

The objective function (4) (excluding the regularization term which can be trivially computed) is equivalent to the following:

$$\mathcal{L}(\Lambda) = \sum_i \log \sum_{\mathbf{h} \in \mathcal{H}(\mathbf{n}_i, \mathbf{m}_i)} e^{\Lambda \cdot \Phi(\mathbf{n}_i, \mathbf{m}_i, \mathbf{h})} - \sum_i \log \sum_{\mathbf{m}', \mathbf{h}' \in \mathcal{H}(\mathbf{n}_i, \mathbf{m}')} e^{\Lambda \cdot \Phi(\mathbf{n}_i, \mathbf{m}', \mathbf{h}')} \quad (6)$$

In the first term, $\sum_{\mathbf{h} \in \mathcal{H}(\mathbf{n}_i, \mathbf{m}_i)} e^{\Lambda \cdot \Phi(\mathbf{n}_i, \mathbf{m}_i, \mathbf{h})}$ is in fact the sum of the scores (as defined by Φ and Λ) associated with all such latent structures that contain both $\mathbf{m}_i$ and $\mathbf{n}_i$ exactly. The second term is the sum of the scores associated with all the latent structures that contain $\mathbf{n}_i$ exactly. We focus our discussions on the computation of the first part first.

We use $\overset{m_{(p)}}{\underset{w_i \ldots w_j}{\triangle}}$ to denote the combined score of all such latent relaxed hybrid tree structures that contain both the semantic tree rooted by m and the natural language word sequence $w_i \ldots w_j$ that forms the word association pattern p with respect to m. For example, the score of the relaxed hybrid tree in Figure 2 (b) is contained by $\overset{m_{a(\mathbf{wX})}}{\underset{w_1 \ldots w_{10}}{\triangle}}$ (here p = $\mathbf{wX}$ because only $w_1 w_2 w_3$ are immediately associated with m_a).

We give an illustrative example that shows how these scores can be computed efficiently using dynamic programming. Consider the following case when m has at least one child semantic unit:

$$\overset{m_{(\mathbf{wXw})}}{\underset{w_i \ldots w_j}{\triangle}} = \overset{m_{(\mathbf{w})}}{\underset{w_i}{\triangle}} \otimes \overset{m_{(\mathbf{wXw})}}{\underset{w_{i+1} \ldots w_j}{\triangle}}$$

$$+ \overset{m_{(\mathbf{w})}}{\underset{w_i}{\triangle}} \otimes \overset{m_{(\mathbf{Xw})}}{\underset{w_{i+1} \ldots w_j}{\triangle}}$$

Here the symbol $\otimes$ means *extract and compute*, a process that involves 1) extraction of additional features when the two structures on the right-hand side are put together (for example, the local bigram feature "$w_i w_{i+1}$" can be extracted in the above case), and 2) computation of the score for the new structure when the two structures from both sides of $\otimes$ are combined, based on the scores of these structures and newly extracted features.

The above equation holds because for any relaxed hybrid tree contained by the left-hand side, the left-most word w_i is always immediately associated with m. The term $\overset{m_{(\mathbf{wXw})}}{\underset{w_{i+1} \ldots w_j}{\triangle}}$ is presenting a similar but smaller structure to the term on the left-hand side. The other term $\overset{m_{(\mathbf{wX})}}{\underset{w_i \ldots w_j}{\triangle}}$ can also be computed based on similar equations. In other words, such terms can be computed from even smaller similar terms in a recursive manner. A bottom-up dynamic programming algorithm is used for computing such terms.

When the semantic unit m has two child nodes, similar equations can also be established. Here we give an illustrative example:

$$\overset{m_{(\mathbf{wXwYw})}}{\underset{w_i \ldots w_j}{\triangle}} = \sum_{k=i}^{j-1} \overset{m_{(\mathbf{wX})}}{\underset{w_i \ldots w_k}{\triangle}} \otimes \overset{m_{(\mathbf{Yw})}}{\underset{w_{k+1} \ldots w_j}{\triangle}}$$

Finally, we have the following equation:

$$\overset{m}{\underset{w_i \ldots w_j}{\triangle}} = \sum_p \overset{m_{(p)}}{\underset{w_i \ldots w_j}{\triangle}}$$

The left-hand side simply means the combined score for all such relaxed hybrid trees that have (n, m) as the root, where $n \equiv w_i \ldots w_j$. Once the computation for a certain (n, m) pair is done, we

can move up to process such pairs that involve m's parent node.

The above process essentially computes the *inside* score associated with the (n, m) pair, which gives the sum of the scores of all such (incomplete) relaxed hybrid trees that can be constructed with (n, m) as the root. Similar to (Lu et al., 2008), we can also define and compute the *outside* scores for (n, m) (the combined score of such incomplete relaxed hybrid trees that contain (n, m) as one of its leave nodes) in an analogous manner, where the computation of the gradient functions can be efficiently integrated in this process.

Computation of the second part of the objective function (6) involves dynamic programming over a packed forest representation rather than a single tree, which requires an extension to the algorithm described in (Lu et al., 2008). The resulting algorithm is similar to the one used in (Lu and Ng, 2011), which has been used for language generation from packed forest representations of typed λ-calculus expressions.

4.2 Decoding

The decoding phase involves finding the optimal semantic tree $\mathbf{m}^*$ given a new input sentence $\mathbf{n}$:

$$\mathbf{m}^* = \arg\max_{\mathbf{m}} P(\mathbf{m}|\mathbf{n}) \tag{7}$$

This in fact is equivalent to finding the following optimal semantic tree $\mathbf{m}^*$:

$$\mathbf{m}^* = \arg\max_{\mathbf{m}} \sum_{\mathbf{h} \in \mathcal{H}(\mathbf{n},\mathbf{m})} e^{\Lambda \cdot \Phi(\mathbf{n},\mathbf{m},\mathbf{h})} \tag{8}$$

Unfortunately, the summation operation inside the $\arg\max$ prevents us from employing a similar version of the dynamic programming algorithm we developed for learning in Section 4.1. To overcome this difficulty, we instead find the optimal semantic tree using the following equation:

$$\mathbf{m}^* = \arg\max_{\mathbf{m},\mathbf{h} \in \mathcal{H}(\mathbf{n},\mathbf{m})} e^{\Lambda \cdot \Phi(\mathbf{n},\mathbf{m},\mathbf{h})} \tag{9}$$

We essentially replace the $\sum$ operation by the $\max$ operation inside the $\arg\max$. In other words, we first find the best latent relaxed hybrid tree $\mathbf{h}^*$ that contains the input sentence $\mathbf{n}$, and next we extract the optimal semantic tree $\mathbf{m}^*$ from $\mathbf{h}^*$.

This decoding algorithm is similar to the dynamic programming algorithm used for computing the inside score for a given natural language sentence $\mathbf{n}$ (i.e., the algorithm for computing the second term of Equation (6)). The difference here is, at each intermediate step, instead of computing the combined score for all possible relaxed hybrid tree structures (i.e., performing `sum`), we find the single-best relaxed hybrid tree structure (i.e., performing $\max$).

5 Experiments

We present evaluations on the standard GeoQuery dataset which is publicly available. This dataset has been used for evaluations in various semantic parsing works (Wong and Mooney, 2006; Kate and Mooney, 2006; Lu et al., 2008; Jones et al., 2012). It consists of 880 natural language sentences paired with their corresponding formal semantic representations. Each semantic representation is a tree structured representation derived from a Prolog query that can be used to interact with a database of U.S. geography facts for retrieving answers. The original dataset was fully annotated in English, and recently Jones et al. (2012) released a new version of this dataset with three additional language annotations (German, Greek and Thai). For all the experiments, we used the identical experimental setup as described in Jones et al. (2012). Specifically, we trained on 600 instances, and evaluated on the remaining 280.

We note that there exist two different versions of the GeoQuery dataset annotated with completely different semantic representations. Besides the version that we use in this work, which is annotated with tree structured semantic representations, the other version is annotated with lambda calculus expressions (Zettlemoyer and Collins, 2005). Results obtained from these two versions are not comparable.[4] Like many previous works, we focus on tree structured semantic representations for evaluations in this work since our model is designed for handling the class of semantic representations with recursive tree structures.

We used the standard evaluation criteria for judging the correctness of the outputs. Specifically, our system constructs Prolog queries from the output parses, and uses such queries to retrieve answers from the GeoQuery database. An output is considered correct if and only if it retrieves the

[4] Kwiatkowski et al. (2010) showed in Table 3 of their work that the version with tree-structured representations appeared to be more challenging – their semantic parser's performance on this version was substantially lower than that on the lambda calculus version.

System	English		Thai		German		Greek	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
WASP	71.1	77.7	71.4	75.0	65.7	74.9	70.7	**78.6**
HYBRIDTREE+	76.8	81.0	73.6	76.7	62.1	68.5	69.3	74.6
UBL-S	82.1	82.1	66.4	66.4	**75.0**	**75.0**	73.6	73.7
TREETRANS	79.3	79.3	78.2	78.2	74.6	74.6	75.4	75.4
RHT (*all features*)	**83.6**	**83.6**	**79.3**	**79.3**	74.3	74.3	**78.2**	78.2

Table 3: Performance on the benchmark data, using four different languages as inputs. RHT: relaxed hybrid tree (this work).

same answers as the gold standard (Jones et al., 2012). We report accuracy scores – the percentage of inputs with correct answers, and F1 measures – the harmonic mean of precision (the proportion of correct answers out of inputs with an answer) and recall (the proportion of correct answers out of all inputs). By adopting such an evaluation method we will be able to directly compare our model's performance against those of the previous works.

The evaluations were conducted under such a setting in order to make comparisons to previous works. We would like to stress that our model is designed for general-purpose semantic parsing that is not only natural language-independent, but also task-independent. We thus distinguish our work from several previous works in the literature which focused on semantic parsing under other assumptions. Specifically, for example, works such as (Liang et al., 2013; Poon and Domingos, 2009; Clarke et al., 2010) essentially performed semantic parsing under different settings where the goal was to optimize the performance of certain downstream NLP tasks such as answering questions, and different semantic formalisms and language-specific features were usually involved.

For all our experiments, we used the L-BFGS algorithm for learning the feature weights, where feature weights were all initialized to zeros and the regularization hyper-parameter κ was set to 0.01. We set the maximum number of L-BFGS steps to 100. When all the features are considered, our model creates over 2 million features for each language on the dataset (English: 2.1M, Thai: 2.3M, German: 2.7M, Greek: 2.6M). Our model requires (on average) a per-instance learning time of 0.428 seconds and a per-instance decoding time of 0.235 seconds, on an Intel machine with a 2.2 GHz CPU. Our implementation is in Java. Here the per-instance learning time refers to the time spent on computing the instance-level log-likelihood as well as the expected feature counts (needed for the gradients).

Table 3 shows the evaluation results of our system as well as those of several other comparable previous works which share the same experimental setup as ours. UBL-S is the system presented in Kwiatkowski et al. (2010) which performs semantic parsing with the CCG based on mapping between graphs, and is the only non-tree based top-performing system. Their system, similar to ours, also uses a discriminative log-linear model where two types of features are defined. WASP is a model based on statistical phrase-based machine translation as we have described earlier. The hybrid tree model (HYBRIDTREE+) performs learning using a generative process which is augmented with an additional discriminative-reranking stage, where certain global features are incorporated (Lu et al., 2008). The Bayesian tree transducer model (TREETRANS) learns under a Bayesian generative framework, using hyper-parameters manually tuned on the German training data.

We can observe from Table 3 that the semantic parser based on relaxed hybrid tree gives competitive performance when all the features (described in Sec 3.4) are used. It significantly outperforms the hybrid tree model that is augmented with a discriminative reranking step. The model reports the best accuracy and F1 scores on English and Thai and best accuracy score on Greek. The scores on German are lower than those of UBL-S and TREETRANS, mainly because the span features appear not to be effective for this language, as we will discuss next.

We report in Table 4 the test set performance when certain types of features are excluded from our system. Such results can help us understand the effectiveness of features of different types. As we can see from the table, in general, all features play essential roles, though their effective-

System	English		Thai		German		Greek	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
RHT (*all features*)	**83.6**	**83.6**	79.3	79.3	74.3	74.3	78.2	78.2
RHT (*no local features*)	81.4	81.4	78.2	78.2	74.3	74.3	75.7	75.7
RHT (*no span features*)	81.1	81.1	77.9	77.9	**78.2**	**78.2**	**78.9**	**78.9**
RHT (*no char features*)	79.6	79.6	**82.1**	**82.1**	73.6	73.6	76.1	76.1

Table 4: Results when certain types of features (local features, span features and character-level features) are excluded.

ness vary across different languages. The local features, which capture local dependencies, are of particular importance. Performance on three languages (English, Thai, and Greek) will drop when such features are excluded. Character-level features are very helpful for the three European languages (English, German, and Greek), but appear to be harmful for Thai. This indicates the character-level features that we propose do not perform effective morphological analysis for this Asian language.[5] The span features, which are able to capture certain long-distance dependencies, also play important roles. Specifically, if such features are excluded, our model's performance on three languages (Greek, English, Thai) will drop. Such features do not appear to be helpful for Thai and appear to be harmful for German. Clearly, such long-distance features are not contributing useful information to the model when these two languages are considered. This is especially the case for German, where we believe such features are contributing substantial noisy information to the model. What underlying language-specific, syntactic properties are generally causing these gaps in the performances? We believe this is an important question that needs to be addressed in future research. As we have mentioned, to make an appropriate comparison with previous works, only simple features are used. We believe that our system's performance can be further improved when additional informative language-specific features can be extracted from effective language tools and incorporated into our system.

6 Conclusions

In this work, we present a new discriminative model for semantic parsing which extends the *hybrid tree* model. Such an extension is similar to the extension of the generative syntactic parser based on probabilistic context-free grammars (PCFG) to the feature-based CRF parser (Finkel et al., 2008), but is slightly more complex due to latent structures. Developed on top of our novel *relaxed hybrid tree* representations, our model allows certain long-distance dependencies to be captured. We also present efficient algorithms for learning and decoding. Experiments on benchmark data show that our model is competitive to previous works and achieves the state-of-the-art performance across several different languages.

Future works include development of efficient algorithms for feature-based semantic parsing with alternative loss functions (Zhou et al., 2013), development of feature-based language generation models (Lu et al., 2009; Lu and Ng, 2011) and multilingual semantic parsers (Jie and Lu, 2014), as well as the development of efficient semantic parsing algorithms for optimizing the performance of certain downstream NLP tasks with less supervision (Clarke et al., 2010; Liang et al., 2013).

Being able to efficiently exploit features defined over individual words, our model also opens up the possibility for us to exploit alternative representations of words for learning (Turian et al., 2010), or to perform joint learning of both distributional and logical semantics (Lewis and Steedman, 2013). Furthermore, as a general string-to-tree structured prediction model, this work may find applications in other areas within NLP.

The system and code can be downloaded from `http://statnlp.org/research/sp/`.

Acknoledgments

The author would like to thank the anonymous reviewers for their helpful comments. This work was supported by SUTD grant SRG ISTD 2013 064.

[5]The character-level features that we introduced are indeed very general. We have conducted several additional experiments, which show that our model's performance for each language can be further improved when certain language-specific character-level features are introduced.

References

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

David Chiang. 2007. Hierarchical phrase-based translation. *Comput. Linguist.*, 33(2):201–228, June.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proc. of CONLL '10*, pages 18–27.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proc. of ACL/HLT*, pages 959–967.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proc. of CONLL '05*, pages 9–16.

Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *Proc. of ACL '11*, pages 1486–1495.

Zhanming Jie and Wei Lu. 2014. Multilingual semantic parsing: Parsing multiple languages into semantic representations. In *Proc. of COLING*, pages 1291–1301.

Bevan Keeley Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic parsing with bayesian tree transducers. In *Proc. of ACL '12*, pages 488–496.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proc. of COLING/ACL*, pages 913–920.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proc. EMNLP'10*, pages 1223–1233.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

D. C. Liu and J. Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(3):503–528, December.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proc. of EMNLP '11*, pages 1611–1622.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proc. of EMNLP '08*, pages 783–792.

Wei Lu, Hwee Tou Ng, and Wee Sun Lee. 2009. Natural language generation with tree conditional random fields. In *Proc. of EMNLP*, pages 400–409.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet:: Similarity: measuring the relatedness of concepts. In *Proc. of HLT-NAACL '04 (Demonstration)*, pages 38–41.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proc. of EMNLP '09*, pages 1–10.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Mark Steedman. 1996. *Surface structure and interpretation*. MIT press.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proc. of ACL '10*, pages 384–394.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proc. of HLT/NAACL '06*, pages 439–446.

Yuk Wah Wong and Raymond J Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proc. of ACL '07*.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of UAI '05*.

Junsheng Zhou, Juhong Xu, and Weiguang Qu. 2013. Efficient latent structural perceptron with hybrid trees for semantic parsing. In *IJCAI*, pages 2246–2252.

Low-dimensional Embeddings for Interpretable Anchor-based Topic Inference

Moontae Lee
Dept. of Computer Science
Cornell University
Ithaca, NY, 14853
moontae@cs.cornell.edu

David Mimno
Dept. of Information Science
Cornell University
Ithaca, NY, 14853
mimno@cornell.edu

Abstract

The anchor words algorithm performs provably efficient topic model inference by finding an approximate convex hull in a high-dimensional word co-occurrence space. However, the existing greedy algorithm often selects poor anchor words, reducing topic quality and interpretability. Rather than finding an approximate convex hull in a high-dimensional space, we propose to find an exact convex hull in a visualizable 2- or 3-dimensional space. Such low-dimensional embeddings both improve topics and clearly show users why the algorithm selects certain words.

1 Introduction

Statistical topic modeling is useful in exploratory data analysis (Blei et al., 2003), but model inference is known to be NP-hard even for the simplest models with only two topics (Sontag and Roy, 2011), and training often remains a black box to users. Likelihood-based training requires expensive approximate inference such as variational methods (Blei et al., 2003), which are deterministic but sensitive to initialization, or Markov chain Monte Carlo (MCMC) methods (Griffiths and Steyvers, 2004), which have no finite convergence guarantees. Recently Arora et al. proposed the Anchor Words algorithm (Arora et al., 2013), which casts topic inference as statistical recovery using a *separability assumption*: each topic has a specific anchor word that appears only in the context of that single topic. Each anchor word can be used as a unique pivot to disambiguate the corresponding topic distribution. We then reconstruct the word co-occurrence pattern of each non-anchor words as a convex combination of the co-occurrence patterns of the anchor words.

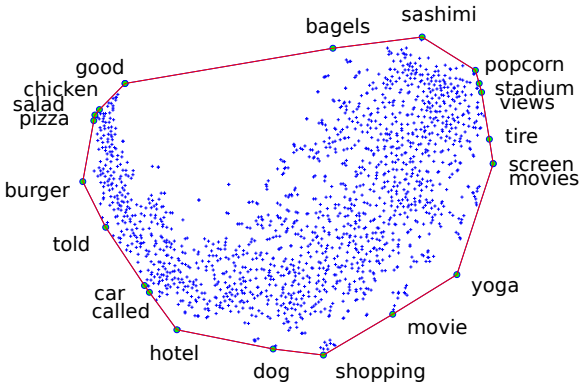


Figure 1: 2D t-SNE projection of a Yelp review corpus and its convex hull. The words corresponding to vertices are anchor words for topics, whereas non-anchor words correspond to the interior points.

This algorithm is fast, requiring only one pass through the training documents, and provides provable guarantees, but results depend entirely on selecting good anchor words. (Arora et al., 2013) propose a greedy method that finds an approximate convex hull around a set of vectors corresponding to the word co-occurrence patterns for each vocabulary word. Although this method is an improvement over previous work that used impractical linear programming methods (Arora et al., 2012), serious problems remain. The method greedily chooses the farthest point from the current subspace until the given number of anchors have been found. Particularly at the early stages

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1319–1328,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

of the algorithm, the words associated with the farthest points are likely to be infrequent and idiosyncratic, and thus form poor bases for human interpretation and topic recovery. This poor choice of anchors noticeably affects topic quality: the anchor words algorithm tends to produce large numbers of nearly identical topics.

Besides providing a separability criterion, anchor words also have the potential to improve topic interpretability. After learning topics for given text collections, users often request a label that summarizes each topic. Manually labeling topics is arduous, and labels often do not carry over between random initializations and models with differing numbers of topics. Moreover, it is hard to control the subjectivity in labelings between annotators, which is open to interpretive errors. There has been considerable interest in automating the labeling process (Mei et al., 2007; Lau et al., 2011; Chuang et al., 2012). (Chuang et al., 2012) propose a measure of *saliency*: a good summary term should be both distinctive specifically to one topic and probable in that topic. Anchor words are by definition optimally distinct, and therefore may seem to be good candidates for topic labels, but greedily selecting extreme words often results in anchor words that have low probability.

In this work we explore the opposite of Arora et al.'s method: rather than finding an approximate convex hull for an exact set of vectors, we find an exact convex hull for an approximate set of vectors. We project the $V \times V$ word co-occurrence matrix to visualizable 2- and 3-dimensional spaces using methods such as t-SNE (van der Maaten and Hinton, 2008), resulting in an input matrix up to 3600 times narrower than the original input for our training corpora. Despite this radically low-dimensional projection, the method not only finds topics that are as good or better than the greedy anchor method, it also finds highly salient, interpretable anchor words and provides users with a clear visual explanation for why the algorithm chooses particular words, all while maintaining the original algorithm's computational benefits.

2 Related Work

Latent Dirichlet allocation (LDA) (Blei et al., 2003) models D documents with a vocabulary V using a predefined number of topics by K. LDA views both $\{A_k\}_{k=1}^K$, a set of K topic-word distributions for each topic k, and $\{W_d\}_{d=1}^D$, a set of D document-topic distributions for each document d, and $\{\mathbf{z}_d\}_{d=1}^D$, a set of topic-assignment vectors for word tokens in the document d, as randomly generated from known stochastic processes. Merging $\{A_k\}$ as k-th column vector of $V \times K$ matrix A, $\{W_d\}$ as d-th column vector of $K \times D$ matrix W, the learning task is to estimate the posterior distribution of latent variables A, W, and $\{\mathbf{z}_d\}$ given $V \times D$ word-document matrix $\hat{M}$, which is the only observed variable where d-th column corresponds to the empirical word frequencies in the training documents d.

(Arora et al., 2013) recover word-topic matrix A and topic-topic matrix $R = \mathbb{E}[WW^T]$ instead of W in the spirit of nonnegative matrix factorization. Though the true underlying word distribution for each document is unknown and could be far from the sample observation $\hat{M}$, the empirical word-word matrix $\hat{Q}$ converges to its expectation $A\mathbb{E}[WW^T]A^T = ARA^T$ as the number of documents increases. Thus the learning task is to approximately recover A and R pretending that the empirical $\hat{Q}$ is close to the true second-order moment matrix Q.

The critical assumption for this method is to suppose that every topic k has a specific anchor word s_k that occurs with non-negligible probability (> 0) only in that topic. The anchor word s_k need not always appear in every document about the topic k, but we can be confident that the document is at least to some degree about the topic k if it contains s_k. This assumption drastically improves inference by guaranteeing the presence of a diagonal sub-matrix inside the word-topic matrix A. After constructing an estimate $\hat{Q}$, the algorithm in (Arora et al., 2013) first finds a set $S = \{s_1, ..., s_K\}$ of K anchor words (K is user-specified), and recovers A and R subsequently based on S. Due to this structure, overall performance depends heavily on the quality of anchor words.

In the matrix algebra literature this greedy anchor finding method is called *QR with row-pivoting*. Previous work classifies a matrix into two sets of row (or column) vectors where the vectors in one set can effectively reconstruct the vectors in another set, called *subset-selection algorithms*. (Gu and Eisenstat, 1996) suggest one important deterministic algorithm. A randomized algorithm provided by (Boutsidis et al., 2009) is the state-of-the art using a pre-stage that selects the

candidates in addition to (Gu and Eisenstat, 1996). We found no change in anchor selection using these algorithms, verifying the difficulty of the anchor finding process. This difficulty is mostly because anchors must be nonnegative convex bases, whereas the classified vectors from the subset selection algorithms yield unconstrained bases.

The t-SNE model has previously been used to display high-dimensional embeddings of words in 2D space by Turian.[1] Low-dimensional embeddings of topic spaces have also been used to support user interaction with models: (Eisenstein et al., 2011) use a visual display of a topic embedding to create a navigator interface. Although t-SNE has been used to visualize the *results* of topic models, for example by (Lacoste-Julien et al., 2008) and (Zhu et al., 2009), we are not aware of any use of the method as a fundamental component of topic inference.

3 Low-dimensional Embeddings

Real text corpora typically involve vocabularies in the tens of thousands of distinct words. As the input matrix $\hat{Q}$ scales quadratically with V, the Anchor Words algorithm must depend on a low-dimensional projection of $\hat{Q}$ in order to be practical. Previous work (Arora et al., 2013) uses random projections via either Gaussian random matrices (Johnson and Lindenstrauss, 1984) or sparse random matrices (Achlioptas, 2001), reducing the representation of each word to around 1,000 dimensions. Since the dimensionality of the compressed word co-occurrence space is an order of magnitude larger than K, we must still approximate the convex hull by choosing extreme points as before.

In this work we explore two projection methods: PCA and t-SNE (van der Maaten and Hinton, 2008). Principle Component Analysis (PCA) is a commonly-used dimensionality reduction scheme that linearly transforms the data to new coordinates where the largest variances are orthogonally captured for each dimension. By choosing only a few such principle axes, we can represent the data in a lower dimensional space. In contrast, t-SNE embedding performs a non-linear dimensionality reduction preserving the local structures. Given a set of points $\{\mathbf{x}_i\}$ in a high-dimensional space X, t-SNE allocates probability mass for each pair of points so that pairs of similar (closer) points become more likely to co-occur than dissimilar (distant) points.

[1] http://metaoptimize.com/projects/wordreprs/

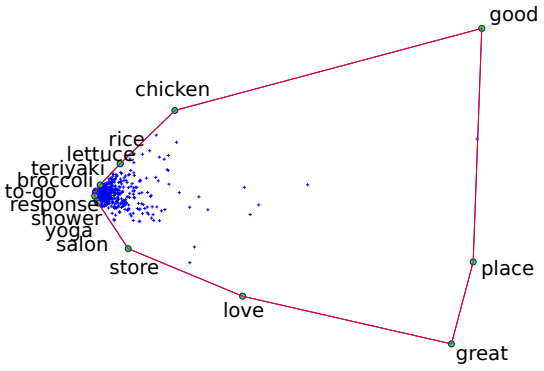


Figure 2: 2D PCA projections of a Yelp review corpus and its convex hulls.

$$p_{j|i} = \frac{\exp(-d(\mathbf{x}_i, \mathbf{x}_j)^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-d(\mathbf{x}_i, \mathbf{x}_k)^2/2\sigma_i^2)} \quad (1)$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \quad \text{(symmetrized)} \quad (2)$$

Then it generates a set of new points $\{\mathbf{y}_i\}$ in low-dimensional space Y so that probability distribution over points in Y behaves similarly to the distribution over points in X by minimizing KL-divergence between two distributions:

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}} \quad (3)$$

$$min\ KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \quad (4)$$

Instead of approximating a convex hull in such a high-dimensional space, we select the exact vertices of the convex hull formed in a low-dimensional projected space, which can be calculated efficiently. Figures 1 and 2 show the convex hulls for 2D projections of $\hat{Q}$ using t-SNE and PCA for a corpus of Yelp reviews. Figure 3 illustrates the convex hulls for 3D t-SNE projection for the same corpus. Anchor words correspond to the vertices of these convex hulls. Note that we present the 2D projections as illustrative examples only; we find that three dimensional projections perform better in practice.

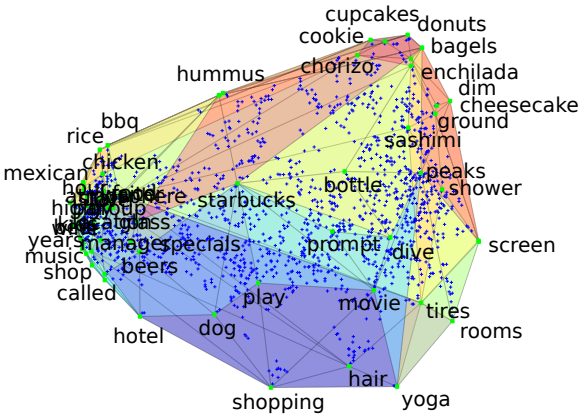


Figure 3: 3D t-SNE projection of a Yelp review corpus and its convex hull. Vertices on the convex hull correspond to anchor words.

In addition to the computational advantages, this approach benefits anchor-based topic modeling in two aspects. First, as we now compute the exact convex hull, the number of topics depends on the dimensionality of the embedding, v. For example in the figures, 2D projection has 21 vertices, whereas 3D projection supports 69 vertices. This implies users can easily tune the granularity of topic clusters by varying $v = 2, 3, 4, ...$ without increasing the number of topics by one each time. Second, we can effectively visualize the thematic relationships between topic anchors and the rest of words in the vocabulary, enhancing both interpretability and options for further vocabulary curation.

4 Experimental Results

We find that radically low-dimensional t-SNE projections are effective at finding anchor words that are much more salient than the greedy method, and topics that are more distinctive, while maintaining comparable held-out likelihood and semantic coherence. As noted in Section 1, the previous greedy anchor words algorithm tends to produce many nearly identical topics. For example, 37 out of 100 topics trained on a 2008 political blog corpus have *obama, mccain, bush, iraq* or *palin* as their most probable word, including 17 just for *obama*. Only 66% of topics have a unique top word. In contrast, the t-SNE model on the same dataset has only one topic whose most probable word is *obama*, and 86% of topics have a unique top word (*mccain* is the most frequent top word, with five topics).

We use three real datasets: business reviews from the Yelp Academic Dataset,[2] political blogs from the 2008 US presidential election (Eisenstein and Xing, 2010), and New York Times articles from 2007.[3] Details are shown in Table 1. Documents with fewer than 10 word tokens are discarded due to possible instability in constructing $\hat{Q}$. We perform minimal vocabulary curation, eliminating a standard list of English stopwords[4] and terms that occur below frequency cutoffs: 100 times (Yelp, Blog) and 150 times (NYT). We further restrict possible anchor words to words that occur in more than three documents. As our datasets are not artificially synthesized, we reserve 5% from each set of documents for held-out likelihood computation.

Name	Documents	Vocab.	Avg. length
Yelp	20,000	1,606	40.6
Blog	13,000	4,447	161.3
NYT	41,000	10,713	277.8

Table 1: Statistics for datasets used in experiments

Unlike (Arora et al., 2013), which presents results on synthetic datasets to compare performance across different *recovery* methods given increasing numbers of documents, we are are interested in comparing *anchor finding* methods, and are mainly concerned with semantic quality. As a result, although we have conducted experiments on synthetic document collections,[5] we focus on real datasets for this work. We also choose to compare only anchor finding algorithms, so we do not report comparisons to likelihood-based methods, which can be found in (Arora et al., 2013).

For both PCA and t-SNE, we use three-dimensional embeddings across all experiments. This projection results in matrices that are 0.03% as wide as the original $V \times V$ matrix for the NYT dataset. Without low-dimensional embedding, each word is represented by a V-dimensional vector where only several terms are non-zero due to the sparse co-occurrence patterns. Thus a ver-

[2] https://www.yelp.com/academic_dataset

[3] http://catalog.ldc.upenn.edu/LDC2008T19

[4] We used the list of 524 stop words included in the Mallet library.

[5] None of the algorithms are particularly effective at finding synthetically introduced anchor words possibly because there are other candidates around anchor vertices that approximate the convex hull to almost the same degree.

tex captured by the greedy anchor-finding method is likely to be one of many eccentric vertices in very high-dimensional space. In contrast, t-SNE creates an effective dense representation where a small number of pivotal vertices are more clearly visible, improving both performance and interpretability.

Note that since we can find an *exact* convex hull in these spaces,[6] there is an upper bound to the number of topics that can be found given a particular projection. If more topics are desired, one can simply increase the dimensionality of the projection. For the greedy algorithm we use sparse random projections with 1,000 dimensions with 5% negative entries and 5% positive entries. PCA and t-SNE choose (49, 32, 47) and (69, 77, 107) anchors, respectively for each of three Yelp, Blog, and NYTimes datasets.

4.1 Anchor-word Selection

We begin by comparing the behavior of low-dimensional embeddings to the behavior of the standard greedy algorithm. Table 2 shows ordered lists of the first 12 anchor words selected by three algorithms: t-SNE embedding, PCA embedding, and the original greedy algorithm. Anchor words selected by t-SNE (*police, business, court*) are more general than anchor words selected by the greedy algorithm (*cavalry, al-sadr, yiddish*). Additional examples of anchor words and their associated topics are shown in Table 3 and discussed in Section 4.2.

#	t-SNE	PCA	Greedy
1	police	beloved	cavalry
2	bonds	york	biodiesel
3	business	family	h/w
4	day	loving	kingsley
5	initial	late	mourners
6	million	president	pcl
7	article	people	carlisle
8	wife	article	al-sadr
9	site	funeral	kaye
10	mother	million	abc's
11	court	board	yiddish
12	percent	percent	great-grandmother

Table 2: The first 12 anchor words selected by three algorithms for the NYT corpus.

The Gram-Schimdt process used by Arora et al. greedily selects anchors in high-dimensional space. As each word is represented within V-dimensions, finding the word that has the next most distinctive co-occurrence pattern tends to prefer overly eccentric words with only short, intense bursts of co-occurring words. While the bases corresponding to these anchor words could be theoretically relevant for the original space in high-dimension, they are less likely to be equally important in low-dimensional space. Thus projecting down to low-dimensional space can rearrange the points emphasizing not only uniqueness, but also longevity, achieving the ability to form measurably more specific topics.

Concretely, neither *cavalry, al-sadr, yiddish* nor *police, business, court* are full representations of New York Times articles, but the latter is a much better basis than the former due to its greater generality. We see the effect of this difference in the specificity of the resulting topics (for example in 17 *obama* topics). Most words in the vocabulary have little connection to the word *cavalry*, so the probability $p(z|w)$ does not change much across different w. When we convert these distributions into $P(w|z)$ using the Bayes' rule, the resulting topics are very close to the corpus distribution, a

Type	#	HR	Top Words (**Yelp**)
t-SNE	16	0	**mexican** good service great eat restaurant authentic delicious
PCA	15	0	**mexican** authentic eat chinese don't restaurant fast salsa
Greedy	34	35	good great food place service restaurant it's mexican
t-SNE	6	0	**beer** selection good pizza great wings tap nice
PCA	39	6	wine beer selection nice list glass **wines** bar
Greedy	99	11	beer selection great happy place wine good bar
t-SNE	3	0	**prices** great good service selection price nice quality
PCA	12	0	**atmosphere** prices drinks friendly selection nice beer ambiance
Greedy	34	35	good great food place service restaurant it's mexican
t-SNE	10	0	**chicken** salad good lunch sauce ordered fried soup
PCA	10	0	**chicken** salad lunch fried pita time back sauce
Greedy	69	12	chicken rice sauce fried ordered i'm spicy soup
Type	#	HR	Top Words (**Blog**)
t-SNE	10	0	**hillary** clinton campaign democratic bill party win race
PCA	4	0	**hillary** clinton campaign democratic party bill democrats vote
Greedy	45	19	obama hillary campaign clinton obama's barack it's democratic
t-SNE	3	0	**iraq** war troops iraqi mccain surge security american
PCA	9	1	iraq **iraqi** war troops military forces security american
Greedy	91	8	iraq mccain war bush troops withdrawal obama iraqi
t-SNE	9	0	**allah** muhammad qur verses unbelievers ibn muslims world
PCA	18	14	allah muhammad qur verses unbelievers story time update
Greedy	4	5	allah muhammad people qur verses **unbelievers** ibn sura
t-SNE	19	0	**catholic** abortion catholics life hagee time biden human
PCA	2	0	**people** it's time don't good make years palin
Greedy	40	1	abortion **parenthood** planned people time state life government
Type	#	HR	Top Words (**NYT**)
t-SNE	0	0	**police** man yesterday officers shot officer year-old charged
PCA	6	0	**people** it's police way those three back don't
Greedy	50	198	police man yesterday officers officer people street city
t-SNE	19	0	**senator** republican senate democratic democrat state bill
PCA	33	2	state republican **republicans** senate senator house bill party
Greedy	85	33	senator republican president state campaign presidential people
t-SNE	2	0	**business** chief companies executive group yesterday billion
PCA	21	0	**billion** companies business deal group chief states united
Greedy	55	10	radio business companies percent day music article satellite
t-SNE	14	0	**market** sales stock companies prices billion investors price
PCA	11	0	**percent** market rate week state those increase high
Greedy	77	44	companies percent billion million group business chrysler people

Table 3: Example t-SNE topics and their most similar topics across algorithms. The Greedy algorithm can find similar topics, but the anchor words are much less salient.

[6] In order to efficiently find an exact convex hull, we use the *Quickhull* algorithm.

unigram distribution $p(w)$.

$$p(w|z = k_{cavalry}) \propto p(z = k_{cavalry}|w)p(w)$$
$$\approx p(w)$$

This lack of specificity results in the observed similarity of topics.

4.2 Quantitative Results

In this section we compare PCA and t-SNE projections to the greedy algorithm along several quantitative metrics. To show the effect of different values of K, we report results for varying numbers of topics. As the anchor finding algorithms are deterministic, the anchor words in a K-dimensional model are identical to the first K anchor words in a $(K+1)$-dimensional model. For the greedy algorithm we select anchor words in the order they are chosen. For the PCA and t-SNE methods, which find anchors jointly, we sort words in descending order by their distance from their centroid.

Recovery Error. Each non-anchor word is approximated by a convex combination of the K anchor words. The projected gradient algorithm (Arora et al., 2013) determines these convex coefficients so that the gap between the original word vector and the approximation becomes minimized. As choosing a good basis of K anchor words decreases this gap, Recovery Error (RE) is defined by the average ℓ_2-residuals across all words.

$$RE = \frac{1}{V}\sum_{i=1}^{V} \|\bar{Q}_i - \sum_{k=1}^{K} p(z_1 = k|w_1 = i)\bar{Q}_{S_k}\|_2 \quad (5)$$

Recovery error decreases with the number of topics, and improves substantially after the first 10–15 anchor words for all methods. The t-SNE method has slightly better performance than the greedy algorithm, but they are similar. Results for recovery with the original, unprojected matrix (not shown) are much worse than the other algorithms, suggesting that the initial anchor words chosen are especially likely to be uninformative.

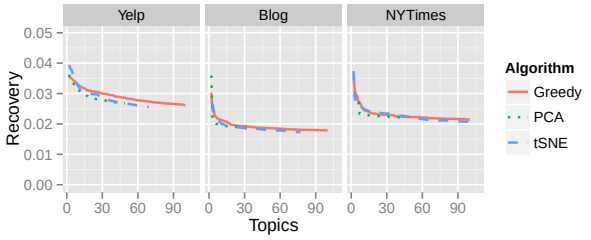


Figure 4: Recovery error is similar across algorithms.

Normalized Entropy. As shown previously, if the probability of topics given a word is close to uniform, the probability of that word in topics will be close to the corpus distribution. Normalized Entropy (NE) measures the entropy of this distribution relative to the entropy of a K-dimensional uniform distribution:

$$NE = \frac{1}{V}\sum_{i=1}^{V} \frac{H(z|w = i)}{\log K}. \quad (6)$$

The normalized entropy of topics given word distributions usually decreases as we add more topics, although both t-SNE and PCA show a dip in entropy for low numbers of topics. This result indicates that words become more closely associated with particular topics as we increase the number of topics. The low-dimensional embedding methods (t-SNE and PCA) have consistently lower entropy.

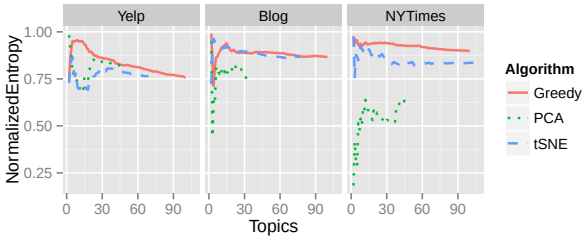


Figure 5: Words have higher topic entropy in the greedy model, especially in NYT, resulting in less specific topics.

Topic Specificity and **Topic Dissimilarity.** We want topics to be both specific (that is, not overly general) and different from each other. When there are insufficient number of topics, $p(w|z)$ often resembles the corpus distribution $p(w)$, where high frequency terms become the top words contributing to most topics. Topic Specificity (TS) is defined by the average KL divergence from each topic's conditional distribution to the corpus distribution.[7]

$$TS = \frac{1}{K}\sum_{k=1}^{K} KL\big(p(w|z = k) \;||\; p(w)\big) \quad (7)$$

[7]We prefer *specificity* to (AlSumait et al., 2009)'s term *vacuousness* because the metric increases as we move away from the corpus distribution.

One way to define the distance between multiple points is the minimum radius of a ball that covers every point. Whereas this is simply the distance form the centroid to the farthest point in the Euclidean space, it is an itself difficult optimization problem to find such centroid of distributions under metrics such as KL-divergence and Jensen-Shannon divergence. To avoid this problem, we measure Topic Dissimilarity (TD) viewing each conditional distribution $p(w|z)$ as a simple V-dimensional vector in $\mathbb{R}^V$. Recall $a_{ik} = p(w = i|z = k)$,

$$TD = \max_{1 \le k \le K} \| \frac{1}{K} \sum_{k'=1}^{K} a_{*k'} - a_{*k} \|_2. \quad (8)$$

Specificity and dissimilarity increase with the

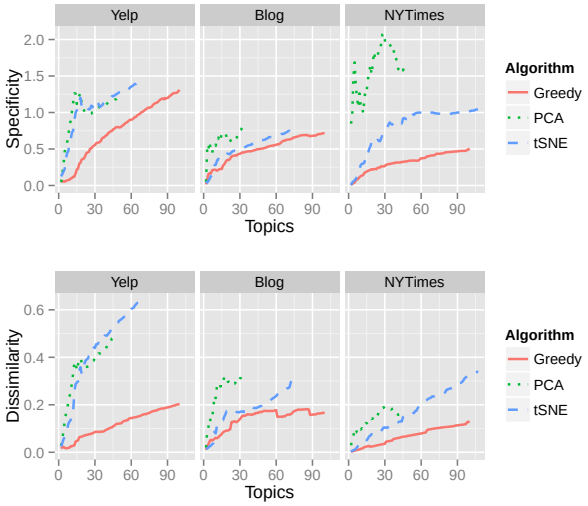


Figure 6: Greedy topics look more like the corpus distribution and more like each other.

number of topics, suggesting that with few anchor words, the topic distributions are close to the overall corpus distribution and very similar to one another. The t-SNE and PCA algorithms produce consistently better specificity and dissimilarity, indicating that they produce more useful topics early with small numbers of topics. The greedy algorithm produces topics that are closer to the corpus distribution and less distinct from each other (17 *obama* topics).

Topic Coherence is known to correlate with the semantic quality of topic judged by human annotators (Mimno et al., 2011). Let $\mathcal{W}_k^{(T)}$ be T most probable words (i.e., top words) for the topic k.

$$TC = \sum_{w_1 \neq w_2 \in \mathcal{W}_k^{(T)}} \log \frac{D(w_1, w_2) + \epsilon}{D(w_1)} \quad (9)$$

Here $D(w_1, w_2)$ is the co-document frequency, which is the number of documents in $\mathcal{D}$ consisting of two words w_1 and w_2 simultaneously. $D(w)$ is the simple document frequency with the word w. The numerator contains smoothing count ϵ in order to avoid taking the logarithm of zero. Coherence scores for t-SNE and PCA are worse

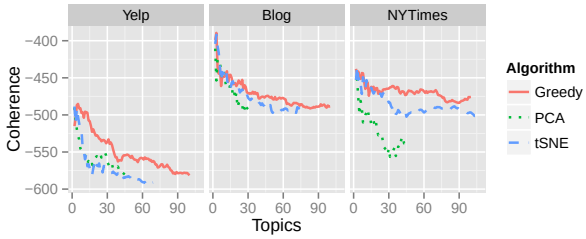


Figure 7: The greedy algorithm creates more coherent topics (higher is better), but at the cost of many overly general or repetitive topics.

than those for the greedy method, but this result must be understood in combination with the Specificity and Dissimilarity metrics. The most frequent terms in the overall corpus distribution $p(w)$ often appear together in documents. Thus a model creating many topics similar to the corpus distribution is likely to achieve high Coherence, but low Specificity by definition.

Saliency. (Chuang et al., 2012) define saliency for topic words as a combination of distinctiveness and probability within a topic. Anchor words are distinctive by construction, so we can increase saliency by selecting more probable anchor words. We measure the probability of anchor words in two ways. First, we report the zero-based rank of anchor words within their topics. Examples of this metric, which we call "hard" rank are shown in Table 3. The hard rank of the anchors in the PCA and t-SNE models are close to zero, while the anchor words for the greedy algorithm are much lower ranked, well below the range usually displayed to users. Second, while hard rank measures the perceived difference in rank of contributing words, position may not fully capture the relative importance of the anchor word. "Soft" rank quantifies the average log ratio between probabilities of the

prominent word w_k^* and the anchor word s_k.

$$SR = \frac{1}{K}\sum_{k=1}^{K} \log \frac{p(w = w_k^* | z = k)}{p(w = s_k | z = k)} \quad (10)$$

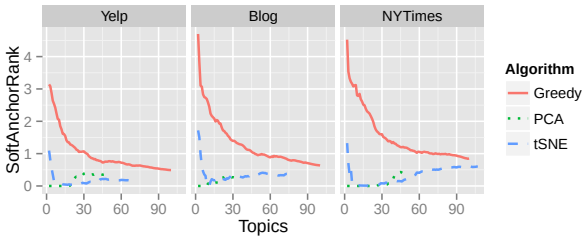


Figure 8: Anchor words have higher probability, and therefore greater salience, in t-SNE and PCA models (1 $\approx$ one third the probability of the top ranked word).

Lower values of soft rank (Fig. 8 indicate that the anchor word has greater relative probability to occur within a topic. As we increase the number of topics, anchor words become more prominent in topics learned by the greedy method, but t-SNE anchor words remain relatively more probable within their topics as measured by soft rank.

Held-out Probability. Given an estimate of the topic-word matrix A, we can compute the marginal probability of held-out documents under that model. We use the left-to-right estimator introduced by (Wallach et al., 2009), which uses a sequential algorithm similar to a Gibbs sampler. This method requires a smoothing parameter for document-topic Dirichlet distributions, which we set to $\alpha_k = 0.1$. We note that the greedy algorithm run on the original, unprojected matrix has better held-out probability values than t-SNE for the Yelp corpus, but as this method does not scale to realistic vocabularies we compare here to the sparse random projection method used in (Arora et al., 2013). The t-SNE method appears to do best, particularly in the NYT corpus, which has a larger vocabulary and longer training documents. The length of individual held-out documents has no correlation with held-out probability.

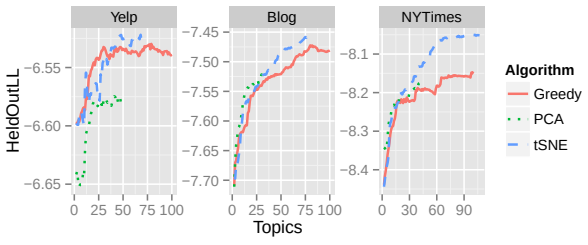


Figure 9: t-SNE topics have better held-out probability than greedy topics.

We emphasize that Held-out Probability is sensitive to smoothing parameters and should only be used in combination with a range of other topic-quality metrics. In initial experiments, we observed significantly worse held-out performance for the t-SNE algorithm. This phenomenon was because setting the probability of anchor words to zero for all but their own topics led to large negative values in held-out log probability for those words. As t-SNE tends to choose more frequent terms as anchor words, these "spikes" significantly affected overall probability estimates. To make the calculation more fair, we added 10^{-5} to any zero entries for anchor words in the topic-word matrix A across *all* models and renormalized.

Because t-SNE is a stochastic model, different initializations can result in different embeddings. To evaluate how steady anchor word selection is, we ran five random initializations for each dataset. For the Yelp dataset, the number of anchor words varies from 59 to 69, and 43 out of those are shared across at least four trials. For the Blog dataset, the number of anchor words varies from 80 to 95, with 56 shared across at least four trials. For the NYT dataset, this number varies between 83 and 107, with 51 shared across at least four models.

4.3 Qualitative Results

Table 3 shows topics trained by three methods (t-SNE, PCA, and greedy) for all three datasets. For each model, we select five topics *at random* from the t-SNE model, and then find the closest topic from each of the other models. If anchor words present in the top eight words, they are shown in boldface.

A fundamental difference between anchor-based inference and traditional likelihood-based inference is that we can give an *order* to topics according to their contribution to word co-occurrence convex hull. This order is intrinsic to the original algorithm, and we heuristically give orders to t-SNE and PCA based on their contributions. This order is listed as # in the previous table. For all but one topic, the closest topic from the greedy model has a higher order number than

the associated t-SNE topic. As shown above, the standard algorithm tends to pick less useful anchor words at the initial stage; only the later, higher ordered topics are specific.

The most clear distinction between models is the rank of anchor words represented by Hard Rank for each topic. Only one topic corresponding to (*initial*) has the anchor word which does not coincide with the top-ranked word. For the greedy algorithm, anchor words are often tens of words down the list in rank, indicating that they are unlikely to find a connection to the topic's semantic core. In cases where the anchor word is highly ranked (*unbelievers*, *parenthood*) the word is a good indicator of the topic, but still less decisive.

t-SNE and PCA are often consistent in their selection of anchor words, which provides useful validation that low-dimensional embeddings discern more relevant anchor words regardless of linear vs non-linear projections. Note that we are only varying the anchor selection part of the Anchor Words algorithm in these experiments, recovering topic-word distributions in the same manner given anchor words. As a result, any differences between topics with the same anchor word (for example *chicken*) are due to the difference in either the number of topics or the rest of anchor words. Since PCA suffers from a crowding problem in lower-dimensional projection (see Figure 2) and the problem could be severe in a dataset with a large vocabulary, t-SNE is more likely to find the proper number of anchors given a specified granularity.

5 Conclusion

One of the main advantages of the anchor words algorithm is that the running time is largely independent of corpus size. Adding more documents would not affect the size of the co-occurrence matrix, requiring more times to construct the co-occurrence matrix at the beginning. While the inference is scalable depending only on the size of the vocabulary, finding quality anchor words is crucial for the performance of the inference.

(Arora et al., 2013) presents a greedy anchor finding algorithm that improves over previous linear programming methods, but finding quality anchor words remains an open problem in spectral topic inference. We have shown that previous approaches have several limitations. Exhaustively finding anchor words by eliminating words that are reproducible by other words (Arora et al., 2012) is impractical. The anchor words selected by the greedy algorithm are overly eccentric, particularly at the early stages of the algorithm, causing topics to be poorly differentiated. We find that using low-dimensional embeddings of word co-occurrence statistics allows us to approximate a better convex hull. The resulting anchor words are highly *salient*, being both distinctive and probable. The models trained with these words have better quantitative and qualitative properties along various metrics. Most importantly, using radically low-dimensional projections allows us to provide users with clear visual explanations for the model's anchor word selections.

An interesting property of using low-dimensional embeddings is that the number of topics depends only on the projecting dimension. Since we can efficiently find an exact convex hull in low-dimensional space, users can achieve topics with their preferred level of granularities by changing the projection dimension. We do not insist this is the "correct" number of topics for a corpus, but this method, along with the range of metrics described in this paper, provides users with additional perspective when choosing a dimensionality that is appropriate for their needs.

We find that the t-SNE method, besides its well-known ability to produce high quality layouts, provides the best overall anchor selection performance. This method consistently selects higher-frequency terms as anchor words, resulting in greater clarity and interpretability. Embeddings with PCA are also effective, but they result in less well-formed spaces, being less effective in held-out probability for sufficiently large corpora.

Anchor word finding methods based on low-dimensional projections offer several important advantages for topic model users. In addition to producing more salient anchor words that can be used effectively as topic labels, the relationship of anchor words to a visualizable word co-occurrence space offers significant potential. Users who can see why the algorithm chose a particular model will have greater confidence in the model and in any findings that result from topic-based analysis. Finally, visualizable spaces offer the potential to produce interactive environments for semi-supervised topic reconstruction.

Acknowledgments

We thank David Bindel and the anonymous reviewers for their valuable comments and suggestions, and Laurens van der Maaten for providing his t-SNE implementation.

References

Dimitris Achlioptas. 2001. Database-friendly random projections. In *SIGMOD*, pages 274–281.

Loulwah AlSumait, Daniel Barbar, James Gentle, and Carlotta Domeniconi. 2009. Topic significance ranking of lda generative models. In *ECML*.

S. Arora, R. Ge, and A. Moitra. 2012. Learning topic models – going beyond svd. In *FOCS*.

Sanjeev Arora, Rong Ge, Yonatan Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. 2013. A practical algorithm for topic modeling with provable guarantees. In *ICML*.

D. Blei, A. Ng, and M. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, pages 993–1022. Preliminary version in *NIPS* 2001.

Christos Boutsidis, Michael W. Mahoney, and Petros Drineas. 2009. An improved approximation algorithm for the column subset selection problem. In *SODA*, pages 968–977.

Jason Chuang, Christopher D. Manning, and Jeffrey Heer. 2012. Termite: Visualization techniques for assessing textual topic models. In *International Working Conference on Advanced Visual Interfaces (AVI)*, pages 74–77.

Jacob Eisenstein and Eric Xing. 2010. The CMU 2008 political blog corpus. Technical report, CMU, March.

Jacob Eisenstein, Duen Horng Chau, Aniket Kittur, and Eric P. Xing. 2011. Topicviz: Semantic navigation of document collections. *CoRR*, abs/1110.6200.

T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235.

Ming Gu and Stanley C. Eisenstat. 1996. Efficient algorithms for computing a strong rank-revealing qr factorization. In *SIAM J. Sci Comput*, pages 848–869.

William B. Johnson and Joram Lindenstrauss. 1984. Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, 26:189–206.

Simon Lacoste-Julien, Fei Sha, and Michael I. Jordan. 2008. DiscLDA: Discriminative learning for dimensionality reduction and classification. In *NIPS*.

Jey Han Lau, Karl Grieser, David Newman, and Timothy Baldwin. 2011. Automatic labelling of topic models. In *HLT*, pages 1536–1545.

Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. 2007. Automatic labeling of multinomial topic models. In *KDD*, pages 490–499.

David Mimno, Hanna Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *EMNLP*.

D. Sontag and D. Roy. 2011. Complexity of inference in latent dirichlet allocation. In *NIPS*, pages 1008–1016.

L.J.P. van der Maaten and G.E. Hinton. 2008. Visualizing high-dimensional data using t-SNE. *JMLR*, 9:2579–2605, Nov.

Hanna Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009. Evaluation methods for topic models. In *ICML*.

Jun Zhu, Amr Ahmed, and Eric P. Xing. 2009. MedLDA: Maximum margin supervised topic models for regression and classication. In *ICML*.

Weakly-Supervised Learning with Cost-Augmented Contrastive Estimation

Kevin Gimpel **Mohit Bansal**
Toyota Technological Institute at Chicago, IL 60637, USA
{kgimpel,mbansal}@ttic.edu

Abstract

We generalize contrastive estimation in two ways that permit adding more knowledge to unsupervised learning. The first allows the modeler to specify not only the *set* of corrupted inputs for each observation, but also *how bad* each one is. The second allows specifying structural preferences on the latent variable used to explain the observations. They require setting additional hyperparameters, which can be problematic in unsupervised learning, so we investigate new methods for unsupervised model selection and system combination. We instantiate these ideas for part-of-speech induction without tag dictionaries, improving over contrastive estimation as well as strong benchmarks from the PASCAL 2012 shared task.

1 Introduction

Unsupervised NLP aims to discover useful structure in unannotated text. This structure might be part-of-speech (POS) tag sequences (Merialdo, 1994), morphological segmentation (Creutz and Lagus, 2005), or syntactic structure (Klein and Manning, 2004), among others. Unsupervised systems typically improve when researchers incorporate knowledge to bias learning to capture characteristics of the desired structure.[1]

There are many successful examples of adding knowledge to improve learning without labeled examples, including: sparsity in POS tag distributions (Johnson, 2007; Ravi and Knight, 2009; Ganchev et al., 2010), short attachments for dependency parsing (Smith and Eisner, 2006), agreement of word alignment models (Liang et al., 2006), power law effects in lexical distributions (Blunsom and Cohn, 2010; Blunsom and Cohn, 2011), multilingual constraints (Smith and Eisner, 2009; Ganchev et al., 2009; Snyder et al., 2009; Das and Petrov, 2011), and orthographic cues (Spitkovsky et al., 2010c; Spitkovsky et al., 2011b), *inter alia*.

Contrastive estimation (CE; Smith and Eisner, 2005) is a general approach to weakly-supervised learning with a particular way of incorporating knowledge. CE increases the likelihood of the observations at the expense of those in a particular **neighborhood** of each observation. The neighborhood typically contains corrupted versions of the observations. The latent structure is marginalized out for both the observations and their corruptions; the intent is to learn latent structure that helps to explain why the observation was generated rather than any of the corrupted alternatives.

In this paper, we present a new objective function for weakly-supervised learning that generalizes CE by including two types of **cost functions**, one on observations and one on output structures. The first (§4) allows us to specify not only the set of corrupted observations, but also *how bad* each corruption was. We use n-gram language models to measure the severity of each corruption.

The second (§5) allows us to specify preferences on desired output structures, regardless of the input sentence. For POS tagging, we attempt to learn language-independent tag frequencies by computing counts from treebanks for 11 languages not used in our POS induction experiments. For example, we encourage tag sequences that contain adjacent nouns and penalize those that contain adjacent adpositions.

We consider several unsupervised ways to set hyperparameters for these cost functions (§7), including the recently-proposed log-likelihood estimator of Bengio et al. (2013). We also circumvent

[1] We note that doing so strains the definition of the term *unsupervised*. Hence we will use the term *weakly-supervised* to refer to methods that do not explicitly train on labeled examples for the task of interest, but do use some form of task-specific knowledge.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1329–1341,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

hyperparameter selection via system combination, developing a novel voting scheme for POS induction that aligns tag identifiers across runs.

We evaluate our approach, which we call **cost-augmented contrastive estimation (CCE)**, on POS induction without tag dictionaries for five languages from the PASCAL shared task (Gelling et al., 2012). We find that CCE improves over both standard CE as well as strong baselines from the shared task. In particular, our final average accuracies are better than all entries in the shared task that use the same number of tags.

2 Related Work

Weakly-supervised techniques can be roughly categorized in terms of whether they influence the model, the learning procedure, or explicitly target the output structure. Examples abound in NLP; we focus on those that have been applied to POS tagging.

There have been many efforts at biasing models, including features (Smith and Eisner, 2005a; Berg-Kirkpatrick et al., 2010), sparse priors (Johnson, 2007; Goldwater and Griffiths, 2007; Toutanova and Johnson, 2007), sparsity in tag transition distributions (Ravi and Knight, 2009), small models via minimum description length criteria (Vaswani et al., 2010; Poon et al., 2009), a one-tag-per-type constraint (Blunsom and Cohn, 2011), and power law effects via Bayesian nonparametrics (Van Gael et al., 2009; Blunsom and Cohn, 2010; Blunsom and Cohn, 2011).

We focus below on efforts that induce bias into the learning (§2.1) or more directly in the output structure (§2.2), as they are more closely related to our contributions in this paper.

2.1 Biasing Learning

Some unsupervised methods do not change the model or attempt to impose structural bias; rather, they change the learning. This may involve optimizing a different objective function for the same model, e.g., by switching from soft to hard EM (Spitkovsky et al., 2010b). Or it may involve changing the objective during learning via annealing (Smith and Eisner, 2004) or more general multi-objective techniques (Spitkovsky et al., 2011a; Spitkovsky et al., 2013).

Other learning modifications relate to automatic data selection, e.g., choosing examples for generative learning (Spitkovsky et al., 2010a) or automatically generating negative examples for discriminative unsupervised learning (Li et al., 2010; Xiao et al., 2011).

CE does both, automatically generating negative examples and changing the objective function to include them. Our observation cost function alters CE's objective function, sharpening the effective distribution of the negative examples.

2.2 Structural Bias

Our output cost function is used to directly specify preferences on desired output structures. Several others have had similar aims. For dependency grammar induction, Smith and Eisner (2006) favored short attachments using a fixed-weight feature whose weight was optionally annealed during learning. Their bias could be implemented as an output cost function in our framework.

Posterior regularization (PR; Ganchev et al., 2010) is a general framework for declaratively specifying preferences on model outputs. Naseem et al. (2010) proposed universal syntactic rules for unsupervised dependency parsing and used them in a PR regime; we use analogous universal tag sequences in our cost function.

Our output cost is similar to posterior regularization. The difference is that we specify preferences via an arbitrary cost function on output structures, while PR uses expectation constraints on posteriors of the model. We compare to the PR tag induction system of Graça et al. (2011) in our experiments, improving over it in several settings.

2.3 Exploiting Resources

Much of the work mentioned above also benefits from leveraging existing resources. These may be curated or crowdsourced resources like the Wiktionary (Li et al., 2012), or traditional annotated treebanks for languages other than those under investigation (Cohen et al., 2011). In this paper, we use tag statistics from treebanks for 11 languages to impose our structural bias for a different set of languages used in our POS induction experiments.

Substantial recent work has improved many NLP tasks by leveraging multilingual or parallel text (Cohen and Smith, 2009; Snyder et al., 2009; Wang and Manning, 2014), including unsupervised POS tagging (Naseem et al., 2009; Das and Petrov, 2011; Täckström et al., 2013; Ganchev and Das, 2013). This sort of multilingual guidance could also be captured by particular output cost functions, though we leave this to future work.

3 Unsupervised Structure Learning

We consider a structured unsupervised learning setting. We use $\mathcal{X}$ to denote our set of possible structured inputs, and for a particular $\boldsymbol{x} \in \mathcal{X}$, we use $\mathcal{Y}(\boldsymbol{x})$ to denote the set of valid structured outputs for $\boldsymbol{x}$. We are given a dataset of inputs $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$. To map inputs to outputs, we start by building a model of the joint probability distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})$. We use a log-linear parameterization with feature vector $\boldsymbol{f}$ and weight vector $\boldsymbol{\theta}$:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})\right\}}{\sum_{\boldsymbol{x}' \in \mathcal{X}, \boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{y}')\right\}}$$

where the sum in the denominator ranges over all possible inputs and all valid outputs for them.

In this paper, we consider ways of learning the parameters $\boldsymbol{\theta}$. Given $\boldsymbol{\theta}$, at test time we output a $\boldsymbol{y}$ for a new $\boldsymbol{x}$ using, e.g., Viterbi or minimum Bayes risk decoding; we use the latter in this paper.

3.1 EM and Contrastive Estimation

We start by reviewing two ways of choosing $\boldsymbol{\theta}$. The expectation-maximization algorithm (EM; Dempster et al., 1977) finds a local optimum of the marginal (log-)likelihood of the observations $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$. The marginal log-likelihood is a sum over all $\boldsymbol{x}^{(i)}$ of the **gain function** $\gamma_{\text{EM}}(\boldsymbol{x}^{(i)})$:

$$\begin{aligned}
\gamma_{\text{EM}}(\boldsymbol{x}^{(i)}) &= \log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, \boldsymbol{y}) \\
&= \log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\right\} \\
&\quad - \underbrace{\log \sum_{\boldsymbol{x}' \in \mathcal{X}, \boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{y}')\right\}}_{Z(\boldsymbol{\theta})}
\end{aligned}$$

The difficulty is the final term, $\log Z(\boldsymbol{\theta})$, which requires summing over all possible inputs and all valid outputs for them. This summation is typically intractable for structured problems, and may even diverge. For this reason, EM is typically only used to train log-linear model weights when $Z(\boldsymbol{\theta}) = 1$, e.g., for hidden Markov models, probabilistic context-free grammars, and models composed of locally-normalized log-linear models (Berg-Kirkpatrick et al., 2010), among others.

There have been efforts at approximating the summation over elements of $\mathcal{X}$, whether by limiting sequence length (Haghighi and Klein, 2006), only summing over observations in the training data (Riezler, 1999), restricting the observation space based on the task (Dyer et al., 2011), or using Gibbs sampling to obtain an unbiased sample of the full space (Della Pietra et al., 1997; Rosenfeld, 1997).

Contrastive estimation (CE) addresses this challenge by using a **neighborhood** function $\mathcal{N} : \mathcal{X} \rightarrow 2^{\mathcal{X}}$ that generates a set of inputs that are "corruptions" of an input $\boldsymbol{x}$; $\mathcal{N}(\boldsymbol{x})$ always includes $\boldsymbol{x}$. Using shorthand $\mathcal{N}_i$ for $\mathcal{N}(\boldsymbol{x}^{(i)})$, CE corresponds to maximizing the sum over inputs $\boldsymbol{x}^{(i)}$ of the gain

$$\begin{aligned}
\gamma_{\text{CE}}(\boldsymbol{x}^{(i)}) &= \log \Pr(\boldsymbol{x}^{(i)} \mid \mathcal{N}_i) \\
&= \log \frac{\sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, \boldsymbol{y})}{\sum_{\boldsymbol{x}' \in \mathcal{N}_i} \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} p_{\boldsymbol{\theta}}(\boldsymbol{x}', \boldsymbol{y}')} \\
&= \log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\right\} - \\
&\quad \log \sum_{\boldsymbol{x}' \in \mathcal{N}_i} \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{y}')\right\}
\end{aligned}$$

Two $\log Z(\boldsymbol{\theta})$ terms cancel out, leaving the summation over input/output pairs in the neighborhood instead of the full summation over pairs.

Two desiderata govern the choice of $\mathcal{N}$. One is to make the summation over its elements computationally tractable. If $\mathcal{N}(\boldsymbol{x}) = \mathcal{X}$ for all $\boldsymbol{x} \in \mathcal{X}$, we obtain EM, so a smaller neighborhood typically must be used in practice. The second consideration is to target learning for the task of interest. For POS tagging and dependency parsing, Smith and Eisner (2005a, 2005b) used neighborhood functions that corrupted the observations in systematic ways, e.g., their TRANS1 neighborhood contains the original sentence along with those that result from transposing a single pair of adjacent words. The intent was to force the learner to explain why the given sentences were observed at the expense of the corrupted sentences.

Next we present our modifications to contrastive estimation. Both can be viewed as adding specialized cost functions that penalize some part of the structured input/output pair.

4 Modeling Corruption Costs

While CE allows us to specify a set of corrupted $\boldsymbol{x}$ for each $\boldsymbol{x}^{(i)}$ via the neighborhood function $\mathcal{N}$, it says nothing about how bad each corruption is. The same type of corruption might be harmful in one context and not harmful in another.

This fact was suggested as the reason why certain neighborhoods did not work as well for POS

tagging as others (Smith and Eisner, 2005a). One poorly-performing neighborhood consisted of sentences in which a single word of the original was deleted. Deleting a single word in a sentence might not harm grammaticality. By contrast, neighborhoods that transpose adjacent words led to better results. These kinds of corruptions are expected to be more frequently harmful, at least for languages with relatively rigid word order. However, there may still be certain transpositions that are benign, at least for grammaticality.

To address this, we introduce an **observation cost function** $\Delta : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_{\geq 0}$ that indicates how much two observations differ. Using Δ, we define the following gain function $\gamma_{\mathrm{CCE}_1}(\boldsymbol{x}^{(i)}) =$

$$\log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\right\} -$$
$$\log \sum_{\boldsymbol{x}' \in \mathcal{N}_i} \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{y}') + \Delta(\boldsymbol{x}^{(i)}, \boldsymbol{x}')\right\}$$

The function Δ inflates the score of neighborhood entries with larger differences from the observed $\boldsymbol{x}^{(i)}$. This gain function is inspired by ideas from structured large-margin learning (Taskar et al., 2003; Tsochantaridis et al., 2005), specifically softmax-margin (Povey et al., 2008; Gimpel and Smith, 2010). Softmax-margin extends conditional likelihood by allowing the user to specify a cost function to give partial credit for structures that are partially correct. Conditional likelihood, by contrast, treats all incorrect structures equally.

While softmax-margin uses a cost function to specify how two *output* structures differ, our gain function γ_{CCE_1} uses a cost function Δ to specify how two *inputs* differ. But the motivations are similar: since poor structures have their scores artificially inflated by Δ, learning pays more attention to them, choosing weights that penalize them more than the lower-cost structures.

4.1 Observation Cost Functions

What types of cost functions should we consider? For efficient inference, we want to ensure that Δ decomposes additively across parts of the corrupted input $\boldsymbol{x}'$ in the same way as the features; we assume unigram and bigram features in this paper.

In addition, the choice of the observation cost function Δ is tied to the choice of neighborhood function. In our experiments, we use neighborhoods that change the *order* of words in the observation but not the *set* of words. Our first cost function simply counts the number of novel bigrams introduced when corrupting the original:

$$\Delta_I(\boldsymbol{x}^{(i)}, \boldsymbol{x}) = \alpha \sum_{j=1}^{|\boldsymbol{x}|+1} \mathbb{I}\Big[x_{j-1}x_j \notin 2\mathrm{grams}(\boldsymbol{x}^{(i)})\Big]$$

where x_j is the jth word of sentence $\boldsymbol{x}$, x_0 is the start-of-sentence marker, $x_{|\boldsymbol{x}|+1}$ is the end-of-sentence marker, $2\mathrm{grams}(\boldsymbol{x})$ returns the set of bigrams in $\boldsymbol{x}$, $\mathbb{I}[]$ returns 1 if its argument is true and 0 otherwise, and α is a constant to be tuned. We call this cost function MATCH. Only $\boldsymbol{x}^{(i)}$ (which is always contained in $\mathcal{N}_i$) is guaranteed to have cost 0. In the TRANS1 neighborhood, corrupted sequences will be penalized more if their transpositions occur in the middle of the sentence rather than at the beginning or end.

We also consider a version that weights the indicator by the negative log probability of the novel bigram: $\Delta_{LM}(\boldsymbol{x}^{(i)}, \boldsymbol{x}) =$

$$\alpha \sum_{j=1}^{|\boldsymbol{x}|+1} -\log \mathrm{P}(x_j | x_{j-1}) \mathbb{I}\Big[x_{j-1}x_j \notin 2\mathrm{grams}(\boldsymbol{x}^{(i)})\Big]$$

where $\mathrm{P}(x_j|x_{j-1})$ is obtained from a bigram language model. Among novel bigrams in the corruption $\boldsymbol{x}$, if the second word is highly surprising conditioned on the first, the bigram will incur high cost. We refer to $\Delta_{LM}(\boldsymbol{x}^{(i)}, \boldsymbol{x})$ as MATLM.

5 Expressing Structural Preferences

Our second modification to CE allows us to specify structural preferences for outputs $\boldsymbol{y}$. We first note that there exist objective functions for *supervised* structure prediction that never require computing the feature vector for the true output $\boldsymbol{y}^{(i)}$. Examples include Bayes risk (Kaiser et al., 2000; Povey and Woodland, 2002) and structured ramp loss (Do et al., 2008). These two objectives do, however, need to compute a cost function $\mathrm{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})$, which requires the true output $\boldsymbol{y}^{(i)}$. We start with the following form of structured ramp loss from Gimpel and Smith (2012), transformed here to a gain function:

$$\max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \Big(\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}) - \mathrm{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\Big) -$$
$$\max_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \Big(\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}') + \mathrm{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y}')\Big) \quad (1)$$

Maximizing this gain function for supervised learning corresponds to increasing the model score

of outputs that have both high model score ($\boldsymbol{\theta}^\top \boldsymbol{f}$) and low cost, while decreasing the model score of outputs with high model score and *high* cost.

For unsupervised learning, we do not have $\boldsymbol{y}^{(i)}$, so we simply drop $\boldsymbol{y}^{(i)}$ from the cost function. The result is an **output cost function** $\pi : \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$ which captures our *a priori* knowledge about desired output structures. The value of $\pi(\boldsymbol{y})$ should be large for outputs $\boldsymbol{y}$ that are far from the ideal. In this paper, we consider POS induction and use intrinsic evaluation; however, in a real-world scenario, the output cost function could use signals derived from the downstream task in which the tags are being used.

Given π, we convert each max to a $\log \sum \exp$ in Eq. 1 and introduce the contrastive neighborhood into the second term, defining our new gain function $\gamma_{\mathrm{CCE}_2}(\boldsymbol{x}^{(i)}) =$

$$\log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}) - \pi(\boldsymbol{y})\right\} -$$
$$\log \sum_{\boldsymbol{x}' \in \mathcal{N}_i} \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{y}') + \pi(\boldsymbol{y}')\right\}$$

Gimpel (2012) found that using such "softened" versions of the ramp losses worked better than the original versions (e.g., Eq. 1) when training machine translation systems.

5.1 Output Cost Functions

The output cost π should capture our desiderata about $\boldsymbol{y}$ for the task of interest. We consider universal POS tag subsequences analogous to the universal syntactic rules of Naseem et al. (2010). In doing so, we use the universal tags of Petrov et al. (2012): NOUN, VERB, ADJ (adjective), ADV (adverb), PRON (pronoun), DET (determiner), ADP (pre/postposition), NUM (numeral), CONJ (conjunction), PRT (particle), '.' (punctuation), and X (other).

We aimed for a set of rules that would be robust across languages. So, we used treebanks for 11 languages from the CoNLL 2006/2007 shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007) other than those used in our POS induction experiments. In particular, we used Arabic, Bulgarian, Catalan, Czech, English, Spanish, German, Hungarian, Italian, Japanese, and Turkish. We replicated shorter treebanks a sufficient number of times until they were a similar size as the largest treebank. Then we counted gold POS tag unigrams and bigrams from the concatenation.

tag unigram	count	cost
X	50783	3.83
NUM	174613	2.59
PRT	179131	2.57
ADV	330210	1.96
CONJ	436649	1.68
PRON	461880	1.62
DET	615284	1.33
ADJ	694685	1.21
ADP	906922	0.95
VERB	1018989	0.83
.	1042662	0.81
NOUN	2337234	0

tag bigram	count	cost
DET PRT	109	84.41
DET CONJ	518	68.82
NUM ADV	1587	57.63
NOUN NOUN	409828	2.09
DET NOUN	454980	1.04
NOUN .	504897	0

Table 1: Counts and costs for universal tags based on treebanks for 11 languages not used in POS induction experiments.

Where $\#(y)$ is the count of tag y in the treebank concatenation, the cost of y is

$$u(y) = \log\left(\frac{\max_{y'} \#(y')}{\#(y)}\right)$$

and, where $\#(\langle y_1, y_2\rangle)$ is the count of tag bigram $\langle y_1, y_2\rangle$, the cost of $\langle y_1, y_2\rangle$ is

$$u(\langle y_1, y_2\rangle) = 10 \times \log\left(\frac{\max_{\langle y_1', y_2'\rangle} \#(\langle y_1', y_2'\rangle)}{\#(\langle y_1, y_2\rangle)}\right)$$

We use a multiplier of 10 in order to exaggerate count differences among bigrams, which generally are closer together than unigram counts. In Table 1, we show counts and costs for all tag unigrams and selected tag bigrams.[2]

Given these costs for individual tag unigrams and bigrams, we use the following π function, which we call UNIV:

$$\pi(\boldsymbol{y}) = \beta \sum_{j=1}^{|\boldsymbol{y}|+1} u(y_j) + u(\langle y_{j-1}, y_j\rangle)$$

where β is a constant to be tuned and y_j is the jth tag of $\boldsymbol{y}$. We define y_0 to be the beginning-of-sentence marker and $y_{|\boldsymbol{y}|+1}$ to be the end-of-sentence marker (which has unigram cost 0).

Many POS induction systems use one-tag-per-type constraints (Blunsom and Cohn, 2011; Gelling et al., 2012), which often lead to higher

[2]The complete tag bigram list is provided in the supplementary material.

$$\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\right\} - \log \sum_{\boldsymbol{x}' \in \mathcal{N}_i} \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{y}')\right\} \quad (2)$$

$$\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}) - \pi(\boldsymbol{y})\right\} - \log \sum_{\boldsymbol{x}' \in \mathcal{N}_i} \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{y}') + \Delta(\boldsymbol{x}^{(i)}, \boldsymbol{x}') + \pi(\boldsymbol{y}')\right\} \quad (3)$$

Figure 1: Contrastive estimation (Eq. 2) and cost-augmented contrastive estimation (Eq. 3). L2 regularization terms ($\frac{C}{2}\sum_{j=1}^{|\boldsymbol{\theta}|}\theta_j^2$) are not shown here but were used in our experiments.

accuracies even though the gold standard is not constrained in this way. This constraint can be encoded as an output cost function, though it would require approximate inference (Poon et al., 2009).

6 Cost-Augmented CE

We extended the objective function underlying CE by defining two new types of cost functions, one on observations (§4) and one on outputs (§5). We combine them into a single objective, which we call **cost-augmented contrastive estimation** (CCE), shown as Eq. 3 in Figure 1.

If the cost functions Δ and π factor in the same way as the features $\boldsymbol{f}$, then it is straightforward to implement CCE atop an existing CE implementation. The additional terms in the cost functions can be implemented as features with fixed weights (albeit where the weight differs depending on the context).

7 Model Selection

Our modifications give increased flexibility, but require setting new hyperparameters. In addition to the choice of the cost functions, each has a weight: α for Δ and β for π. We need ways to set these weights that do not require labeled data.

Smith and Eisner (2005a) chose the hyperparameter values that yielded the best CE objective on held-out development data. We use their strategy, though we experiment with two others as well.[3] In particular, we estimate held-out data log-likelihood via the method of Bengio et al. (2013) and also consider ways of combining outputs from multiple models.

7.1 Estimating Held-Out Log-Likelihood

Bengio et al. (2013) recently proposed ways to efficiently estimate held-out data log-likelihood for generative models. They showed empirically that a simple, biased version of their conservative sampling-based log-likelihood (CSL) estimator can be useful for model selection.

The biased CSL requires a Markov chain on the variables in the model (i.e., $\boldsymbol{x}$ and $\boldsymbol{y}$) as well as the ability to compute $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y})$. It generates consecutive samples of $\boldsymbol{y}$ from a Markov chain initialized at each $\boldsymbol{x}$ in a development set D, with S Markov chains run for each $\boldsymbol{x}$. We compute and sum $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y}^j)$ for each sampled $\boldsymbol{y}^j$, then sum over all $\boldsymbol{x}$ in D. The result is a biased estimate for the log-likelihood of D. Bengio et al. showed that these biased estimates could give the same model ranking as unbiased estimates, though more efficiently. They also showed that taking the single, initial sample from the S Markov chains resulted in the same model ranking as using many samples from each chain. We follow suit here.

Our Markov chain is a blocked Gibbs sampler in which we alternate between sampling from $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y})$. Since we only use a single sample from each Markov chain and initialize each chain to $\boldsymbol{x}$, this simply amounts to drawing S samples from $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$. To sample from $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$, we use the exact algorithm obtained by running the backward algorithm and then performing left-to-right sampling of tags using the local features and requisite backward terms to define the local tag distributions.

We then compute $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y})$ for each sampled $\boldsymbol{y}$. If there are no features in $\boldsymbol{f}$ that look at more than one word (which is the case with the features used in our experiments), then this probability factors:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y}) = \prod_{k=1}^{|\boldsymbol{y}|} p_{\boldsymbol{\theta}}(x_k|y_k)$$

This is easily computable assuming that we have normalization constants $Z(y)$ cached for each tag y. To compute each $Z(y)$, we sum over all words observed in the training data (replacing some with a special UNK token; see below). We can then compute likelihoods for individual words and mul-

[3] When using their strategy for CCE, we compute the CE criterion only, omitting the costs. We do so because the weights of the cost terms can have a large impact on the magnitude of the objective, making it difficult to do a fair comparison of models with different cost weights.

tiply them across the words in the sentence to compute $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y})$.

To summarize, we get a log-likelihood estimate for development set $D = \{\boldsymbol{x}^{(i)}\}_{i=1}^{|D|}$ by sampling S times from $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}^{(i)})$ for each $\boldsymbol{x}^{(i)}$, getting samples $\{\{\boldsymbol{y}^{(i),j}\}_{j=1}^{S}\}_{i=1}^{|D|}$, then we compute

$$\sum_{i=1}^{|D|} \sum_{j=1}^{S} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}|\boldsymbol{y}^{(i),j})$$

We used values of $S \in \{1, 10, 100\}$, finding that the ranking of models was consistent across S values. We used $S = 10$ in all results reported below.

We note that this estimator was originally presented for generative models, and that (C)CE is not a generative training criterion. It seeks to maximize the conditional probability of an observation given its neighborhood. Nonetheless, when implementing our log-likelihood estimator, we treat the model as a generative model, computing the $Z(y)$ constants by summing over all words in the vocabulary.

7.2 System Combination

We can avoid choosing a single model by combining the outputs of multiple models via system combination. We decode test data by using posterior decoding. To combine the outputs of multiple models, we find the max-posterior tag under each model, then choose the highest vote-getter, breaking ties arbitrarily.

However, when doing POS induction without a tag dictionary, the tags are simply unique identifiers and may not have consistent meaning across runs. To address this, we propose a novel voting scheme that is inspired by the widely-used 1-to-1 accuracy metric for POS induction (Haghighi and Klein, 2006). This metric maps system tags to gold tags to maximize accuracy with the constraint that each gold tag is mapped to at most once. The optimal mapping can be found by solving a maximum weighted bipartite matching problem.

We adapt this idea to map tags between two systems, rather than between system tags and gold tags. Given k systems that we want to combine, we choose one to be the **backbone** and map the remaining $k - 1$ systems' outputs to the backbone.[4] After mapping each system's output to the backbone system, we perform simple majority voting among all k systems. To choose the backbone, we consider each of the k systems in turn as backbone and maximize the sum of the weights of the weighted bipartite matching solutions found. This is a heuristic that attempts to choose a backbone that is similar to all other systems. We found that highly-weighted matchings often led to high POS tagging accuracy metrics. We call this voting scheme ALIGN. To see the benefit of ALIGN, we also compare to a simple scheme (NAÏVE) that performs majority voting without any tag mapping.

[4]We use the LEMON C++ toolkit (Dezs et al., 2011) to solve the maximum weighted bipartite matching problems.

8 Experiments

Task and Datasets We consider POS induction without tag dictionaries using five freely-available datasets from the PASCAL shared task (Gelling et al., 2012).[5] These include Danish (DA), using the Copenhagen Dependency Treebank v2 (Buch-Kromann et al., 2007); Dutch (NL), using the Alpino treebank (Bouma et al., 2001); Portuguese (PT), using the Floresta Sintá(c)tica treebank (Afonso et al., 2002); Slovene (SL), using the jos500k treebank (Erjavec et al., 2010); and Swedish (SV), using the Talbanken treebank (Nivre et al., 2006). We use their provided training, development, and test sets.

Evaluation We fix the number of tags in our models to 12, which matches the number of universal tags from Petrov et al. (2012). We use both many-to-1 (M-1) and 1-to-1 (1-1) accuracy as our evaluation metrics, using the universal tags for the gold standard (which was done for the official evaluation for the shared task).[6] We note that our π function assigns identities to tags (e.g., tag 1 is assumed to be NOUN), so we could use actual tagging accuracy when training with the π cost function. But we use M-1 and 1-1 accuracy to enable easier comparison both among different settings and to prior work.

Baselines From the shared task, we compare to all entries that used 12 tags. These include

[5]`http://wiki.cs.ox.ac.uk/InducingLinguisticStructure/SharedTask`

[6]It is common to use a greedy algorithm to compute 1-to-1 accuracy, e.g., as in the shared task scoring script (`http://www.dcs.shef.ac.uk/~tcohn/wils/eval.tar.gz`), though the optimal mapping can be computed efficiently via the maximum weighted bipartite matching algorithm, as stated above. We use the shared task scorer for all results here for ease of comparison. When we instead evaluate using the optimal mapping, we find that accuracies are usually only slightly higher than those found by the greedy algorithm.

BROWN clusters (Brown et al., 1992), clusters obtained using the `mkcls` tool (Och, 1995), and the featurized HMM with sparsity constraints trained using posterior regularization (PR), described by Graça et al. (2011). The PR system achieved the highest average 1-1 accuracy in the shared task.

We restrict our attention to systems that use 12 tags because the M-1 and 1-1 metrics are highly dependent upon the number of hypothesized tags. In general, using more tags leads to higher M-1 and lower 1-1 (Gelling et al., 2012). By keeping the number of tags fixed, we hope to provide a cleaner comparison among approaches.

We compare to two other baselines: an HMM trained with 500 iterations of EM and an HMM trained with 100 iterations of stepwise EM (Liang and Klein, 2009). We used random initialization as done by Liang and Klein: we set each parameter in each multinomial to $\exp\{1 + c\}$, where $c \sim U[0, 1]$, then normalized to get probability distributions. For stepwise EM, we used minibatch size 3 and stepsize reduction power 0.7.

For all models we trained, including both baselines and CCE, we used only the training data during training and used the unannotated development data for certain model selection criteria. No labels were used except for final evaluation on the test data. Therefore, we need a way to handle unknown words in test data. When running EM and stepwise EM, while reading in the final 10% of sentences in the training set, we replace novel words with the special token UNK. We then replace unknown words in test data with UNK.

8.1 CCE Setup

Features We use standard indicator features on tag-tag transitions and tag-word emissions, the spelling features from Smith and Eisner (2005a), and additional emission features based on Brown clusters. The latter features are simply indicators for tag-cluster pairs—analogous to tag-word emissions in which the word is replaced by its Brown cluster identifier. We run Brown clustering (Liang, 2005) on the POS training data for each language, once with 12 clusters and once with 40, then add tag-cluster emission features for each clustering and one more for their conjunction.[7]

Learning We solve Eq. 2 and Eq. 3 by running LBFGS until convergence on the training data, up to 100 iterations. We tag the test data with minimum Bayes risk decoding and evaluate.

We use two neighborhood functions:

- TRANS1: the original sentence along with all sentences that result from doing a single transposition of adjacent words.
- SHUFF10: the original sentence along with 10 random permutations of it.

We use L2 regularization, adding $\frac{C}{2}\sum_{j=1}^{|\boldsymbol{\theta}|}\theta_j^2$ to the objectives shown in Figure 1. We use a fixed (untuned) $C = 0.0001$ for all experiments reported below.[8] We initialize each CE model by sampling weights from $\mathcal{N}(0, 1)$.

Cost Functions The cost functions Δ and π have constants α and β which balance their contributions relative to the model score and must be tuned. We consider the ways proposed in Section 7, namely tuning based on the contrastive estimation criterion computed on development data (CE), the log-likelihood estimate on development data with $S = 10$ (LL), and our two system combination algorithms: naïve voting (NAÏVE) and aligned voting (ALIGN), both of which use as input the 4 system outputs whose hyperparameters led to the highest values for the CE criterion on development data.

We used $\alpha \in \{3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}, 0.01, 0.03, 0.1, 0.3\}$ and $\beta \in \{3 \times 10^{-6}, 10^{-5}, 3 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}\}$. Setting $\alpha = \beta = 0$ gives us CE, which we also compare to. When using both MATLM and UNIV simultaneously, we first choose the best two α values by the LL criterion and the best two β values by the CE criterion when using only those individual costs. This gives us 4 pairs of values; we run experiments with these pairs and choose the pair to report using each of the model selection criteria. For system combination, we use the 4 system outputs resulting from these 4 pairs.

For training bigram language models for the MATLM cost, we use the language's POS training data concatenated with its portion of the Europarl v7 corpus (Koehn, 2005) and the text of its

[7] To handle unknown words: for words that only appear in the final 10% of training sentences, we replace them with UNK when firing their tag-word emission features. We use special Brown cluster identifiers reserved for UNK. But we still use all spelling features derived from the actual word type. For unknown words at test time, we use the UNK emission feature, the Brown cluster features with the special UNK cluster identifiers, and the word's actual spelling features.

[8] In subsequent experiments we tried $C \in \{0.01, 0.001\}$ for the baseline CE setting and found minimal differences.

neigh-borhood	cost	mod. sel.	DA		NL		PT		SL		SV		avg	
			M-1	1-1	M-1	1-1	M-1	1-1	M-1	1-1	M-1	1-1	M-1	1-1
SHUFF10	none	N/A	45.0	38.0	55.1	45.7	**54.2**	38.0	54.7	45.7	47.4	31.3	51.3	39.7
	MATCH	CE	48.9	31.5	56.5	46.4	**54.2**	37.7	55.9	46.8	**48.9**	**33.8**	52.9	39.2
		LL	49.9	34.4	56.5	46.4	54.1	**38.9**	57.2	**48.9**	**48.9**	**33.8**	53.3	40.5
	MATLM	CE	49.1	34.3	**59.6**	**50.4**	53.6	37.1	55.0	46.2	48.8	33.1	53.2	40.2
		LL	**50.2**	**40.0**	**59.6**	**50.4**	53.1	36.0	**58.0**	48.4	48.8	33.1	**53.9**	**41.6**
TRANS1	none	N/A	58.5	42.7	62.5	49.5	**70.7**	**43.8**	58.6	46.1	58.7	53.8	61.8	47.2
	MATCH	CE	58.5	42.5	66.3	53.3	70.6	43.3	59.1	45.6	59.3	54.2	62.7	47.8
		LL	58.8	42.8	66.3	53.3	70.6	43.3	**60.3**	43.7	**59.8**	**54.9**	**63.1**	47.6
	MATLM	CE	**59.4**	**43.5**	63.8	50.1	70.2	43.0	58.5	46.1	59.2	54.8	62.2	47.5
		LL	58.7	42.8	**66.5**	**60.4**	70.5	43.6	59.1	**47.7**	59.2	54.8	62.8	**49.9**

Table 2: Results for observation cost functions. The CE baseline corresponds to rows where cost="none". Other rows are CCE. Best score for each column and each neighborhood is bold.

Wikipedia. The word counts for the Wikipedias used range from 18M for Slovene to 1.9B for Dutch. We used modified Kneser-Ney smoothing as implemented by SRILM (Stolcke, 2002).

8.2 Results

We present two sets of results. First we compare our MATCH and MATLM observation cost functions for our two neighborhoods and two ways of doing model selection. Then we do a broader comparison, comparing both types of costs and their combination to our full set of baselines.

Observation Cost Functions In Table 2, we show results for observation cost functions. We note that the TRANS1 neighborhood works much better than the SHUFF10 neighborhood, but we find that using cost functions can close the gap in certain cases, particularly for Dutch and Slovene for which the SHUFF10 MATLM scores approach or exceed the TRANS1 scores without a cost.

Since the SHUFF10 neighborhood exhibits more diversity than TRANS1, we expect to see larger gains from using observation cost functions. We do in fact see larger gains in M-1, e.g., average improvements are 1.6-2.6 for SHUFF10 and 0.4-1.3 for TRANS1, though 1-1 gains are closer.

For TRANS1, while MATCH does reach a slightly higher average M-1 than MATLM, the latter does much better in 1-1 (49.9 vs. 47.6 when using LL for model selection). For SHUFF10, MATLM consistently does better than MATCH. Nonetheless, we suspect MATCH works as well as it does because it at least differentiates the observation (which is always part of the neighborhood) from the corruptions.

We find that the LL model selection criterion consistently works better than the CE criterion for model selection. When using LL model selection and fixing the neighborhood, all average scores are better than their CE baselines. For M-1, the average improvement is 1.0 to 2.6 points, and for 1-1 the average improvement ranges from 0.4 to 2.7.

We find the best overall performance when using MATLM with LL model selection with the TRANS1 neighborhood, and we report this setting in our subsequent experiments.

Output Cost Function Table 3 shows results when using our UNIV output cost function, as well as our full set of baselines. All (C)CE experiments used the TRANS1 neighborhood.

We find that our contrastive estimation baseline (cost="none") has a higher average M-1 (61.8) than all results from the shared task, but its average 1-1 accuracy is lower than that reached by posterior regularization, the best system in the shared task according to 1-1. Using an observation cost function increases both M-1 and 1-1: MATLM yields an average 1-1 of 49.9, nearing the 50.1 of PR while exceeding it in M-1 by nearly 2 points.

When using the UNIV cost function, we see some variation in performance across model selection criteria, but we find improvements in both M-1 and 1-1 accuracy under most settings. When doing model selection via ALIGN voting, we roughly match the average 1-1 of PR, and when using the CE criterion, we beat it by 1 point on average (51.3 vs. 50.1).

Combined Costs When using the UNIV cost, we find that model selection via CE works better than LL. So for the combined costs, we took the two best MATLM weights (α values) according to LL and the two best UNIV weights (β values) according to CE and ran combined cost experiments (MATCHLM+UNIV) with the four pairs of hyperparameters. Then from among these four,

system	DA		NL		PT		SL		SV		avg	
	M-1	1-1	M-1	1-1	M-1	1-1	M-1	1-1	M-1	1-1	M-1	1-1
HMM, EM	42.5	28.1	53.0	40.6	59.4	33.7	50.3	34.7	49.3	33.9	50.9	34.2
HMM, stepwise EM	51.7	38.2	61.6	45.2	66.5	46.7	53.6	35.7	55.3	39.6	57.7	41.1
BROWN	47.1	39.2	57.3	43.1	67.6	51.6	58.3	42.3	57.6	51.3	57.6	45.5
mkcls	53.1	44.2	63.0	54.1	68.1	46.3	50.4	40.6	57.3	43.6	58.4	45.8
posterior regularization	53.8	45.6	57.6	45.4	74.4	56.1	60.0	**48.5**	58.8	54.9	60.9	50.1

contrastive estimation cost	model sel.											
none	N/A	58.5	42.7	62.5	49.5	70.7	43.8	58.6	46.1	58.7	53.8	61.8	47.2
MATCH	LL	58.8	42.8	66.3	53.3	70.6	43.3	60.3	43.7	59.8	54.9	63.1	47.6
MATLM	LL	58.7	42.8	**66.5**	60.4	70.5	43.6	59.1	47.7	59.2	54.8	62.8	49.9
UNIV	CE	59.7	45.6	60.6	51.1	70.0	62.7	**60.9**	44.1	57.1	52.8	61.7	51.3
	LL	59.5	42.2	62.1	56.3	70.7	43.1	**60.9**	44.1	57.1	52.8	62.1	47.7
	NAÏVE	59.2	45.6	62.2	52.8	72.7	52.7	60.0	43.8	56.2	53.0	62.2	49.6
	ALIGN	**61.6**	**47.3**	63.7	54.5	74.4	53.1	59.7	42.1	56.6	53.2	63.2	50.0
MATLM + UNIV	CE	59.8	45.7	60.4	48.4	70.0	**62.8**	52.9	45.0	59.4	54.9	60.5	51.4
	LL	59.3	42.5	61.9	56.2	70.8	43.1	59.3	41.9	**60.0**	**55.1**	62.3	47.8
	NAÏVE	58.5	44.4	64.9	60.3	65.4	52.1	55.5	45.9	59.0	54.4	60.6	51.4
	ALIGN	61.1	45.4	66.2	**60.9**	**75.8**	49.8	59.5	48.2	59.0	54.4	**64.3**	**51.7**

Table 3: Unsupervised POS tagging accuracies for five languages, showing results for three systems from the PASCAL shared task as well as three other baselines (EM, stepwise EM, and contrastive estimation). All (C)CE results use the TRANS1 neighborhood. The best score in each column is bold.

we again chose results by CE, LL, and both voting schemes.

The results are shown in the lower part of Table 3. We find different trends in M-1 and 1-1 depending on whether we use CE or LL for model selection, which may be due to our limited hyperparameter search stemming from computational constraints. However, by comparing NAÏVE to ALIGN, we see a consistent benefit from aligning tags before voting, leading to our highest average accuracies. In particular, using MATCHLM+UNIV and ALIGN, we improve over CE by 2.5 in M-1 and 4.5 in 1-1, also improving over the best results from the shared task.

9 Conclusion

We have shown how to modify contrastive estimation to use additional sources of knowledge, both in terms of observation and output cost functions. We adapted a recently-proposed technique for estimating the log-likelihood of held-out data, finding it to be effective as a model selection criterion when using observation cost functions. We improved tagging accuracy by using weak supervision in the form of universal tag frequencies. We proposed a system combination method for POS induction systems that consistently performs better than naïve voting and circumvents hyperparameter selection. We reported results on par with or exceeding the best systems from the PASCAL 2012 shared task.

Contrastive estimation has been shown effective for numerous NLP tasks, including dependency grammar induction (Smith and Eisner, 2005b), bilingual part-of-speech induction (Chen et al., 2011), morphological segmentation (Poon et al., 2009), and machine translation (Xiao et al., 2011). The hope is that our contributions can benefit these and other applications of weakly-supervised learning.

Acknowledgments

We thank the anonymous reviewers for their insightful comments and Waleed Ammar, Chris Dyer, David McAllester, Sasha Rush, Nathan Schneider, Noah Smith, and John Wieting for helpful discussions.

References

S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. Floresta sintá(c)tica: a treebank for Portuguese. In *Proc. of LREC*.

Y. Bengio, L. Yao, and K. Cho. 2013. Bounding the test log-likelihood of generative models. *arXiv preprint arXiv:1311.6184*.

T. Berg-Kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein. 2010. Painless unsupervised learning with features. In *Proc. of NAACL*.

P. Blunsom and T. Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proc. of EMNLP*.

P. Blunsom and T. Cohn. 2011. A hierarchical Pitman-Yor process HMM for unsupervised part of speech induction. In *Proc. of ACL*.

G. Bouma, G. Van Noord, and R. Malouf. 2001. Alpino: Wide-coverage computational analysis of Dutch. *Language and Computers*, 37(1).

P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. 1992. Class-based N-gram models of natural language. *Computational Linguistics*, 18(4).

M. Buch-Kromann, J. Wedekind, and J. Elming. 2007. The Copenhagen Danish-English dependency treebank v. 2.0. *code.google.com/p/copenhagen-dependency-treebank*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

D. Chen, C. Dyer, S. B. Cohen, and N. A. Smith. 2011. Unsupervised bilingual POS tagging with Markov random fields. In *Proc. of the First Workshop on Unsupervised Learning in NLP*.

S. Cohen and N. A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proc. of NAACL*.

S. B. Cohen, D. Das, and N. A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proc. of EMNLP*.

M. Creutz and K. Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology.

D. Das and S. Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proc. of ACL*.

S. Della Pietra, V. Della Pietra, and J. Lafferty. 1997. Inducing features of random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(4).

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38.

B. Dezs, A. Jüttner, and P. Kovács. 2011. LEMON - an open source C++ graph template library. *Electron. Notes Theor. Comput. Sci.*, 264(5).

C. B. Do, Q. Le, C. H. Teo, O. Chapelle, and A. Smola. 2008. Tighter bounds for structured estimation. In *Advances in NIPS*.

C. Dyer, J. H. Clark, A. Lavie, and N. A. Smith. 2011. Unsupervised word alignment with arbitrary features. In *Proc. of ACL*.

T. Erjavec, D. Fiser, S. Krek, and N. Ledinek. 2010. The JOS linguistically tagged corpus of Slovene. In *Proc. of LREC*.

K. Ganchev and D. Das. 2013. Cross-lingual discriminative learning of sequence models with posterior regularization. In *Proc. of EMNLP*.

K. Ganchev, J. Gillenwater, and B. Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proc. of ACL*.

K. Ganchev, J. V. Graça, J. Gillenwater, and B. Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11.

D. Gelling, T. Cohn, P. Blunsom, and J. V. Graça. 2012. The PASCAL challenge on grammar induction. In *Proc. of NAACL-HLT Workshop on the Induction of Linguistic Structure*.

K. Gimpel and N. A. Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Proc. of NAACL*.

K. Gimpel and N. A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proc. of NAACL*.

K. Gimpel. 2012. *Discriminative Feature-Rich Modeling for Syntax-Based Machine Translation*. Ph.D. thesis, Carnegie Mellon University.

S. Goldwater and T. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proc. of ACL*.

J. V. Graça, K. Ganchev, L. Coheur, F. Pereira, and B. Taskar. 2011. Controlling complexity in part-of-speech induction. *J. Artif. Int. Res.*, 41(2).

A. Haghighi and D. Klein. 2006. Prototype-driven learning for sequence models. In *Proc. of HLT-NAACL*.

M. Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proc. of EMNLP-CoNLL*.

J. Kaiser, B. Horvat, and Z. Kacic. 2000. A novel loss function for the overall risk criterion based discriminative training of HMM models. In *Proc. of ICSLP*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. of ACL*.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of MT Summit*.

Z. Li, Z. Wang, S. Khudanpur, and J. Eisner. 2010. Unsupervised discriminative language model training for machine translation using simulated confusion sets. In *Proc. of COLING*.

S. Li, J. V. Graça, and B. Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *Proc. of EMNLP*.

P. Liang and D. Klein. 2009. Online EM for unsupervised models. In *Proc. of NAACL*.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *Proc. of HLT-NAACL*.

P. Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2).

T. Naseem, B. Snyder, J. Eisenstein, and R. Barzilay. 2009. Multilingual part-of-speech tagging: Two unsupervised approaches. *JAIR*, 36.

T. Naseem, H. Chen, R. Barzilay, and M. Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proc. of EMNLP*.

J. Nivre, J. Nilsson, and J. Hall. 2006. Talbanken05: A Swedish treebank with phrase structure and dependency annotation. In *Proc. of LREC*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of CoNLL*.

F. J. Och. 1995. Maximum-likelihood-schätzung von wortkategorien mit verfahren der kombinatorischen optimierung. Bachelor's thesis (Studienarbeit), Friedrich-Alexander-Universität Erlangen-Nürnburg, Germany.

S. Petrov, D. Das, and R. McDonald. 2012. A universal part-of-speech tagset. In *Proc. of LREC*.

H. Poon, C. Cherry, and K. Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proc. of HLT: NAACL*.

D. Povey and P. C. Woodland. 2002. Minimum phone error and I-smoothing for improved discrimative training. In *Proc. of ICASSP*.

D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah. 2008. Boosted MMI for model and feature space discriminative training. In *Proc. of ICASSP*.

S. Ravi and K. Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proc. of ACL*.

S. Riezler. 1999. *Probabilistic Constraint Logic Programming*. Ph.D. thesis, Universität Tübingen.

R. Rosenfeld. 1997. A whole sentence maximum entropy language model. In *Proc. of ASRU*.

N. A. Smith and J. Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *Proc. of ACL*.

N. A. Smith and J. Eisner. 2005a. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL*.

N. A. Smith and J. Eisner. 2005b. Guiding unsupervised grammar induction using contrastive estimation. In *Proc. of IJCAI Workshop on Grammatical Inference Applications*.

N. A. Smith and J. Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proc. of COLING-ACL*.

D. A. Smith and J. Eisner. 2009. Parser adaptation and projection with quasi-synchronous features. In *Proc. of EMNLP*.

B. Snyder, T. Naseem, and R. Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proc. of ACL*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2010a. From Baby Steps to Leapfrog: How "Less is More" in unsupervised dependency parsing. In *Proc. of NAACL-HLT*.

V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning. 2010b. Viterbi training improves unsupervised dependency parsing. In *Proc. of CoNLL*.

V. I. Spitkovsky, D. Jurafsky, and H. Alshawi. 2010c. Profiting from mark-up: Hyper-text annotations for guided parsing. In *Proc. of ACL*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2011a. Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *Proc. of EMNLP*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2011b. Punctuation: Making a point in unsupervised dependency parsing. In *Proc. of CoNLL*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proc. of EMNLP*.

A. Stolcke. 2002. SRILM—an extensible language modeling toolkit. In *Proc. of ICSLP*.

O. Täckström, D. Das, S. Petrov, R. McDonald, and J. Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *Advances in NIPS 16*.

K. Toutanova and M. Johnson. 2007. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Advances in NIPS*.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6.

J. Van Gael, A. Vlachos, and Z. Ghahramani. 2009. The infinite HMM for unsupervised POS tagging. In *Proc. of EMNLP*.

A. Vaswani, A. Pauls, and D. Chiang. 2010. Efficient optimization of an MDL-inspired objective function for unsupervised part-of-speech tagging. In *Proc. of ACL*.

M. Wang and C. D. Manning. 2014. Cross-lingual projected expectation regularization for weakly supervised learning. *Transactions of the Association for Computational Linguistics*, 2.

X. Xiao, Y. Liu, Q. Liu, and S. Lin. 2011. Fast generation of translation forest for large-scale SMT discriminative training. In *Proc. of EMNLP*.

Don't Until the Final Verb Wait: Reinforcement Learning for Simultaneous Machine Translation

Alvin C. Grissom II
and **Jordan Boyd-Graber**
Computer Science
University of Colorado
Boulder, CO
Alvin.Grissom@colorado.edu
Jordan.Boyd.Graber@colorado.edu

He He, John Morgan,
and **Hal Daumé III**
Computer Science and UMIACS
University of Maryland
College Park, MD
{hhe,jjm,hal}@cs.umd.edu

Abstract

We introduce a reinforcement learning-based approach to simultaneous machine translation—producing a translation while receiving input words—between languages with drastically different word orders: from verb-final languages (e.g., German) to verb-medial languages (English). In traditional machine translation, a translator must "wait" for source material to appear before translation begins. We remove this bottleneck by predicting the final verb in advance. We use reinforcement learning to learn when to trust predictions about unseen, future portions of the sentence. We also introduce an evaluation metric to measure expeditiousness and quality. We show that our new translation model outperforms batch and monotone translation strategies.

1 Introduction

We introduce a simultaneous machine translation (MT) system that predicts unseen verbs and uses reinforcement learning to learn when to trust these predictions and when to wait for more input.

Simultaneous translation is producing a partial translation of a sentence before the input sentence is complete, and is often used in important diplomatic settings. One of the first noted uses of human simultaneous interpretation was the Nuremberg trials after the Second World War. Siegfried Ramler (2009), the Austrian-American who organized the translation teams, describes the linguistic predictions and circumlocutions that translators would use to achieve a tradeoff between translation latency and accuracy. The audio recording technology used by those interpreters sowed the seeds of technology-assisted interpretation at the United Nations (Gaiba, 1998).

Performing real-time translation is especially difficult when information that comes early in the target language (the language you're translating *to*) comes late in the source language (the language you're translating *from*). A common example is when translating from a verb-final (SOV) language (e.g., German or Japanese) to a verb-medial (SVO) language, (e.g., English). In the example in Figure 1, for instance, the main verb of the sentence (in **bold**) appears at the end of the German sentence. An offline (or "batch") translation system waits until the end of the sentence before translating anything. While this is a reasonable approach, it has obvious limitations. Real-time, interactive scenarios—such as online multilingual video conferences or diplomatic meetings—require comprehensible partial interpretations *before* a sentence ends. Thus, a significant goal in interpretation is to make the translation as **expeditious** as possible.

We present three components for an SOV-to-SVO simultaneous MT system: a reinforcement learning framework that uses predictions to create expeditious translations (Section 2), a system to predict how a sentence will end (e.g., predicting the main verb; Section 4), and a metric that balances quality and expeditiousness (Section 3). We combine these components in a framework that learns *when* to begin translating sections of a sentence (Section 5).

Section 6 combines this framework with a

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1342–1352,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

ich	bin	mit	dem	Zug	nach	Ulm **gefahren**
I	am	with	the	train	to	Ulm **traveled**
I		*(. . . . . waiting. . . . .)*				**traveled** by train to Ulm

Figure 1: An example of translating from a verb-final language to English. The verb, in **bold**, appears at the end of the sentence, preventing coherent translations until the final source word is revealed.

translation system that produces simultaneous translations. We show that our data-driven system can successfully predict unseen parts of the sentence, learn when to trust them, and outperform strong baselines (Section 7).

While some prior research has approached the problem of simultaneous translation—we review these systems in more detail in Section 8—no current model learns *when* to definitively begin translating chunks of an incomplete sentence. Finally, in Section 9, we discuss the limitations of our system: it only uses the most frequent source language verbs, it only applies to sentences with a single main verb, and it uses an idealized translation system. However, these limitations are not insurmountable; we describe how a more robust system can be assembled from these components.

2 Decision Process for Simultaneous Translation

Human interpreters learn strategies for their profession with experience and practice. As words in the source language are observed, a translator—human or machine—must decide whether and how to translate, while, for certain language pairs, simultaneously predicting future words. We would like our system to do the same. To this end, we model simultaneous MT as a Markov decision process (MDP) and use reinforcement learning to effectively combine predicting, waiting, and translating into a coherent strategy.

2.1 States: What is, what is to come

The **state** s_t represents the current view of the world given that we have seen t words of a source language sentence.[1] The state contains information both about what is known and what is predicted based on what is known. To compare the system to a human translator in a decision-making process, the state is akin to the translator's cognitive state. At any given time, we have *knowledge* (observations) and *beliefs* (predictions) with varying degrees of certainty: that is, the state contains the revealed words $x_{1:t}$ of a sentence; the state also contains predictions about the remainder of the sentence: we predict the next word in the sentence and the final verb.

More formally, we have a prediction at time t of the next source language word that will appear, $n_{t+1}^{(t)}$, and for the final verb, $v^{(t)}$. For example, given the partial observation "`ich bin mit dem`", the state might contain a prediction that the next word, $n_{t+1}^{(t)}$, will be "`Zug`" and that the final verb $v^{(t)}$ will be "`gefahren`".

We discuss the mechanics of next-word and verb prediction further in Section 4; for now, consider these black boxes which, after observing every new source word x_t, make predictions of future words in the source language. This representation of the state allows for a richer set of actions, described below, permitting simultaneous translations that outpace the source language input[2] by predicting the future.

2.2 Actions: What our system can do

Given observed and hypothesized input, our simultaneous translation system must decide when to translate them. This is expressed in the form of four actions: our system can **commit** to a partial translation, predict the **next word** and use it to update the translation, predict the **verb** and use it to update the translation, or **wait** for more words.

We discuss each of these actions in turn before describing how they come together to incrementally translate an entire sentence:

Wait Waiting is the simplest action. It produces no output and allows the system to receive more input, biding its time, so that when it does choose to translate, the translation is based on more information.

Commit Committing produces translation output: given the observed source sentence, produce the best translation possible.

[1] We use t to evoke a discrete version of time. We only allow actions after observing a complete source word.

[2] Throughout, "input" refers to source language input, and "output" refers to target language translation.

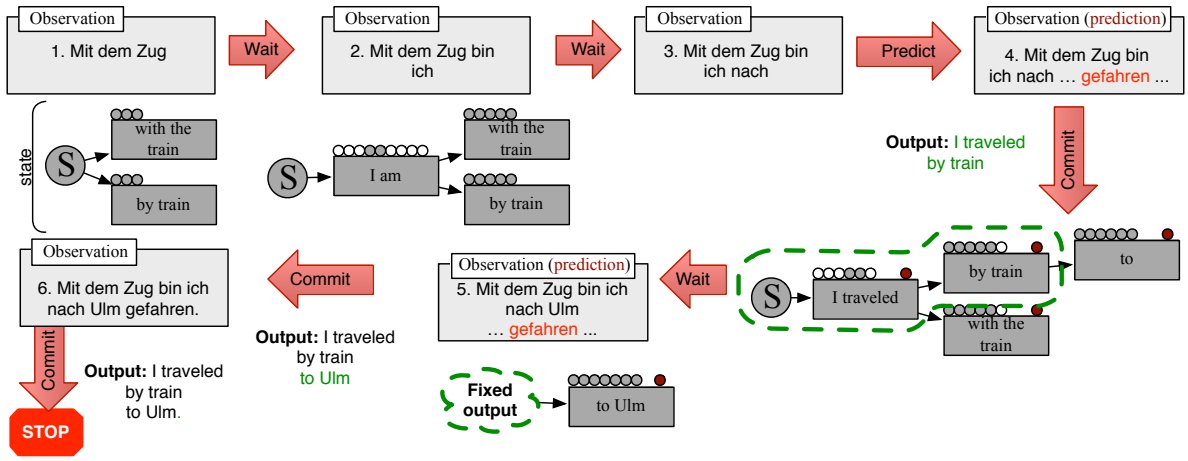


Figure 2: A simultaneous translation from source (German) to target (English). The agent chooses to wait until after (3). At this point, it is sufficiently confident to predict the final verb of the sentence (4). Given this additional information, it can now begin translating the sentence into English, constraining future translations (5). As the rest of the sentence is revealed, the system can translate the remainder of the sentence.

Next Word The **next word** action takes a prediction of the next source word and produces an updated translation based on that prediction, i.e., appending the predicted word to the source sentence and translating the new sentence.

Verb Our system can also predict the source sentence's final verb (the last word in the sentence). When our system takes the **verb** action, it uses its verb prediction to update the translation using the prediction, by placing it at the end of the source sentence.

We can recreate a traditional batch translation system (interpreted temporally) by a sequence of **wait** actions until all input is observed, followed by a **commit** to the complete translation. Our system can **commit** to partial translations if it is confident, but producing a good translation early in the sentence often depends on missing information.

2.3 Translation Process

Having described the state, its components, and the possible actions at a state, we present the process in its entirety. In Figure 2, after each German word is received, the system arrives at a new *state*, which consists of the source input, target translation so far, and predictions of the unseen words. The translation system must then take an *action* given information about the current state. The action will result in receiving and translating more source words, transitioning the system to the next state. In the example, for the first few source-language words, the translator lacks the confidence to produce any output due to insufficient information at the state. However, after State 3, the state shows high confidence in the predicted verb "gefahren". Combined with the German input it has observed, the system is sufficiently confident to act on that prediction to produce English translation.

2.4 Consensus Translations

Three straightforward actions—**commit**, **next word**, and **verb**—all produce translations. These rely black box access to a translation (discussed in detail in Section 6): that is, given a source language sentence fragment, the translation system produces a target language sentence fragment.

Because these actions can happen more than once in a sentence, we must form a single consensus translation from all of the translations that we might have seen. If we have only one translation or if translations are identical, forming the consensus translation is trivial. But how should we resolve conflicting translations?

Any time our system chooses an action that

produces output, the observed input (plus extra predictions in the case of **next-word** or **verb**), is passed into the translation system. That system then produces a complete translation of its input fragment.

Any new words—i.e., words whose target index is greater than the length of any previous translation—are appended to the previous translation.[3] Table 1 shows an example of forming these consensus translations.

Now that we have defined how states evolve based on our system's actions, we need to know how to select which actions to take. Eventually, we will formalize this as a learned policy (Section 5) that maps from states to actions. First, however, we need to define a reward that measures how "good" an action is.

3 Objective: What is a good simultaneous translation?

Good simultaneous translations must optimize two objectives that are often at odds, i.e., producing translations that are, in the end, accurate, and producing them in pieces that are presented expeditiously. While there are existing automated metrics for assessing translation quality (Papineni et al., 2002; Banerjee and Lavie, 2005; Snover et al., 2006), these must be modified to find the necessary compromise between translation quality and expeditiousness. That is, a good metric for simultaneous translation must achieve a balance between translating chunks early and translating accurately. All else being equal, maximizing either goal in isolation is trivial: for accurate translations, use a **batch** system and wait until the sentence is complete, translating it all at once; for a maximally expeditious translation, create **monotone** translations, translating each word as it appears, as in Tillmann et al. (1997) and Pytlik and Yarowsky (2006). The former is not simultaneous at all; the latter is mere word-for-word replacement and results in awkward, often unintelligible translations of distant language pairs.

Once we have predictions, we have an expanded array of possibilities, however. On one extreme, we can imagine a **psychic** translator—one that can completely translate an imagined sentence after one word is uttered—as an unobtainable system. On the other extreme is a standard **batch** translator, which waits until it has access to the utterer's complete sentence before translating anything.

Again, we argue that a system can improve on this by *predicting* unseen parts of the sentence to find a better tradeoff between these conflicting goals. However, to evaluate and optimize such a system, we must measure where a system falls on the continuum of accuracy versus expeditiousness.

Consider partial translations in a two-dimensional space, with time (quantized by the number of source words seen) increasing from left to right on the x axis and the BLEU score (including brevity penalty against the reference length) on the y axis. At each point in time, the system may add to the consensus translation, changing the precision (Figure 3). Like an ROC curve, a good system will be high and to the left, optimizing the area under the curve: the ideal system would produce points as high as possible immediately. A translation which is, in the end, accurate, but which is less expeditious, would accrue its score more slowly but outperform a similarly expeditious system which nevertheless translates poorly.

An idealized psychic system achieves this, claiming all of the area under the curve, as it would have a perfect translation instantly, having no need of even waiting for future input.[4] A batch system has only a narrow (but tall) sliver to the right, since it translates nothing until all of the words are observed.

Formally, let Q be the score function for a partial translation, $\boldsymbol{x}$ the sequentially revealed source words $x_1, x_2, \ldots, x_T$ from time step 1 to T, and $\boldsymbol{y}$ the partial translations $y_1, y_2, \ldots, y_T$, where T is the length of the source language input. Each incremental translation y_t has a BLEU-n score with respect to a reference $\boldsymbol{r}$. We apply the usual BLEU brevity penalty to all the incremental translations (initially empty) to

[3] Using constrained decoding to enforce consistent translation prefixes would complicate our method but is an appealing extension.

[4] One could reasonably argue that this is not ideal: a fluid conversation requires the prosody and timing between source and target to match exactly. Thus, a psychic system would provide too much information too quickly, making information exchange unnatural. However, we take the information-centric approach: more information faster is better.

Pos	Input	Intermediate	Consensus
1			
2	Er	He_1	He_1
3	Er wurde *gestaltet*	It_1 was_2 designed_3	He_1 was_2 designed_3
4		It_1 was_2 designed_3	He_1 was_2 designed_3
5	Er wurde gestern renoviert	It_1 was_2 renovated_3 yesterday_4	He_1 was_2 designed_3 yesterday_4

Table 1: How intermediate translations are combined into a consensus translation. Incorrect translations (e.g., "he" for an inanimate object in position 3) and incorrect predictions (e.g., incorrectly predicting the verb *gestaltet* in position 5) are kept in the consensus translation. When no translation is made, the consensus translation remains static.

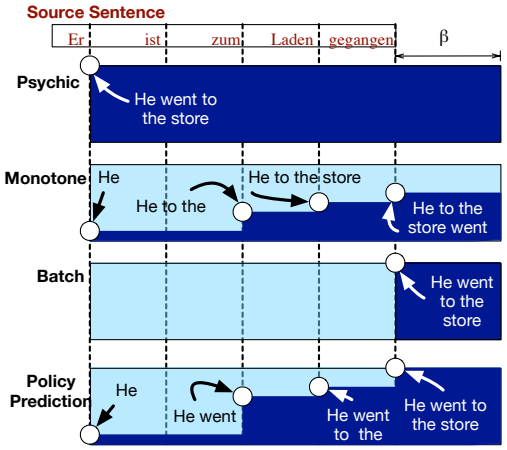


Figure 3: Comparison of LBLEU (the area under the curve given by Equation 1) for an impossible psychic system, a traditional batch system, a monotone (German word order) system, and our prediction-based system. By correctly predicting the verb "gegangen" (to go), we achieve a better overall translation more quickly.

obtain **latency-bleu** (LBLEU),

$$Q(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{T} \sum_t \text{BLEU}(y_t, r) + T \cdot \text{BLEU}(y_T, r) \quad (1)$$

The LBLEU score is a word-by-word integral across the input source sentence. As each source word is observed, the system receives a reward based on the BLEU score of the partial translation. LBLEU, then, represents the sum of these T rewards at each point in the sentence. The score of a simultaneous translation is the sum of the scores of all individual segments that contribute to the overall translation.

We multiply the final BLEU score by T to ensure good final translations in learned systems to compensate for the implicit bias toward low latency.[5]

4 Predicting Verbs and Next Words

The **next** and **verb** actions depend on predictions of the sentence's next word and final verb; this section describes our process for predicting verbs and next words given a partial source language sentence.

The prediction of the next word in the source language sentence is modeled with a left-to-right language model. This is (naïvely) analogous to how a human translator might use his own "language model" to guess upcoming words to gain some speed by completing, for example, collocations before they are uttered. We use a simple bigram language model for next-word prediction. We use Heafield et al. (2013).

For verb prediction, we use a generative model that combines the prior probability of a particular verb v, $p(v)$, with the likelihood of the source context at time t given that verb (namely, $p(x_{1:t} \mid v)$), as estimated by a smoothed Kneser-Ney language model (Kneser and Ney, 1995). We use Pauls and Klein (2011). The prior probability $p(v)$ is estimated by simple relative frequency estimation. The context, $x_{1:t}$, consists of all words observed. We model $p(x_{1:t} \mid v)$ with verb-specific n-gram language models. The predicted verb $v^{(t)}$ at time t is then:

$$\arg\max_v p(v) \prod_{i=1}^{t} p(x_i \mid v, x_{i-n+1:i-1}) \quad (2)$$

[5] One could replace T with a parameter, β, to bias towards different kinds of simultaneous translations. As $\beta \to \infty$, we recover batch translation.

where $x_{i-n+1:i-1}$ is the $n-1$-gram context. To narrow the search space, we consider only the 100 most frequent final verbs, where a "final verb" is defined as the sentence-final sequence of verbs and particles as detected by a German part-of-speech tagger (Toutanova et al., 2003).[6]

5 Learning a Policy

We have a framework (states and actions) for simultaneous machine translation and a metric for assessing simultaneous translations. We now describe the use of reinforcement learning to learn a **policy**, a mapping from states to actions, to maximize LBLEU reward.

We use imitation learning (Abbeel and Ng, 2004; Syed et al., 2008): given an optimal sequence of actions, learn a generalized policy that maps states to actions. This can be viewed as a cost-sensitive classification (Langford and Zadrozny, 2005): a state is represented as a feature vector, the loss corresponds to the quality of the action, and the output of the classifier is the action that should be taken in that state.

In this section, we explain each of these components: generating an optimal policy, representing states through features, and learning a policy that can generalize to new sentences.

5.1 Optimal Policies

Because we will eventually learn policies via a classifier, we must provide training examples to our classifier. These training examples come from an **oracle policy** π^* that demonstrates the optimal sequence—i.e., with maximal LBLEU score—of actions for each sequence. Using dynamic programming, we can determine such actions for a fixed translation model.[7] From this oracle policy, we generate training examples for a supervised classifier. State s_t is represented as a tuple of the observed words $x_{1:t}$, predicted verb $v^{(t)}$, and the predicted word $n_{t+1}^{(t)}$. We represent the state to a classifier as a feature vector $\phi(x_{1:t}, n_{t+1}^{(t)}, v^{(t)})$.

[6]This has the obvious disadvantage of ignoring morphology and occasionally creating duplicates of common verbs that have may be associated with multiple particles; nevertheless, it provides a straightforward verb to predict.

[7]This is possible for the limited class of consensus translation schemes discussed in Section 2.4.

5.2 Feature Representation

We want a feature representation that will allow a classifier to generalize beyond the specific examples on which it is trained. We use several general classes of features: features that describe the input, features that describe the possible translations, and features that describe the quality of the predictions.

Input We include both a bag of words representation of the input sentence as well as the most recent word and bigram to model word-specific effects. We also use a feature that encodes the length of the source sentence.

Prediction We include the identity of the predicted verb and next word as well as their respective probabilities under the language models that generate the predictions. If the model is confident in the prediction, the classifier can learn to more so trust the predictions.

Translation In addition to the state, we include features derived from the possible actions the system might take. This includes a bag of words representation of the target sentence, the score of the translation (decreasing the score is undesirable), the score of the current consensus translation, and the difference between the current and potential translation scores.

5.3 Policy Learning

Our goal is to learn a classifier that can accurately mimic the oracle's choices on previously unseen data. However, at test time, when we run the learned policy classifier, the learned policy's state distribution may deviate from the optimal policy's state distribution due to imperfect imitation, arriving in states not on the oracle's path. To address this, we use SEARN (Daumé III et al., 2009), an iterative imitation learning algorithm. We learn from the optimal policy in the first iteration, as in standard supervised learning; in the following iterations, we run an interpolated policy

$$\pi_{k+1} = \epsilon\pi_k + (1-\epsilon)\pi^*, \qquad (3)$$

with k as the iteration number and ϵ the mixing probability. We collect examples by asking the policy to label states on its path. The interpolated policy will execute the optimal action with probability $1-\epsilon$ and the learned

policy's action with probability ϵ. In the first iteration, we have $\pi_0 = \pi^*$.

Mixing in the learned policy allows the learned policy to slowly change from the oracle policy. As it trains on these no-longer-perfect state trajectories, the state distribution at test time will be more consistent with the states used in training.

SEARN learns the policy by training a cost-sensitive classifier. Besides providing the optimal action, the oracle must also assign a cost to an action

$$\mathcal{C}(a_t, \boldsymbol{x}) \equiv Q(\boldsymbol{x}, \pi^*(x_t)) - Q(\boldsymbol{x}, a_t(x_t)), \quad (4)$$

where $a_t(x_t)$ represents the translation outcome of taking action a_t. The cost is the regret of not taking the optimal action.

6 Translation System

The focus of this work is to show that given an effective batch translation system and predictions, we can learn a policy that will turn this into a simultaneous translation system. Thus, to separate translation errors from policy errors, we perform experiments with a nearly optimal translation system we call an *omniscient* translator.

More realistic translation systems will naturally lower the objective function, often in ways that make it difficult to show that we can effectively predict the verbs in verb-final source languages. For instance, German to English translation systems often drop the verb; thus, predicting a verb that will be ignored by the translation system will not be effective.

The omniscient translator translates a source sentence correctly once it has been fed the appropriate source words as input. There are two edge cases: empty input yields an empty output, while a complete, correct source sentence returns the correct, complete translation. Intermediate cases—where the input is either incomplete or incorrect—require using an alignment. The omniscient translator assumes as input a reference translation r, a partial source language input $x_{1:t}$ and a corresponding partial output y. In addition, the omniscient translator assumes access to an *alignment* between r and x. In practice, we use the HMM aligner (Vogel et al., 1996; Och and Ney, 2003).

We first consider incomplete but correct inputs. Let $y = \tau(x_{1:t})$ be the translator's output given a partial source input $x_{1:t}$ with translation y. Then, $\tau(x_{1:t})$ produces all target words y_j if there is a source word x_i in the input aligned to those words—i.e., $(i, j) \in a_{x,y}$—*and* all preceding target words can be translated. (That translations must be contiguous is a natural requirement for human recipients of translations). In the case where y_j is unaligned, the closest aligned target word to y_j that has a corresponding alignment entry is aligned to x_i; then, if x_i is present in the input, y_j appears in the output. Thus, our omniscient translation system will always produce the correct output given the correct input.

However, our learned policy can make wrong predictions, which can produce partial translations y that do *not* match the reference. In this event, an incorrect source word $\tilde{x}_i$ produces incorrect target words $\tilde{y}_j$, for all j: $(i, j) \in a_{x,y}$. These $\tilde{y}_j$ are sampled from the IBM Model 1 lexical probability table multiplied by the source language model $\tilde{y}_j \sim \text{Mult}(\theta_{\tilde{x}_i}) p_{LM}(\tilde{\boldsymbol{x}})$.[8] Thus, even if we predict the correct verb using a **next word** action, it will be in the wrong position and thus generate a translation from the lexical probabilities. Since translations based on Model 1 probabilities are generally inaccurate, the omniscient translator will do very well when given correct input but will produce very poor translations otherwise.

7 Experiments

In this section, we describe our experimental framework and results from our experiments. From aligned data, we derive an omniscient translator. We use monolingual data in the source language to train the verb predictor and the next word predictor. From these features, we compute an optimal policy from which we train a learned policy.

7.1 Data sets

For translation model and policy training, we use data from the German-English Parallel "de-news" corpus of radio broadcast news (Koehn, 2000), which we lower-cased and stripped of

[8] If a policy chooses an incorrect unaligned word, it has no effect on the output. Alignments are position-specific, so "wrong" refers to position and type.

punctuation. A total of 48,601 sentence pairs are randomly selected for building our system. Of these, we use 70% (34,528 pairs) for training word alignments.

For training the translation policy, we restrict ourselves to sentences that end with one of the 100 most frequent verbs (see Section 4). This results in a data set of 4401 training sentences and 1832 test sentences from the de-news data. We did this to narrow the search space (from thousands of possible, but mostly very infrequent, verbs).

We used 1 million words of news text from the Leipzig Wortschatz (Quasthoff et al., 2006) German corpus to train 5-gram language models to predict a verb from the 100 most frequent verbs.

For next-word prediction, we use the 18,345 most frequent German bigrams from the training set to provide a set of candidates in a language model trained on the same set. We use frequent bigrams to reduce the computational cost of finding the completion probability of the next word.

7.2 Training Policies

In each iteration of SEARN, we learn a multi-class classifier to implement the policy. The specific learning algorithm we use is AROW (Crammer et al., 2013). In the complete version of SEARN, the cost of each action is calculated as the highest expected reward starting at the current state minus the actual roll-out reward. However, computing the full roll-out reward is computationally very expensive. We thus use a surrogate binary cost: if the predicted action is the same as the optimal action, the cost is 0; otherwise, the cost is 1. We then run SEARN for five iterations. Results on the development data indicate that continuing for more iterations yields no benefit.

7.3 Policy Rewards on Test Set

In Figure 4, we show performance of the optimal policy *vis-à-vis* the learned policy, as well as the two baseline policies: the batch policy and the monotone policy. The x-axis is the percentage of the source sentence seen by the model, and the y-axis is a smoothed average of the reward as a function of the percentage of the sentence revealed. The monotone policy's performance is close to the optimal policy for

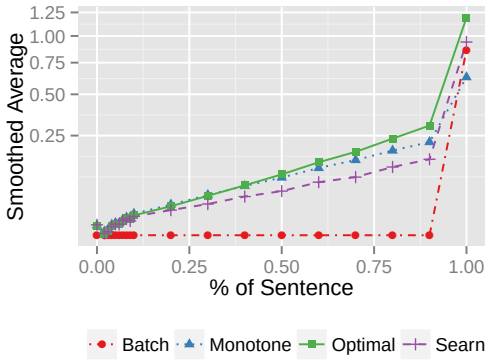


Figure 4: The final reward of policies on German data. Our policy outperforms all baselines by the end of the sentence.

Figure 5: Histogram of actions taken by the policies.

the first half of the sentence, as German and English have similar word order, though they diverge toward the end. Our learned policy outperforms the monotone policy toward the end and of course outperforms the batch policy throughout the sentence.

Figure 5 shows counts of actions taken by each policy. The batch policy always commits at the end. The monotone policy commits at each position. Our learned policy has an action distribution similar to that of the optimal policy, but is slightly more cautious.

7.4 What Policies Do

Figure 6 shows a policy that, predicting incorrectly, still produces sensible output. The policy correctly intuits that the person discussed

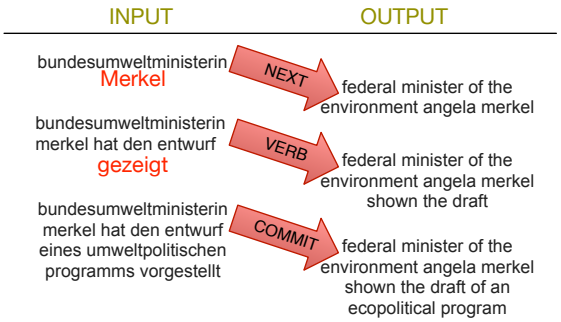


Figure 6: An imperfect execution of a learned policy. Despite choosing the wrong verb "gezeigt" (showed) instead of "vorgestellt" (presented), the translation retains the meaning.

is Angela Merkel, who was the environmental minister at the time, but the policy uses an incorrectly predicted verb. Because of our poor translation model (Section 6), it renders this word as "shown", which is a poor translation. However, it is still comprehensible, and the overall policy is similar to what a human would do: intuit the subject of the sentence from early clues and use a more general verb to stand in for a more specific one.

8 Related Work

Just as MT was revolutionized by statistical learning, we suspect that simultaneous MT will similarly benefit from this paradigm, both from a systematic system for simultaneous translation and from a framework for learning how to incorporate predictions.

Simultaneous translation has been dominated by rule and parse-based approaches (Mima et al., 1998a; Ryu et al., 2006). In contrast, although Verbmobil (Wahlster, 2000) performs incremental translation using a statistical MT module, its incremental decision-making module is rule-based. Other recent approaches in speech-based systems focus on waiting until a pause to translate (Sakamoto et al., 2013) or using word alignments (Ryu et al., 2012) between languages to determine optimal translation units.

Unlike our work, which focuses on prediction and learning, previous strategies for dealing with SOV-to-SVO translation use rule-based methods (Mima et al., 1998b) (for instance, passivization) to buy time for the translator to hear more information in a spoken context—or use phrase table and reordering probabilities to decide where to translate with less delay (Fujita et al., 2013). Oda et al. (2014) is the most similar to our work on the translation side. They frame word segmentation as an optimization problem, using a greedy search and dynamic programming to find segmentation strategies that maximize an evaluation measure. However, unlike our work, the direction of translation was from *from* SVO to SVO, obviating the need for verb prediction. Simultaneous translation is more straightforward for languages with compatible word orders, such as English and Spanish (Fügen, 2008).

To our knowledge, the only attempt to specifically predict verbs or any late-occurring terms (Matsubara et al., 2000) uses pattern matching on what would today be considered a small data set to predict English verbs for Japanese to English simultaneous MT.

Incorporating verb predictions into the translation process is a significant component of our framework, though n-gram models strongly prefer highly frequent verbs. Verb prediction might be improved by applying the insights from psycholinguistics. Ferreira (2000) argues that verb lemmas are required early in sentence production—prior to the first noun phrase argument—and that multiple possible syntactic hypotheses are maintained in parallel as the sentence is produced. Schriefers et al. (1998) argues that, in simple German sentences, non-initial verbs do not need lemma planning at all. Momma et al. (2014), investigating these prior claims, argues that the abstract relationship between the internal arguments and verbs triggers *selective* verb planning.

9 Conclusion and Future Work

Creating an effective simultaneous translation system for SOV to SVO languages requires not only translating partial sentences, but also effectively predicting a sentence's verb. Both elements of the system require substantial refinement before they are usable in a real-world system.

Replacing our idealized translation system is the most challenging and most important next step. Supporting multiple translation hypotheses and incremental decoding (Sankaran

et al., 2010) would improve both the efficiency and effectiveness of our system. Using data from human translators (Shimizu et al., 2014) could also add richer strategies for simultaneous translation: passive constructions, reordering, etc.

Verb prediction also can be substantially improved both in its scope in the system and how we predict verbs. Verb-final languages also often place verbs at the end of clauses, and also predicting these verbs would improve simultaneous translation, enabling its effective application to a wider range of sentences. Instead predicting an *exact* verb early (which is very difficult), predicting a semantically close or a more general verb might yield interpretable translations.

A natural next step is expanding this work to other languages, such as Japanese, which not only has SOV word order but also requires tokenization and morphological analysis, perhaps requiring sub-word prediction.

Acknowledgments

We thank the anonymous reviewers, as well as Yusuke Miyao, Naho Orita, Doug Oard, and Sudha Rao for their insightful comments. This work was supported by NSF Grant IIS-1320538. Boyd-Graber is also partially supported by NSF Grant CCF-1018625. Daumé III and He are also partially supported by NSF Grant IIS-0964681. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References

Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the International Conference of Machine Learning*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Association for Computational Linguistics*. Association for Computational Linguistics.

Koby Crammer, Alex Kulesza, and Mark Dredze. 2013. Adaptive regularization of weight vectors. *Machine Learning*, 91(2):155–187.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning Journal (MLJ)*.

Fernanda Ferreira. 2000. Syntax in language production: An approach using tree-adjoining grammars. *Aspects of language production*, pages 291–330.

Christian Fügen. 2008. *A system for simultaneous translation of lectures and speeches*. Ph.D. thesis, KIT-Bibliothek.

Tomoki Fujita, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2013. Simple, lexicalized choice of translation timing for simultaneous speech translation. *INTERSPEECH*.

Francesca Gaiba. 1998. *The origins of simultaneous interpretation: The Nuremberg Trial*. University of Ottawa Press.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the Association for Computational Linguistics*.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for n-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. IEEE.

Philipp Koehn. 2000. German-english parallel corpus "de-news".

John Langford and Bianca Zadrozny. 2005. Relating reinforcement learning performance to classification performance. In *Proceedings of the International Conference of Machine Learning*.

Shigeki Matsubara, Keiichi Iwashima, Nobuo Kawaguchi, Katsuhiko Toyama, and Yasuyoshi Inagaki. 2000. Simultaneous Japanese-English interpretation based on early predition of English verb. In *Symposium on Natural Language Processing*.

Hideki Mima, Hitoshi Iida, and Osamu Furuse. 1998a. Simultaneous interpretation utilizing example-based incremental transfer. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pages 855–861. Association for Computational Linguistics.

Hideki Mima, Hitoshi Iida, and Osamu Furuse. 1998b. Simultaneous interpretation utilizing example-based incremental transfer. In *Proceedings of the Association for Computational Linguistics*.

Shota Momma, Robert Slevc, and Colin Phillips. 2014. The timing of verb selection in english active and passive sentences.

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

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the Association for Computational Linguistics*, June.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics*.

Adam Pauls and Dan Klein. 2011. Faster and smaller n-gram language models. In *Proceedings of the Association for Computational Linguistics*.

Brock Pytlik and David Yarowsky. 2006. Machine translation for languages lacking bitext via multilingual gloss transduction. In *5th Conference of the Association for Machine Translation in the Americas (AMTA)*, August.

Uwe Quasthoff, Matthias Richter, and Christian Biemann. 2006. Corpus portal for search in monolingual corpora. In *International Language Resources and Evaluation*, pages 1799–1802.

Siegfried Ramler and Paul Berry. 2009. *Nuremberg and Beyond: The Memoirs of Siegfried Ramler from 20th Century Europe to Hawai'i*. Booklines Hawaii Limited.

Koichiro Ryu, Shigeki Matsubara, and Yasuyoshi Inagaki. 2006. Simultaneous english-japanese spoken language translation based on incremental dependency parsing and transfer. In *Proceedings of the Association for Computational Linguistics*.

Koichiro Ryu, Shigeki Matsubara, and Yasuyoshi Inagaki. 2012. Alignment-based translation unit for simultaneous japanese-english spoken dialogue translation. In *Innovations in Intelligent Machines–2*, pages 33–44. Springer.

Akiko Sakamoto, Nayuko Watanabe, Satoshi Kamatani, and Kazuo Sumita. 2013. Development of a simultaneous interpretation system for face-to-face services and its evaluation experiment in real situation.

Baskaran Sankaran, Ajeet Grewal, and Anoop Sarkar. 2010. Incremental decoding for phrase-based statistical machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation*.

H Schriefers, E Teruel, and RM Meinshausen. 1998. Producing simple sentences: Results from picture–word interference experiments. *Journal of Memory and Language*, 39(4):609–632.

Hiroaki Shimizu, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Collection of a simultaneous translation corpus for comparative analysis. In *International Language Resources and Evaluation*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*.

Umar Syed, Michael Bowling, and Robert E. Schapire. 2008. Apprenticeship learning using linear programming. In *Proceedings of the International Conference of Machine Learning*.

Christoph Tillmann, Stephan Vogel, Hermann Ney, and Alex Zubiaga. 1997. A dp-based search using monotone alignments in statistical translation. In *Proceedings of the Association for Computational Linguistics*.

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

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*.

Wolfgang Wahlster. 2000. *Verbmobil: foundations of speech-to-speech translation*. Springer.

PCFG Induction for Unsupervised Parsing and Language Modelling

James Scicluna
Université de Nantes,
CNRS, LINA, UMR6241,
F-44000, France
james.scicluna@univ-nantes.fr

Colin de la Higuera
Université de Nantes,
CNRS, LINA, UMR6241,
F-44000, France
cdlh@univ-nantes.fr

Abstract

The task of unsupervised induction of probabilistic context-free grammars (PCFGs) has attracted a lot of attention in the field of computational linguistics. Although it is a difficult task, work in this area is still very much in demand since it can contribute to the advancement of language parsing and modelling. In this work, we describe a new algorithm for PCFG induction based on a principled approach and capable of inducing accurate yet compact artificial natural language grammars and typical context-free grammars. Moreover, this algorithm can work on large grammars and datasets and infers correctly even from small samples. Our analysis shows that the type of grammars induced by our algorithm are, in theory, capable of modelling natural language. One of our experiments shows that our algorithm can potentially outperform the state-of-the-art in unsupervised parsing on the WSJ10 corpus.

1 Introduction

The task of unsupervised induction of PCFGs has attracted a lot of attention in the field of computational linguistics. This task can take the form of either parameter search or structure learning. In parameter search, a CFG is fixed and the focus is on assigning probabilities to this grammar using Bayesian methods (Johnson et al., 2007) or maximum likelihood estimation (Lari and Young, 1990). In structure learning, the focus is on building the right grammar rules from scratch. We take the latter approach.

Unsupervised structure learning of (P)CFGs is a notoriously difficult task (de la Higuera, 2010; Clark and Lappin, 2010), with theoretical results showing that in general it is either impossible to achieve (Gold, 1967; de la Higuera, 1997) or requires impractical resources (Horning, 1969; Yang, 2012). At the same time, it is well known that context-free structures are needed for better language parsing and modelling, since less expressive models (such as HMMs) are not good enough (Manning and Schütze, 2001; Jurafsky and Martin, 2008). Moreover, the trend is towards unsupervised (rather than supervised) learning methods due to the lack in most languages of annotated data and the applicability in wider domains (Merlo et al., 2010). Thus, despite its difficulty, unsupervised PCFG grammar induction (or induction of other similarly expressive models) is still an important task in computational linguistics.

In this paper, we describe a new algorithm for PCFG induction based on a principled approach and capable of inducing accurate yet compact grammars. Moreover, this algorithm can work on large grammars and datasets and infers correctly even from small samples. We show that our algorithm is capable of achieving competitive results in both unsupervised parsing and language modelling of typical context-free languages and artificial natural language grammars. We also show that the type of grammars we propose to learn are, in theory, capable of modelling natural language.

2 Preliminaries

2.1 Grammars and Languages

A *context-free grammar* (CFG) is a 4-tuple $\langle N, \Sigma, P, I \rangle$, where N is the set of non-terminals, Σ the set of terminals, P the set of production rules and I a set of starting non-terminals (i.e. multiple starting non-terminals are possible). The language generated from a particular non-terminal A is $L(A) = \{w | A \stackrel{*}{\Rightarrow} w\}$ and the language generated by a grammar G is $L(G) = \bigcup_{S \in I} L(S)$. A CFG is in *Chomsky Normal Form (CNF)* if ev-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1353–1362,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

ery production rule is of the form $N \rightarrow NN$ or $N \rightarrow \Sigma$.

A *probabilistic context-free grammar* (PCFG) is a CFG with a probability value assigned to every rule and every starting non-terminal. The probability of a leftmost derivation from a PCFG is the product of the starting non-terminal probability and the production probabilities used in the derivation. The probability of a string generated by a PCFG is the sum of all its leftmost derivations' probabilities. The stochastic language generated from a PCFG G is $(L(G), \phi_G)$, where ϕ_G is the distribution over Σ^* defined by the probabilities assigned to the strings by G. For a PCFG to be *consistent*, the probabilities of the strings in its stochastic language must add up to 1 (Wetherell, 1980). Any PCFG mentioned from now onwards is assumed to be consistent.

2.2 Congruence Relations

A *congruence relation* $\sim$ on Σ^* is any equivalence relation on Σ^* that respects the following condition: if $u \sim v$ and $x \sim y$ then $ux \sim vy$. The congruence classes of a congruence relation are simply its equivalence classes. The congruence class of $w \in \Sigma^*$ w.r.t. a congruence relation $\sim$ is denoted by $[w]_\sim$. The set of *contexts* of a substring w with respect to a language L, denoted $Con(w, L)$, is $\{(l, r) \in \Sigma^* \times \Sigma^* \mid lwr \in L\}$. Two strings u and v are *syntactically congruent* with respect to L, written $u \equiv_L v$, if $Con(u, L) = Con(v, L)$. This is a congruence relation on Σ^*. The *context distribution* of a substring w w.r.t. a stochastic language (L, ϕ), denoted $C_w^{(L,\phi)}$, is a distribution whose support is all the possible contexts over alphabet Σ (i.e. $\Sigma^* \times \Sigma^*$) and is defined as follows:

$$C_w^{(L,\phi)}(l, r) = \frac{\phi(lwr)}{\sum_{l',r' \in \Sigma^*} \phi(l'wr')}$$

Two strings u and v are *stochastically congruent* with respect to (L, ϕ), written $u \cong_{(L,\phi)} v$, if $C_u^{(L,\phi)}$ is equal to $C_v^{(L,\phi)}$. This is a congruence relation on Σ^*.

2.3 Congruential Grammars

Clark (2010a) defines *Congruential CFGs* (C-CFGs) as being all the CFGs G which, for any non-terminal A, if $u \in L(A)$ then $L(A) \subseteq [u]_{\equiv_{L(G)}}$ (where $[u]_{\equiv_{L(G)}}$ is the syntactic congruence class of u w.r.t. the language of G). This class of grammars was defined with learnability in mind. Since these grammars have a direct relationship between congruence classes and the non-terminals, their learnability is reduced to that of finding the correct congruence classes (Clark, 2010a).

This class of grammars is closely related to the class of NTS-grammars (Boasson and Sénizergues, 1985). Any C-CFG is an NTS-grammar but not vice-versa. However, it is not known whether languages generated by C-CFGs are all NTS-languages (Clark, 2010a). Note that NTS-languages are a subclass of deterministic context-free languages and contain the regular languages, the substitutable (Clark and Eyraud, 2007) and k-l-substitutable context-free languages (Yoshinaka, 2008), the very simple languages and other CFLs such as the Dyck language (Boasson and Sénizergues, 1985).

We define a slightly more restrictive class of grammars, which we shall call *Strongly Congruential CFGs* (SC-CFGs). A CFG G is a SC-CFG if, for any non-terminal A, if $u \in L(A)$ then $L(A) = [u]_{\equiv_{L(G)}}$. The probabilistic equivalent of this is the class of *Strongly Congruential PCFGs* (SC-PCFGs), defined as all the PCFGs G which, for any non-terminal A, if $u \in L(A)$ then $L(A) = [u]_{\cong_{(L(G),\phi)}}$. In other words, the non-terminals (i.e. syntactic categories in natural language) of these grammars directly correspond to classes of substitutable strings (i.e. substitutable words and phrases in NL). One may ask whether this is too strict a restriction for natural language grammars. We argue that it is not, for the following reasons.

First of all, this restriction complies with the approach taken by American structural linguists for the identification of syntactic categories, as shown by Rauh (2010): *"[Zellig and Fries] identified syntactic categories as distribution classes, employing substitution tests and excluding semantic properties of the items analysed. Both describe syntactic categories exclusively on the basis of their syntactic environments and independently of any inherent properties of the members of these categories"*.

Secondly, we know that such grammars are capable of describing languages generated by grammars that contain typical natural language grammatical structures (see Section 4.1; artificial natural language grammars NL1-NL7, taken from various sources, generate languages which can be described by SC-PCFGs).

3 Algorithm

COMINO (our algorithm) induces SC-PCFGs from a positive sample S. The steps involved are:

1. Inducing the stochastically congruent classes of all the substrings of S
2. Selecting which of the induced classes are non-terminals and subsequently building a CFG.
3. Assigning probabilities to the induced CFG.

The approach we take is very different from traditional grammar induction approaches, in which grouping of substitutable substrings is done incrementally as the same groups are chosen to represent non-terminals. We separate these two task so that learning takes place in the grouping phase whilst selection of non-terminals is done independently by solving a combinatorial problem.

For the last step, the standard EM-algorithm for PCFGs (Lari and Young, 1990) is used. In Sections 3.1 and 3.2, the first and second steps of the algorithm are described in detail. We analyse our algorithm in Section 3.3.

3.1 Inducing the Congruence Classes

We describe in Algorithm 1 how the congruence classes are induced.

Algorithm 1: Learn Congruence Classes

Input: A multiset S; parameters: n, d, i; distance function `dist` on local contexts of size k

Output: The congruence classes $\mathcal{CC}$ over the substrings of S

1 $Subs \leftarrow$ Set of all substrings of S ;
2 $\mathcal{CC} \leftarrow \{\{w\} \mid w \in Subs\}$;
3 **while** *True* **do**
4 $\quad Pairs \leftarrow \{(x, y) \mid x, y \in \mathcal{CC},\ x \neq y, |S|_x \geq n\ ,\ |S|_y \geq n\}$;
5 $\quad$ **if** $|Pairs| = 0$ **then exitloop** ;
6 $\quad$ Order $Pairs$ based on $\texttt{dist}_k$;
7 $\quad (x, y) \leftarrow Pairs[0]$;
8 $\quad init = \{[w]_{\mathcal{CC}} \mid w \in S\}$;
9 $\quad$ **if** $\texttt{dist}_k(x, y) \geq d$ **and** $|init| \leq i$ **then exitloop** ;
10 $\quad \mathcal{CC} \leftarrow \texttt{Merge}(x, y, \mathcal{CC})$;
11 **end**
12 **return** $\mathcal{CC}$;

At the beginning, each substring (or phrase for natural language) in the sample is assigned its own congruence class (line 2). Then, pairs of *frequent* congruence classes are *merged* together depending on the *distance* between their *empirical context distribution*, which is calculated on *local contexts*. The following points explain each keyword:

- The *empirical context distribution* of a substring w is simply a probability distribution over all the contexts of w, where the probability for a context (l, r) is the number of occurrences of lwr in the sample divided by the number of occurrences of w. This is extended to congruence classes by treating each substring in the class as one substring (i.e. the sum of occurrences of lw_ir, for all w_i in the class, divided by the sum of occurrences of all w_i).
- Due to the problem of sparsity with contexts (in any reasonably sized corpus of natural language, very few phrases will have more than one occurrence of the same context), only *local contexts* are considered. The local contexts of w are the pairs of first k symbols (or words for natural language) preceding and following w. The lower k is, the less sparsity is a problem, but the empirical context distribution is less accurate. For natural language corpora, k is normally set to 1 or 2.
- A *frequent* congruence class is one whose substring occurrences in the sample add up to more than a pre-defined threshold n. Infrequent congruence classes are ignored due to their unreliable empirical context distribution. However, as more merges are made, more substrings are added to infrequent classes, thus increasing their frequency and eventually they might be considered as frequent classes.
- A *distance* function `dist` between samples of distributions over contexts is needed by the algorithm to decide which is the closest pair of congruence classes, so that they are merged together. We used L1-Distance and Pearson's chi-squared test for experiments in Sections 4.1 and 4.2 respectively.
- After each *merge*, other merges are logically deduced so as to ensure that the relation re-

mains a congruence[1]. In practice, the vast majority of the merges undertaken are logically deduced ones. This clearly relieves the algorithm from taking unnecessary decisions (thus reducing the chance of erroneous decisions). On the downside, one bad merge can have a disastrous ripple effect. Thus, to minimize as much as possible the chance of this happening, every merge undertaken is the best possible one at that point in time (w.r.t. the distance function used). The same idea is used in DFA learning (Lang et al., 1998).

This process is repeated until either 1) no pairs of frequent congruence classes are left to merge (line 5) or 2) the smallest distance between the candidate pairs is bigger or equal to a pre-defined threshold d and the number of congruence classes containing strings from the sample is smaller or equal to a pre-defined threshold i (line 9).

The first condition of point 2 ensures that congruence classes which are sufficiently close to each other are merged together. The second condition of point 2 ensures that the hypothesized congruence classes are generalized enough (i.e. to avoid undergeneralization). For natural language examples, one would expect that a considerable number of sentences are grouped into the same class because of their similar structure. Obviously, one can make use of only one of these conditions by assigning the other a parameter value which makes it trivially true from the outset (0 for d and $|Subs|$ for i).

3.2 Building the Context-Free Grammar

Deciding which substrings are constituents (in our case, this translates into choosing which congruence classes correspond to non-terminals) is a problematic issue and is considered a harder task than the previous step (Klein, 2004). A path followed by a number of authors consists in using an Ockham's razor or Minimal Description Length principle approach (Stolcke, 1994; Clark, 2001; Petasis et al., 2004). This generally leads to choosing as best hypothesis the one which best compresses the data. Applying this principle in our case would mean that the non-terminals should be assigned in such a way that the grammar built is *the smallest possible* one (in terms of the number of non-terminals and/or production rules) consistent with the congruence classes. To our knowledge, only local greedy search is used by systems in the literature which try to follow this approach.

We propose a new method for tackling this problem. We show that all the possible SC-CFGs in CNF consistent with the congruence classes directly correspond to all the solutions of a boolean formula built upon the congruence classes, where the variables of this formula correspond to non-terminals (and, with some minor adjustments, production rules as well). Thus, finding the smallest possible grammar directly translates into finding a solution which has the smallest possible amount of true variables. Finding a minimal solution for this type of formula is a known NP-Hard problem (Khanna et al., 2000). However, sophisticated linear programming solvers (Berkelaar et al., 2008) can take care of this problem. For small examples (e.g. all the examples in Table 1), these solvers are able to find an exact solution in a few seconds. Moreover, these solvers are capable of finding good approximate solutions to larger formulas containing a few million variables.

The formula contains one variable per congruence class. All variables corresponding to congruence classes containing strings from the sample are assigned the value `True` (since there must be a starting non-terminal that generates these strings). All variables corresponding to congruence classes containing symbols from Σ are assigned the value `True` (since for every $a \in \Sigma$, there must be a rule $A \rightarrow a$). Finally, and most importantly, for every congruence class $[w]$ and for every string w in $[w]$ ($|w| = n$), the following conditional statement is added to the formula:

$$v(w) \Rightarrow (v(w_{1,1}) \wedge v(w_{2,n})) \vee (v(w_{1,2}) \wedge v(w_{3,n})) \vee \ldots \vee (v(w_{1,n-1}) \wedge v(w_{n,n}))$$

where $v(x)$ is the variable corresponding to the congruence class $[x]$ and $w_{i,j}$ is the substring of w from the i^{th} to the j^{th} symbol of w. This statement is representing the fact that if a congruence class $[w]$ is chosen as a non-terminal then for each string in $w \in [w]$, there must be at least one CNF rule $A \rightarrow BC$ that generates w and thus there must be at least one division of w into $w_{1,k}w_{k+1,n}$ such that B corresponds to $[w_{1,k}]$ and C corresponds to $[w_{k+1,n}]$.

The grammar extracted from the solution of this

[1] for example, if a congruence class contains the phrases "the big" and "that small", and another class contains "dog barked" and "cat meowed", it can be logically deduced that the phrases "the big dog barked","the big cat meowed", "that small dog barked" and "that small cat meowed" should be in the same class.

formula is made up of all the possible CNF production rules built from the chosen non-terminals. The starting non-terminals are those which correspond to congruence classes that contain at least one string from the sample.

The following is a run of the whole process on a simple example:

Sample $\{ab, aabb, aaabbb\}$

Congruence Classes

$1 : [a]$, $2 : [b]$, $3 : [ab, aabb, aaabbb]$, $4 : [aa]$, $5 : [bb]$, $6 : [aab, aaabb]$, $7 : [abb, aabbb]$, $8 : [aaa]$, $9 : [bbb]$, $10 : [aaab]$, $11 : [abbb]$

Boolean Formula

There is one conditional statement per substring. For example, $X_6 \Rightarrow (X_1 \wedge X_3) \vee (X_4 \wedge X_2)$ represents the two possible ways aab in congruence class 6 can be split ($a|ab$, $aa|b$).

Variables X_1, X_2 and X_3 are true.

$X_3 \Rightarrow (X_1 \wedge X_2)$
$X_3 \Rightarrow (X_1 \wedge X_7) \vee (X_4 \wedge X_5) \vee (X_6 \wedge X_2)$
$X_3 \Rightarrow (X_1 \wedge X_7) \vee (X_4 \wedge X_{11}) \vee (X_8 \wedge X_9) \vee (X_{10} \wedge X_5) \vee (X_6 \wedge X_2)$
$X_4 \Rightarrow (X_1 \wedge X_1)$
$X_5 \Rightarrow (X_2 \wedge X_2)$
$X_6 \Rightarrow (X_1 \wedge X_3) \vee (X_4 \wedge X_2)$
$X_6 \Rightarrow (X_1 \wedge X_3) \vee (X_4 \wedge X_7) \vee (X_8 \wedge X_5) \vee (X_{10} \wedge X_2)$
$X_7 \Rightarrow (X_1 \wedge X_5) \vee (X_3 \wedge X_2)$
$X_7 \Rightarrow (X_1 \wedge X_{11}) \vee (X_4 \wedge X_9) \vee (X_6 \wedge X_5) \vee (X_3 \wedge X_2)$
$X_8 \Rightarrow (X_1 \wedge X_4) \vee (X_4 \wedge X_1)$
$X_9 \Rightarrow (X_2 \wedge X_5) \vee (X_5 \wedge X_2)$
$X_{10} \Rightarrow (X_1 \wedge X_6) \vee (X_4 \wedge X_3) \vee (X_8 \wedge X_2)$
$X_{11} \Rightarrow (X_1 \wedge X_9) \vee (X_3 \wedge X_5) \vee (X_7 \wedge X_2)$

Solution

Running the solver on this formula will return the following true variables that make up a minimal solution: X_1, X_2, X_3 and X_7.

Grammar

For every statement $x \Rightarrow \ldots \vee (y \wedge z) \vee \ldots$ where x,y and z are true, a production rule $x \rightarrow yz$ is added. So, the following grammar is built:

X_3 is the starting non-terminal
$X_3 \rightarrow X_1X_7 \mid X_1X_2 \quad X_7 \rightarrow X_3X_2$
$X_1 \rightarrow a \quad X_2 \rightarrow b$

3.3 Analysis

In the first phase of the algorithm, we are grouping all the substrings of the sample S according to the congruence relation $\cong_{(L,\phi)}$, where (L, ϕ) is the target stochastic language (for natural language, this is the language model). To do so, we are assuming that S was i.i.d. generated from (L, ϕ). In the second phase, we are representing the space of all CFGs consistent with the classes obtained in phase one as different solutions to a boolean formula. Here we introduce our bias in favour of smaller grammars by finding a minimal solution to the formula. In the last phase, probabilities are assigned to the grammar obtained in phase two using the standard MLE algorithm for PCFGs.

Unlike many other systems, in our case the hypothesis space of grammars is well-defined. This allows us to analyse our algorithm in a theoretical framework and obtain theoretical learnability results. Moreover, this gives us an idea on the types of syntactical features our system is capable of learning.

Assuming our algorithm always takes correct merge decisions, the sample required for identification needs only to be structurally complete w.r.t. the target grammar (i.e. every production rules is used at least once in the generation of the sample). This means that, in theory, our algorithm can work with very small samples (polynomial size w.r.t. the number of rules in the target grammar).

Some approaches in the literature assume that whenever a particular substring is a constituent in some sentence, then it is automatically a constituent in all other sentences (whenever it does not overlap with previously chosen constituents) (van Zaanen, 2001; Clark, 2001; Adriaans et al., 2000). In reality, this is clearly not the case. A simple experiment on the WSJ10 corpus reveals that only 16 of the most frequent 1009 POS sequences (occurring 10 or more times in the sample) which are at least once constituents, are in fact always constituents. This assumption does not hold for ambiguous grammars in our class.

The approach we take to solve the smallest grammar problem can be extended to other classes of grammars. A similar formula can be built for grammars whose non-terminals have a one-to-one correspondence with congruence classes containing features of their language (Clark, 2010b).

4 Experiments and Discussion

4.1 Experiments on Artificial Data

We tested our system on 11 typical context-free languages and 9 artificial natural language grammars taken from 4 different sources (Stolcke, 1994; Langley and Stromsten, 2000; Adriaans et al., 2000; Solan et al., 2005). The 11 CFLs include 7 described by unambiguous grammars:

UC1: $a^n b^n$ **UC2**: $a^n b^n c^m d^m$ **UC3**: $a^n b^m$ $n \geq m$ **UC4**: $a^p b^q$, $p \neq q$ **UC5**: Palindromes over alphabet $\{a, b\}$ with a central marker **UC6**:Palindromes over alphabet $\{a, b\}$ without a central marker **UC7**: Lukasiewicz language ($S \rightarrow aSS|b$)

and 4 described by ambiguous grammars:

AC1: $|w|_a = |w|_b$ **AC2**: $2|w|_a = |w|_b$ **AC3**: Dyck language **AC4**: Regular expressions.

The 9 artificial natural language grammars are:

NL1: Grammar 'a', Table 2 in (Langley and Stromsten, 2000) **NL2**: Grammar 'b', Table 2 in (Langley and Stromsten, 2000) **NL3**: Lexical categories and constituency, pg 96 in (Stolcke, 1994) **NL4**: Recursive embedding of constituents, pg 97 in (Stolcke, 1994) **NL5**: Agreement, pg 98 in (Stolcke, 1994) **NL6**: Singular/plural NPs and number agreement, pg 99 in (Stolcke, 1994) **NL7**: Experiment 3.1 grammar in (Adriaans et al., 2000) **NL8**: Grammar in Table 10 (Adriaans et al., 2000) **NL9**: TA1 grammar in (Solan et al., 2005).

The quality of the learned model depends on its capacity to predict the correct structure (parse trees) on the one hand and to predict the correct sentence probabilities on the other (i.e. assigns a probability distribution close to the target one). To evaluate parse trees, we follow suggestions given by van Zaanen and Geertzen (2008) and use micro-precision and micro-recall over all the non-trivial brackets. We take the harmonic mean of these two values to obtain the Unlabelled brackets F_1 score (UF_1). The learned distribution can be evaluated using perplexity (when the target distribution is not known) or some similarity metric between distributions (when the target distribution is known). In our case, the target distribution is

Ex.	$\|\Sigma\|$	$\|N\|$	$\|P\|$
UC1	2	3	4
UC2	4	7	9
UC3	2	3	5
UC4	2	5	9
UC5	2	3	5
UC6	2	3	8
UC7	2	2	3
AC1	2	4	9
AC2	2	5	11
AC3	2	3	5
AC4	7	8	13
NL1	9	8	15
NL2	8	8	13
NL3	12	10	18
NL4	13	11	22
NL5	16	12	23
NL6	19	17	32
NL7	12	3	9
NL8	30	10	35
NL9	50	45	81

Table 1: Size of the alphabet, number of non-terminals and productions rules of the grammars.

		Relative Entropy		UF_1	
Ex.	$\|S\|$	COMINO	ADIOS	COMINO	ABL
UC1	10	**0.029**	1.876	**100**	**100**
UC2	50	**0.0**	1.799	**100**	**100**
UC5	10	**0.111**	7.706	**100**	**100**
UC7	10	**0.014**	1.257	**100**	27.86
AC1	50	**0.014**	4.526	**52.36**	35.51
AC2	50	**0.098**	6.139	**46.95**	14.25
AC3	50	**0.057**	1.934	**99.74**	47.48
AC4	100	**0.124**	1.727	**83.63**	14.58
NL7	100	**0.0**	0.124	**100**	**100**
NL1	100	**0.202**	1.646	**24.08**	**24.38**
NL2	200	**0.333**	0.963	**45.90**	**45.80**
NL3	100	**0.227**	1.491	36.34	**75.95**
NL5	100	**0.111**	1.692	**88.15**	79.16
NL6	400	**0.227**	**0.138**	36.28	**100**
UC3	100	**0.411**	0.864	61.13	**100**
UC4	100	**0.872**	2.480	42.84	**100**
UC6	100	1.449	**1.0**	**20.14**	8.36
NL4	500	**1.886**	2.918	**65.88**	52.87
NL8	1000	**1.496**	**1.531**	**57.77**	50.04
NL9	800	1.701	**1.227**	12.49	**28.53**

Table 2: Relative Entropy and UF_1 results of our system COMINO vs ADIOS and ABL respectively. Best results are highlighted, close results (i.e. with a difference of at most 0.1 for relative entropy and 1% for UF_1) are both highlighted

known. We chose relative entropy[2] as a good measure of distance between distributions.

Our UF_1 results over test sets of one thousand strings were compared to results obtained by ABL (van Zaanen, 2001), which is a system whose primary aim is that of finding good parse trees (rather than identifying the target language). Although ABL does not obtain state-of-the-art results on natural language corpora, it proved to be the best system (for which an implementation is readily available) for unsupervised parsing of sentences generated by artificial grammars. Results are shown in Table 1.

We calculated the relative entropy on a test set of one million strings generated from the target grammar. We compared our results with ADIOS (Solan et al., 2005), a system which obtains competitive results on language modelling (Waterfall et al., 2010) and whose primary aim is of correctly identifying the target language (rather than finding good parse trees). Results are shown in Table 1.

For the tests in the first section of Table 1 (i.e. above the first dashed line), our algorithm was capable of exactly identifying the structure of the target grammar. Notwithstanding this, the bracketing results for these tests did not always yield perfect scores. This happened whenever the target grammar was ambiguous, in which case the most probable parse trees of the target and learned grammar can be different, thus leading to incorrect bracketing. For the tests in the second section of Table 1 (i.e. between the two dashed lines), our algorithm was capable of exactly identifying the target language (but not the grammar). In all of these cases, the induced grammar was slightly smaller than the target one. For the remaining tests, our algorithm did not identify the target language. In fact, it always overgeneralised. The 3 typical CFLs UC3, UC4 and UC6 are not identified because they are not contained in our subclass of CFLs. Inspite of this, the relative entropy results obtained are still relatively good. Overall, it is fair to say that the results obtained by our system, for both language modelling and unsupervised parsing on artificial data, are competitive with the results obtained by other methods.

[2]The relative entropy (or Kullback-Leibler divergence) between a target distribution D and a hypothesized distribution D' is defined as $\sum_{w \in \Sigma^*} ln\Big(\frac{D(w)}{D'(w)}\Big) D(w)$. Add-one smoothing is used to solve the problem of zero probabilities.

4.2 Natural Language Experiments

We also experimented on natural language corpora. For unsupervised parsing, we tested our system on the WSJ10 corpus, using POS tagged sentences as input. Due to time efficiency, we changed the algorithm for finding congruence classes. Instead of always choosing the best possible merge w.r.t. the distance function, a distance threshold is set and all congruence classes whose distance is smaller than the threshold are merged. Also, we changed the distance function from L1-Distance to Pearson's χ^2 test.

In a first experiment (vaguely similar to the one done by Luque and López (2010)), we constructed the best possible SC-CFG consistent with the merges done in the first phase and assigned probabilities to this grammar using Inside-Outside. In other words, we ran the second phase of our system in a supervised fashion by using the treebank to decide which are the best congruence classes to choose as non-terminals. The CNF grammar we obtained from this experiment (COMINO-UBOUND) gives very good parsing results which outperform results from state-of-the-art systems DMV+CCM (Klein, 2004), U-DOP (Bod, 2006a), UML-DOP (Bod, 2006b) and Incremental (Seginer, 2007) as shown in Table 2. Moreover, the results obtained are very close to the best results one can ever hope to obtain from any CNF grammar on WSJ10 (CNF-UBOUND) (Klein, 2004). However, the grammar we obtain does not generalise enough and does not describe a good language model. In a second experiment, we ran the complete COMINO system. The grammar obtained from this experiment did not give competitive parsing results.

The first experiment shows that the merge decisions taken in the first phase do not hinder the possibility of finding a very good grammar for parsing. This means that the merge decisions taken by our system are good in general. Manual analysis on some of the merges taken confirms this. This experiment also shows that there exists a non-trivial PCFG in our restrictive class of grammars that is capable of achieving very good parsing results. This is a positive sign for the question of how adequate SC-PCFGs are for modelling natural languages. However, the real test remains that of finding SC-PCFGs that generate good bracketings *and* good language models. The second experiment shows that the second phase of our al-

Model	UP	UR	UF_1
State-of-the-art			
DMV+CCM	69.3	88.0	77.6
U-DOP	70.8	88.2	78.5
UML-DOP	-	-	82.9
Incremental	75.6	76.2	75.9
Upper bounds			
COMINO-UBOUND	75.8	96.9	85.1
CNF-UBOUND	78.8	100.0	88.1

Table 3: Parsing results on WSJ10. Note that *Incremental* is the only system listed as state-of-the-art which parses from plain text and can generate non-binary trees

gorithm is not giving good results. This means that the smallest possible grammar might not be the best grammar for parsing. Therefore, other criteria alongside the grammar size are needed when choosing a grammar consistent with the merges.

4.3 Discussion and Future Work

In order to improve our system, we think that our algorithm has to take a less conservative merging strategy in the first phase. Although the merges being taken are mostly correct, our analysis shows that not enough merging is being done. The problematic case is that of taking merge decisions on (the many) infrequent long phrases. Although many logically deduced merges involve infrequent phrases and also help in increasing the frequency of some long phrases, this proved to be not enough to mitigate this problem. As for future work, we think that clustering techniques can be used to help solve this problem.

A problem faced by the system is that, in certain cases, the statistical evidence on which merge decisions are taken does not point to the intuitively expected merges. As an example, consider the two POS sequences "DT NN" and "DT JJ NN" in the WSJ corpus. Any linguist would agree that these sequences are substitutable (in fact, they have lots of local contexts in common). However, statistical evidence points otherwise, since their context distributions are not close enough. This happens because, in certain positions of a sentence, "DT NN" is far more likely to occur than "DT JJ NN" (w.r.t. the ratio of their total frequencies) and in other positions, "DT JJ NN" occurs more than expected. The following table shows the frequencies of these two POS sequences over the whole WSJ corpus and their frequencies in contexts (#,VBD) and (IN,#) (the symbol # represents the end or beginning of a sentence):

	Totals	(#,VBD)	(IN,#)
"DT NN"	42,222	1,034	2,123
"DT JJ NN"	15,243	152	1,119
Ratios	3.16	6.80	1.90

It is clear that the ratios do not match, thus leading to context distributions which are not close enough. Thus, this shows that basic sequences such as "DT NN" and "DT JJ NN", which linguists would group into the same concept NP, are statistically derived from different sub-concepts of NP. Our algorithm is finding these sub-concepts, but it is being evaluated on concepts (such as NP) found in the treebank (created by linguists).

From the experiments we did on artificial natural language grammars, it resulted that the target grammar was always slightly bigger than the learned grammar. Although in these cases we still managed to identify the target language or have a good relative entropy result, the bracketing results were in general not good. This and our second experiment on the WSJ10 corpus show that the smallest possible grammar might not be the best grammar for bracketing. To not rely solely on finding the smallest grammar, a bias can be added in favour of congruence classes which, according to constituency tests (like the Mutual Information criterion in Clark (2001)), are more likely to contain substrings that are constituents. This can be done by giving different weights to the congruence class variables in the formula and finding the solution with the smallest sum of weights of its true variables.

The use of POS tags as input can also have its problems. Although we solve the lexical sparsity problem with POS tags, at the same time we lose a lot of information. In certain cases, one POS sequence can include raw phrases which ideally are not grouped into the same congruence class. To mitigate this problem, we can use POS tags only for rare words and subdivide or ignore POS tags for frequent words such as determinants and prepositions. This will reduce the number of raw phrases represented by POS sequences whilst keeping lexical sparsity low.

5 Conclusion

We defined a new class of PCFGs that adequately models natural language syntax. We described a learning algorithm for this class which scales well to large examples and is even capable of learning from small samples. The grammars induced by this algorithm are compact and perform well on unsupervised parsing and language modelling of typical CFLs and artificial natural language grammars.

Acknowledgements

The authors acknowledge partial support by the Région des Pays de la Loire.

References

Pieter W. Adriaans, Marten Trautwein, and Marco Vervoort. Towards High Speed Grammar Induction on Large Text Corpora. In Václav Hlavác, Keith G. Jeffery, and Jirí Wiedermann, editors, *SOFSEM*, volume 1963 of *Lecture Notes in Computer Science*, pages 173–186. Springer, 2000.

Michel Berkelaar et al. lpSolve: Interface to Lp solve v. 5.5 to solve linear/integer programs. *R package version*, 5(4), 2008.

Luc Boasson and Géraud Sénizergues. NTS Languages Are Deterministic and Congruential. *J. Comput. Syst. Sci.*, 31(3):332–342, 1985.

Rens Bod. Unsupervised Parsing with U-DOP. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 85–92, Stroudsburg, PA, USA, 2006a. Association for Computational Linguistics.

Rens Bod. An All-Subtrees Approach to Unsupervised Parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 865–872. Association for Computational Linguistics, 2006b.

Alexander Clark. *Unsupervised Language Acquisition: Theory and Practice*. PhD thesis, University of Sussex, 2001.

Alexander Clark. Distributional Learning of Some Context-Free Languages with a Minimally Adequate Teacher. In Sempere and García (2010), pages 24–37.

Alexander Clark. Towards General Algorithms for Grammatical Inference. In Marcus Hutter, Frank Stephan, Vladimir Vovk, and Thomas Zeugmann, editors, *ALT*, volume 6331 of *Lecture Notes in Computer Science*, pages 11–30. Springer, 2010b.

Alexander Clark and Rémi Eyraud. Polynomial Identification in the Limit of Substitutable Context-free Languages. *Journal of Machine Learning Research*, 8:1725–1745, 2007.

Alexander Clark and Shalom Lappin. Unsupervised Learning and Grammar Induction. In Alexander Clark, Chris Fox, and Shalom Lappin, editors, *The Handbook of Computational Linguistics and Natural Language Processing*, pages 197–220. Wiley-Blackwell, 2010.

Alexander Clark, François Coste, and Laurent Miclet, editors. *Grammatical Inference: Algorithms and Applications, 9th International Colloquium, ICGI 2008, Saint-Malo, France, September 22-24, 2008, Proceedings*, volume 5278 of *Lecture Notes in Computer Science*, 2008. Springer.

Colin de la Higuera. Characteristic Sets for Polynomial Grammatical Inference. *Machine Learning*, 27(2):125–138, 1997.

Colin de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. 2010.

E. Mark Gold. Language Identification in the Limit. *Information and Control*, 10(5):447–474, 1967.

James Jay Horning. *A Study of Grammatical Inference*. PhD thesis, 1969.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. Bayesian Inference for PCFGs via Markov Chain Monte Carlo. In Candace L. Sidner, Tanja Schultz, Matthew Stone, and ChengXiang Zhai, editors, *HLT-NAACL*, pages 139–146. The Association for Computational Linguistics, 2007.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 2 edition, 2008.

Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The Approximability of Constraint Satisfaction Problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000.

Dan Klein. *The Unsupervised Learning of Natural Language Structure*. PhD thesis, Stanford University, 2004.

Kevin J. Lang, Barak A. Pearlmutter, and Rodney A. Price. Results of the Abbadingo One DFA Learning Competition and a New Evidence-Driven State Merging Algorithm. In Vasant Honavar and Giora Slutzki, editors, *ICGI*, volume 1433 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1998.

Pat Langley and Sean Stromsten. Learning Context-Free Grammars with a Simplicity Bias. In Ramon López de Mántaras and Enric Plaza, editors, *ECML*, volume 1810 of *Lecture Notes in Computer Science*, pages 220–228. Springer, 2000.

Karim Lari and Steve J. Young. The Estimation of Stochastic Context-Free Grammars using the Inside-Outside Algorithm. *Computer Speech & Language*, 4(1):35 – 56, 1990.

Franco M. Luque and Gabriel G. Infante López. Bounding the Maximal Parsing Performance of Non-Terminally Separated Grammars. In Sempere and García (2010), pages 135–147.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 2001.

Paola Merlo, Harry Bunt, and Joakim Nivre. Current Trends in Parsing Technology. In *Trends in Parsing Technology*, pages 1–17. Springer, 2010.

Georgios Petasis, Georgios Paliouras, Constantine D. Spyropoulos, and Constantine Halatsis. eg-GRIDS: Context-Free Grammatical Inference from Positive Examples Using Genetic Search. In Georgios Paliouras and Yasubumi Sakakibara, editors, *ICGI*, volume 3264 of *Lecture Notes in Computer Science*, pages 223–234. Springer, 2004.

Gisa Rauh. *Syntactic Categories: Their Identification and Description in Linguistic Theories*. Oxford Surveys in Syntax & Morphology No.7. OUP Oxford, 2010.

Yoav Seginer. Fast Unsupervised Incremental Parsing. In John A. Carroll, Antal van den Bosch, and Annie Zaenen, editors, *ACL*. The Association for Computational Linguistics, 2007.

José M. Sempere and Pedro García, editors. *Grammatical Inference: Theoretical Results and Applications, 10th International Colloquium, ICGI 2010, Valencia, Spain, September 13-16, 2010. Proceedings*, volume 6339 of *Lecture Notes in Computer Science*, 2010. Springer.

Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11629–11634, 2005.

Andreas Stolcke. *Bayesian learning of probabilistic language models*. PhD thesis, University of California, Berkeley, 1994.

Menno van Zaanen. *Bootstrapping Structure into Language: Alignment-Based Learning*. PhD thesis, University of Leeds, 2001.

Menno van Zaanen and Jeroen Geertzen. Problems with Evaluation of Unsupervised Empirical Grammatical Inference Systems. In Clark et al. (2008), pages 301–303.

Heidi R. Waterfall, Ben Sandbank, Luca Onnis, and Shimon Edelman. An Empirical Generative Framework for Computational Modeling of Language Acquisition. *Journal of Child Language*, 37:671–703, 6 2010.

Charles S. Wetherell. Probabilistic Languages: A Review and Some Open Questions. *ACM Comput. Surv.*, 12(4):361–379, 1980.

Charles Yang. Computational Models of Syntactic Acquisition. *Wiley Interdisciplinary Reviews: Cognitive Science*, 3(2):205–213, 2012.

Ryo Yoshinaka. Identification in the Limit of k, l-Substitutable Context-Free Languages. In Clark et al. (2008), pages 266–279.

Can characters reveal your native language? A language-independent approach to native language identification

Radu Tudor Ionescu[◇], **Marius Popescu**[◇], **Aoife Cahill**[†]

[◇]University of Bucharest
Department of Computer Science
14 Academiei, Bucharest, Romania
raducu.ionescu@gmail.com
popescunmarius@gmail.com

[†]Educational Testing Service
660 Rosedale Rd
Princeton, NJ 08541, USA
acahill@ets.org

Abstract

A common approach in text mining tasks such as text categorization, authorship identification or plagiarism detection is to rely on features like words, part-of-speech tags, stems, or some other high-level linguistic features. In this work, an approach that uses character n-grams as features is proposed for the task of native language identification. Instead of doing standard feature selection, the proposed approach combines several string kernels using multiple kernel learning. Kernel Ridge Regression and Kernel Discriminant Analysis are independently used in the learning stage. The empirical results obtained in all the experiments conducted in this work indicate that the proposed approach achieves state of the art performance in native language identification, reaching an accuracy that is 1.7% above the top scoring system of the 2013 NLI Shared Task. Furthermore, the proposed approach has an important advantage in that it is language independent and linguistic theory neutral. In the cross-corpus experiment, the proposed approach shows that it can also be topic independent, improving the state of the art system by 32.3%.

1 Introduction

Using words as basic units is natural in textual analysis tasks such as text categorization, authorship identification or plagiarism detection. Perhaps surprisingly, recent results indicate that methods handling the text at the character level can also be very effective (Lodhi et al., 2002; Sanderson and Guenter, 2006; Popescu and Dinu, 2007; Grozea et al., 2009; Popescu, 2011; Popescu and Grozea, 2012). By disregarding features of natural language such as words, phrases, or meaning, an approach that works at the character level has an important advantage in that it is language independent and linguistic theory neutral. This paper presents a state of the art machine learning system for native language identification that works at the character level. The proposed system is inspired by the system of Popescu and Ionescu (2013), but includes some variations and improvements. A major improvement is that several string kernels are combined via multiple kernel learning (Shawe-Taylor and Cristianini, 2004). Despite the fact that the (histogram) intersection kernel is very popular in computer vision (Maji et al., 2008; Vedaldi and Zisserman, 2010), it has never been used before in text mining. In this work, the intersection kernel is used for the first time in a text categorization task, alone and in combination with other kernels. The intersection kernel lies somewhere in the middle between the kernel that takes into account only the presence of n-grams and the kernel based on the frequency of n-grams (p-spectrum string kernel).

Two kernel classifiers are proposed for the learning task, namely Kernel Ridge Regression (KRR) and Kernel Discriminant Analysis (KDA). The KDA classifier is able to avoid the class-masking problem (Hastie and Tibshirani, 2003), which may often arise in the context of native language identification. Several experiments are conducted to evaluate the performance of the approach proposed in this work. While multiple kernel learning seems to produce a more robust system, the two kernel classifiers obtained mixed results in the experiments. Overall, the empirical results indicate that the approach proposed in this paper achieves state of the art performance in native language identification, while being both lan-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1363–1373,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

guage independent and linguistic theory neutral. Furthermore, the approach based on string kernels does not need any expert knowledge of words or phrases in the language.

The paper is organized as follows. Related work is presented in Section 2. Section 3 presents several similarity measures for strings, including string kernels and Local Rank Distance. The learning methods used in the experiments are described in Section 4. Section 5 presents details about the experiments. Finally, the conclusions are drawn in Section 6.

2 Related Work

2.1 Native Language Identification

The goal of automatic native language identification (NLI) is to determine the native language of a language learner, based on a piece of writing in a foreign language. This can provide useful information in forensic linguistic tasks (Estival et al., 2007) or could be used in an educational setting to provide contrastive feedback to language learners. Most research has focused on identifying the native language of English language learners, though there have been some efforts recently to identify the native language of writing in other languages (Malmasi and Dras, 2014).

In general most approaches to NLI have used multi-way classification with SVMs or similar models along with a range of linguistic features. The seminal paper by Koppel et al. (2005) introduced some of the best-performing features: character, word and part-of-speech n-grams along with features inspired by the work in the area of second-language acquisition such as spelling and grammatical errors. In 2013, Tetreault et al. (2013) organized the first shared task in the field. This allowed researchers to compare approaches for the first time on a specifically designed NLI corpus that was much larger than previously available data sets. In the shared task, 29 teams submitted results for the test set, and one of the most successful aspects of the competition was that it drew submissions from teams working in a variety of research fields. The submitted systems utilized a wide range of machine learning approaches, combined with several innovative feature contributions. The best performing system achieved an overall accuracy of 83.6% on the 11-way classification of the test set, although there was no significant difference between the top teams.

2.2 Methods that Work at the Character Level

In recent years, methods of handling text at the character level have demonstrated impressive performance levels in various text analysis tasks (Lodhi et al., 2002; Sanderson and Guenter, 2006; Popescu and Dinu, 2007; Grozea et al., 2009; Popescu, 2011; Popescu and Grozea, 2012). Lodhi et al. (2002) used string kernels for document categorization with very good results. String kernels were also successfully used in authorship identification (Sanderson and Guenter, 2006; Popescu and Dinu, 2007; Popescu and Grozea, 2012). For example, the system described in (Popescu and Grozea, 2012) ranked first in most problems and overall in the PAN 2012 Traditional Authorship Attribution tasks.

Using string kernels makes the corresponding learning method completely language independent, because the texts will be treated as sequences of symbols (strings). Methods working at the word level or above very often restrict their feature space according to theoretical or empirical principles. For instance, they select only features that reflect various types of spelling errors or only some type of words, such as function words. These features prove to be very effective for specific tasks, but it is possible that other good features also exist. String kernels embed the texts in a very large feature space, given by all the substrings of length p, and leave it to the learning algorithm to select important features for the specific task, by highly weighting these features. It is important to note that this approach is also linguistic theory neutral, since it disregards any features of natural language such as words, phrases, or meaning. On the other hand, a method that considers words as features cannot be completely language independent, since the definition of a word is necessarily language-specific. For example, a method that uses only function words as features is not completely language independent because it needs a list of function words which is specific to a language. When features such as part-of-speech tags are used, as in the work of Jarvis et al. (2013), the method relies on a part-of-speech tagger which might not be available (yet) for some languages. Furthermore, a way to segment a text into words is not an easy task for some languages, such as Chinese.

Character n-grams are used by some of the systems developed for native language identification.

In work where feature ablation results have been reported, the performance with only character n-gram features was modest compared to other types of features (Tetreault et al., 2012). Initially, most work limited the character features to unigrams, bigrams and trigrams, perhaps because longer n-grams were considered too expensive to compute or unlikely to improve performance. However, some of the top systems in the 2013 NLI Shared Task were based on longer character n-grams, up to 9-grams (Jarvis et al., 2013; Popescu and Ionescu, 2013). The results presented in this work are obtained using a range of 5–8 n-grams. Combining all 5–8 n-grams would generate millions of features, which are indeed expensive to compute and represent. The key to avoiding the computation of such a large number of features lies in using the dual representation provided by the string kernel. String kernel similarity matrices can be computed much faster and are extremely useful when the number of samples is much lower than the number of features.

3 Similarity Measures for Strings

3.1 String Kernels

The kernel function gives kernel methods the power to naturally handle input data that is not in the form of numerical vectors, e.g. strings. The kernel function captures the intuitive notion of similarity between objects in a specific domain and can be any function defined on the respective domain that is symmetric and positive definite. For strings, many such kernel functions exist with various applications in computational biology and computational linguistics (Shawe-Taylor and Cristianini, 2004).

Perhaps one of the most natural ways to measure the similarity of two strings is to count how many substrings of length p the two strings have in common. This gives rise to the p-spectrum kernel. Formally, for two strings over an alphabet Σ, $s, t \in \Sigma^*$, the p-spectrum kernel is defined as:

$$k_p(s,t) = \sum_{v \in \Sigma^p} \text{num}_v(s) \cdot \text{num}_v(t),$$

where $\text{num}_v(s)$ is the number of occurrences of string v as a substring in s.[1] The feature map defined by this kernel associates a vector of dimension $|\Sigma|^p$ containing the histogram of frequencies of all its substrings of length p (p-grams) with each string.

A variant of this kernel can be obtained if the embedding feature map is modified to associate a vector of dimension $|\Sigma|^p$ containing the presence bits (instead of frequencies) of all its substrings of length p with each string. Thus, the character p-grams presence bits kernel is obtained:

$$k_p^{0/1}(s,t) = \sum_{v \in \Sigma^p} \text{in}_v(s) \cdot \text{in}_v(t),$$

where $\text{in}_v(s)$ is 1 if string v occurs as a substring in s, and 0 otherwise.

In computer vision, the (histogram) intersection kernel has successfully been used for object class recognition from images (Maji et al., 2008; Vedaldi and Zisserman, 2010). In this paper, the intersection kernel is used for the first time as a kernel for strings. The intersection string kernel is defined as follows:

$$k_p^{\cap}(s,t) = \sum_{v \in \Sigma^p} \min\{\text{num}_v(s), \text{num}_v(t)\},$$

where $\text{num}_v(s)$ is the number of occurrences of string v as a substring in s.

For the p-spectrum kernel, the frequency of a p-gram has a very significant contribution to the kernel, since it considers the product of such frequencies. On the other hand, the frequency of a p-gram is completely disregarded in the p-grams presence bits kernel. The intersection kernel lies somewhere in the middle between the p-grams presence bits kernel and p-spectrum kernel, in the sense that the frequency of a p-gram has a moderate contribution to the intersection kernel. More precisely, the following inequality that describes the relation between the three kernels holds:

$$k_p^{0/1}(s,t) \leq k_p^{\cap}(s,t) \leq k_p(s,t).$$

What is actually more interesting is that the intersection kernel assigns a high score to a p-gram if it has a high frequency in both strings, since it considers the minimum of the two frequencies. The p-spectrum kernel assigns a high score even when the p-gram has a high frequency in only one of the two strings. Thus, the intersection kernel captures something about the correlation between the p-gram frequencies in the two strings, which may lead to a more sensitive similarity between strings.

[1] Note that the notion of substring requires contiguity. Shawe-Taylor and Cristianini (2004) discuss the ambiguity between the terms *substring* and *subsequence* across different domains: biology, computer science.

Normalized versions of these kernels ensure a fair comparison of strings of different lengths:

$$\hat{k}_p(s,t) = \frac{k_p(s,t)}{\sqrt{k_p(s,s) \cdot k_p(t,t)}},$$

$$\hat{k}_p^{0/1}(s,t) = \frac{k_p^{0/1}(s,t)}{\sqrt{k_p^{0/1}(s,s) \cdot k_p^{0/1}(t,t)}},$$

$$\hat{k}_p^{\cap}(s,t) = \frac{k_p^{\cap}(s,t)}{\sqrt{k_p^{\cap}(s,s) \cdot k_p^{\cap}(t,t)}}.$$

Taking into account p-grams of different length and summing up the corresponding kernels, new kernels, termed *blended spectrum kernels*, can be obtained.

The string kernel implicitly embeds the texts in a high dimensional feature space. Then, a kernel-based learning algorithm implicitly assigns a weight to each feature, thus selecting the features that are important for the discrimination task. For example, in the case of text categorization the learning algorithm enhances the features representing stems of content words (Lodhi et al., 2002), while in the case of authorship identification the same learning algorithm enhances the features representing function words (Popescu and Dinu, 2007).

3.2 Local Rank Distance

A recently introduced distance measure, termed Local Rank Distance (Ionescu, 2013), comes from the idea of better adapting rank distance (Dinu, 2003) to string data, in order to capture a better similarity between strings, such as DNA sequences or text. Local Rank Distance (LRD) has already shown promising results in computational biology (Ionescu, 2013) and native language identification (Popescu and Ionescu, 2013).

In order to describe LRD, the following notations are defined. Given a string x over an alphabet Σ, and a character $a \in \Sigma$, the length of x is denoted by $|x|$. Strings are considered to be indexed starting from position 1, that is $x = x[1]x[2]\cdots x[|x|]$. Moreover, $x[i:j]$ denotes its substring $x[i]x[i+1]\cdots x[j-1]$.

Local Rank Distance is inspired by rank distance (Dinu, 2003), the main differences being that it uses p-grams instead of single characters, and that it matches each p-gram in the first string with the nearest equal p-gram in the second string. Given a fixed integer $p \geq 1$, a threshold $m \geq 1$, and two strings x and y over Σ, the *Local Rank Distance* between x and y, denoted by $\Delta_{LRD}(x,y)$, is defined through the following algorithmic process. For each position i in x ($1 \leq i \leq |x|-p+1$), the algorithm searches for that position j in y ($1 \leq j \leq |y|-p+1$) such that $x[i:i+p] = y[j:j+p]$ and $|i-j|$ is minimized. If j exists and $|i-j| < m$, then the offset $|i-j|$ is added to the Local Rank Distance. Otherwise, the maximal offset m is added to the Local Rank Distance. An important remark is that LRD does not impose any mathematically developed global constraints, such as matching the i-th occurrence of a p-gram in x with the i-th occurrence of that same p-gram in y. Instead, it is focused on the local phenomenon, and tries to pair equal p-grams at a minimum offset. To ensure that LRD is a (symmetric) distance function, the algorithm also has to sum up the offsets obtained from the above process by exchanging x and y. LRD can be formally defined as follows.

Definition 1 *Let $x, y \in \Sigma^*$ be two strings, and let $p \geq 1$ and $m \geq 1$ be two fixed integer values. The Local Rank Distance between x and y is defined as:*

$$\Delta_{LRD}(x,y) = \Delta_{left}(x,y) + \Delta_{right}(x,y),$$

where $\Delta_{left}(x,y)$ and $\Delta_{right}(x,y)$ are defined as follows:

$$\Delta_{left}(x,y) = \sum_{i=1}^{|x|-p+1} \min\{|i-j| \text{ such that } 1 \leq j \leq |y|-p+1 \text{ and } x[i:i+p] = y[j:j+p]\} \cup \{m\},$$

$$\Delta_{right}(x,y) = \sum_{j=1}^{|y|-p+1} \min\{|j-i| \text{ such that } 1 \leq i \leq |x|-p+1 \text{ and } y[j:j+p] = x[i:i+p]\} \cup \{m\}.$$

Interestingly, the search for matching p-grams is limited within a window of fixed size. The size of this window is determined by the maximum offset parameter m. This parameter must be set a priori and should be proportional to the size of the alphabet, the p-grams, and to the lengths of the strings.

The following example offers a better understanding of how LRD actually works. LRD is computed between two strings using 2-grams.

Example 1 *Given two strings $x = abcaa$ and $y = cabca$, a fixed maximal offset $m = 3$, and*

a fixed size of p-grams $p = 2$, Δ_{left} and Δ_{right} are computed as follows:

$$\begin{aligned}\Delta_{left}(x, y) &= |1 - 2| + |2 - 3| \\ &\quad + |3 - 4| + 3 = 6, \\ \Delta_{right}(x, y) &= |1 - 3| + |2 - 1| \\ &\quad + |3 - 2| + |4 - 3| = 5.\end{aligned}$$

By summing up the two partial sums, Local Rank Distance is obtained

$$\Delta_{LRD}(x, y) = \Delta_{left}(x, y) + \Delta_{right}(x, y) = 11.$$

The maximum LRD value between two strings can be computed as the product between the maximum offset m and the number of pairs of compared p-grams. Thus, LRD can be normalized to a value in the $[0, 1]$ interval. By normalizing, LRD becomes a dissimilarity measure. LRD can be also used as a kernel, since kernel methods are based on similarity. The classical way to transform a distance or dissimilarity measure into a similarity measure is by using the Gaussian-like kernel (Shawe-Taylor and Cristianini, 2004):

$$\hat{k}_p^{LRD}(s, t) = e^{-\frac{\Delta_{LRD}(s,t)}{2\sigma^2}},$$

where s and t are two strings and p is the p-grams length. The parameter σ is usually chosen so that values of $\hat{k}(s, t)$ are well scaled. In the above equation, Δ_{LRD} is already normalized to a value in the $[0, 1]$ interval to ensure a fair comparison of strings of different length.

4 Learning Methods

Kernel-based learning algorithms work by embedding the data into a Hilbert feature space, and searching for linear relations in that space. The embedding is performed implicitly, that is by specifying the inner product between each pair of points rather than by giving their coordinates explicitly. More precisely, a kernel matrix that contains the pairwise similarities between every pair of training samples is used in the learning stage to assign a vector of weights to the training samples. Let α denote this weight vector. In the test stage, the pairwise similarities between a test sample x and all the training samples are computed. Then, the following binary classification function assigns a positive or a negative label to the test sample:

$$g(x) = \sum_{i=1}^{n} \alpha_i \cdot k(x, x_i),$$

where x is the test sample, n is the number of training samples, $X = \{x_1, x_2, ..., x_n\}$ is the set of training samples, k is a kernel function, and α_i is the weight assigned to the training sample x_i. In the primal form, the same binary classification function can be expressed as:

$$g(x) = \langle w, x \rangle,$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product, $x \in \mathbb{R}^m$ is the test sample represented as a vector of features, and $w \in \mathbb{R}^m$ is a vector of feature weights that can be computed as follows:

$$w = \sum_{i=1}^{n} \alpha_i \cdot x_i,$$

given that the kernel function k can be expressed as a scalar product between samples.

The advantage of using the dual representation induced by the kernel function becomes clear if the dimension of the feature space m is taken into consideration. Since string kernels are based on character n-grams, the feature space is indeed very high. For instance, using 5-grams based only on the 26 letters of the English alphabet will result in a feature space of $26^5 = 11,881,376$ features. However, in the experiments presented in this work the feature space includes 5-grams along with 6-grams, 7-grams and 8-grams. As long as the number of samples n is not greater than the number of features m, it is more efficient to use the dual representation given by the kernel matrix. This fact is also known as the *kernel trick* (Shawe-Taylor and Cristianini, 2004).

Various kernel methods differ in the way they learn to separate the samples. In the case of binary classification problems, kernel-based learning algorithms look for a discriminant function, a function that assigns $+1$ to examples belonging to one class and -1 to examples belonging to the other class. For the NLI experiments, two binary kernel classifiers are used, namely the SVM (Cortes and Vapnik, 1995), and the KRR. Support Vector Machines try to find the vector of weights that defines the hyperplane that maximally separates the images in the Hilbert space of the training examples

belonging to the two classes. Kernel Ridge Regression selects the vector of weights that simultaneously has small empirical error and small norm in the Reproducing Kernel Hilbert Space generated by the kernel function. More details about SVM and KRR can be found in (Shawe-Taylor and Cristianini, 2004). The important fact is that the above optimization problems are solved in such a way that the coordinates of the embedded points are not needed, only their pairwise inner products which in turn are given by the kernel function.

SVM and KRR produce binary classifiers, but native language identification is usually a multi-class classification problem. There are many approaches for combining binary classifiers to solve multi-class problems. Typically, the multi-class problem is broken down into multiple binary classification problems using common decomposing schemes such as: one-versus-all and one-versus-one. There are also kernel methods that take the multi-class nature of the problem directly into account, e.g. Kernel Discriminant Analysis. The KDA classifier is able to improve accuracy by avoiding the masking problem (Hastie and Tibshirani, 2003). In the case of multi-class native language identification, the masking problem may appear when non-native English speakers have acquired, as the second language, a different language to English. For example, an essay written in English produced by a French native speaker that is also proficient in German, could be identified as either French or German.

5 Experiments

5.1 Data Sets Description

In this paper, experiments are carried out on three datasets: a modified version of the ICLEv2 corpus (Granger et al., 2009), the ETS Corpus of Non-Native Written English, or TOEFL11 (Blanchard et al., 2013), and the TOEFL11-Big corpus as used by Tetreault et al. (2012). A summary of the corpora is given in Table 1.

Corpus	Languages	Documents
ICLE	7	770
TOEFL11	11	12,100
TOEFL11-Big	11	87,502

Table 1: Summary of corpora used in the experiments.

The ICLEv2 is a corpus of essays written by highly-proficient non-native college-level students of English. For many years this was the standard corpus used in the task of native language identification. However, the corpus was originally collected for the purpose of corpus linguistic investigations, and because of this contains some idiosyncrasies that make it problematic for the task of NLI (Brooke and Hirst, 2012). Therefore, a modified version of the corpus that has been normalized as much as possible for topic and character encoding (Tetreault et al., 2012) is used. This version of the corpus contains 110 essays each for 7 native languages: Bulgarian, Chinese, Czech, French, Japanese, Russian and Spanish.

The ETS Corpus of Non-Native Written English (TOEFL11) was first introduced by Tetreault et al. (2012) and extended for the 2013 Native Language Identification Shared Task (Tetreault et al., 2013). It was designed to overcome many of the shortcomings identified with using the ICLEv2 corpus for this task. The TOEFL11 corpus contains a balanced distribution of essays per prompt (topic) per native language. It also contains information about the language proficiency of each writer. The corpus contains essays written by speakers of the following 11 languages: Arabic, Chinese, French, German, Hindi, Italian, Japanese, Korean, Spanish, Telugu and Turkish. For the shared task, the 12,100 essays were split into 9,900 for training, 1,100 for development and 1,100 for testing.

Tetreault et al. (2012) present a corpus, TOEFL11-Big, to investigate the performance of their NLI system on a very large data set. This data set contains the same languages as TOEFL11, but with no overlap in content. It contains a total of over 87 thousand essays written to a total of 76 different prompts. The distribution of L1 per prompt is not as even as for TOEFL11, though all topics are represented for all L1s.

5.2 Parameter Tuning and Implementation Choices

In the string kernels approach proposed in this work, documents or essays from this corpus are treated as strings. Therefore, the notions of *string* or *document* is used interchangeably throughout this work. Because the approach works at the character level, there is no need to split the texts into words, or to do any NLP-specific preprocessing. The only editing done to the texts was the replacing of sequences of consecutive space characters

(space, tab, new line, and so on) with a single space character. This normalization was needed in order to prevent the artificial increase or decrease of the similarity between texts, as a result of different spacing. All uppercase letters were converted to the corresponding lowercase ones.

A series of preliminary experiments were conducted in order to select the best-performing learning method. In these experiments the string kernel was fixed to the p-spectrum normalized kernel of length 5 ($\hat{k}_5$), because the goal was to select the best learning method, and not to find the best kernel. The following learning methods were evaluated: one-versus-one SVM, one-versus-all SVM, one-versus-one KRR, one-versus-all KRR, and KDA. A 10-fold cross-validation procedure was carried out on the TOEFL11 training set to evaluate the classifiers. The preliminary results indicate that the one-versus-all KRR and the KDA classifiers produce the best results. Therefore, they are selected for the remaining experiments.

Another set of preliminary experiments were performed to determine the range of n-grams that gives the most accurate results on a 10-fold cross-validation procedure carried out on the TOEFL11 training set. All the n-grams in the range 2-10 were evaluated. Furthermore, experiments with different blended kernels were conducted to see whether combining n-grams of different lengths could improve the accuracy. The best results were obtained when all the n-grams with the length in the range 5-8 were used. Other authors (Bykh and Meurers, 2012; Popescu and Ionescu, 2013) also report better results by using n-grams with the length in a range, rather than using n-grams of fixed length. Consequently, the results reported in this work are based on blended string kernels based on 5-8 n-grams.

Some preliminary experiments were also performed to establish the type of kernel to be used, namely the blended p-spectrum kernel ($\hat{k}_{5-8}$), the blended p-grams presence bits kernel ($\hat{k}^{0/1}_{5-8}$), the blended p-grams intersection kernel ($\hat{k}^{\cap}_{5-8}$), or the kernel based on LRD ($\hat{k}^{LRD}_{5-8}$.). These different kernel representations are obtained from the same data. The idea of combining all these kernels is natural when one wants to improve the performance of a classifier. When multiple kernels are combined, the features are actually embedded in a higher-dimensional space. As a consequence, the search space of linear patterns grows, which helps the classifier to select a better discriminant function. The most natural way of combining two kernels is to sum them up. Summing up kernels or kernel matrices is equivalent to feature vector concatenation. Another option is to combine kernels by kernel alignment (Cristianini et al., 2001). Instead of simply summing kernels, kernel alignment assigns weights for each of the two kernels based on how well they are aligned with the ideal kernel YY' obtained from training labels. The kernels were evaluated alone and in various combinations. The best kernels are the blended p-grams presence bits kernel and the blended p-grams intersection kernel. The best kernel combinations include the blended p-grams presence bits kernel, the blended p-grams intersection kernel and the kernel based on LRD. Since the kernel based on LRD is slightly slower than the other string kernels, the kernel combinations that include it were only evaluated on the TOEFL11 corpus and on the ICLE corpus.

5.3 Experiment on TOEFL11 Corpus

This section describes the results on the TOEFL11 corpus. Thus, results for the 2013 *Closed* NLI Shared Task are also included. In the closed shared task the goal is to predict the native language of testing examples, restricted to learning only from the training and the development data. The additional information from *prompts* or the English language proficiency level were not used in the proposed approach.

The regularization parameters were tuned on the development set. In this case, the systems were trained on the entire training set. A 10-fold cross-validation (CV) procedure was done on the training and the development sets. The folds were provided along with the TOEFL11 corpus. Finally, the results of the proposed systems are also reported on the NLI Shared Task test set. For testing, the systems were trained on both the training set and the development set. The results are summarized in Table 2.

The results presented in Table 2 show that string kernels can reach state of the art accuracy levels for this task. Overall, it seems that KDA is able to obtain better results than KRR. The intersection kernel alone is able to obtain slightly better results than the presence bits kernel. The kernel based on LRD gives significantly lower accuracy rates, but it is able to improve the performance when it is

Method	Development	10-fold CV	Test
Ensemble model (Tetreault et al., 2012)	-	80.9%	-
KRR and string kernels (Popescu and Ionescu, 2013)	-	82.6%	82.7%
SVM and word features (Jarvis et al., 2013)	-	**84.5%**	83.6%
KRR and $\hat{k}^{0/1}_{5-8}$	85.4%	82.5%	82.0%
KRR and $\hat{k}^{\cap}_{5-8}$	84.9%	82.2%	82.6%
KRR and $\hat{k}^{LRD}_{5-8}$	78.7%	77.1%	77.5%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{LRD}_{5-8}$	85.7%	82.6%	82.7%
KRR and $\hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	84.9%	82.2%	82.0%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	85.5%	82.6%	82.5%
KRR and $a_1\hat{k}^{0/1}_{5-8} + a_2\hat{k}^{\cap}_{5-8}$	85.5%	82.6%	82.5%
KDA and $\hat{k}^{0/1}_{5-8}$	86.2%	83.6%	83.6%
KDA and $\hat{k}^{\cap}_{5-8}$	85.2%	83.5%	84.6%
KDA and $\hat{k}^{LRD}_{5-8}$	79.7%	78.5%	79.2%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{LRD}_{5-8}$	**87.1%**	84.0%	84.7%
KDA and $\hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	85.8%	83.4%	83.9%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	86.4%	84.1%	85.0%
KDA and $a_1\hat{k}^{0/1}_{5-8} + a_2\hat{k}^{\cap}_{5-8}$	86.5%	84.1%	**85.3%**
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	87.0%	84.1%	84.8%

Table 2: Accuracy rates on TOEFL11 corpus of various classification systems based on string kernels compared with other state of the art approaches. The best accuracy rates on each set of experiments are highlighted in bold. The weights a_1 and a_2 from the weighted sums of kernels are computed by kernel alignment.

combined with the blended p-grams presence bits kernel. In fact, most of the kernel combinations give better results than each of their components. The best kernel combination is that of the presence bits kernel and the intersection kernel. Results are quite similar when they are combined either by summing them up or by kernel alignment. The best performance on the test set (85.3%) is obtained by the system that combines these two kernels via kernel alignment and learns using KDA. This system is 1.7% better than the state of the art system of Jarvis et al. (2013) based on SVM and word features, this being the top scoring system in the NLI 2013 Shared Task. It is also 2.6% better than the state of the art system based on string kernels of Popescu and Ionescu (2013). On the cross validation procedure, there are three systems that reach the accuracy rate of 84.1%. All of them are based on KDA and various kernel combinations. The greatest accuracy rate of 84.1% reported for the cross validation procedure is 3.2% above the state of the art system of Tetreault et al. (2012) and 0.4% below the top scoring system of Jarvis et al. (2013). The empirical results obtained in this experiment demonstrate that the approach proposed in this paper can reach state of the art accuracy levels. It is worth mentioning that a significance test performed by the organizers of the NLI 2013 Shared Task showed that the top systems that participated in the competition are not essentially different. Further experiments on the ICLE corpus and on the TOEFL11-Big corpus are conducted to determine whether the approach proposed in this paper is significantly better than other state of the art approaches.

5.4 Experiment on ICLE Corpus

The results on the ICLE corpus using a 5-fold cross validation procedure are summarized in Table 3. To adequately compare the results with a state of the art system, the same 5-fold cross validation procedure used by Tetreault et al. (2012) was also used in this experiment. Table 3 shows that the results obtained by the presence bits kernel and by the intersection kernel are systematically better than the state of the art system of Tetreault et al. (2012). While both KRR and KDA produce accuracy rates that are better than the state of the art accuracy rate, it seems that KRR is slightly better in this experiment. Again, the idea of combining kernels seems to produce more robust systems. The best systems are based on combining the presence bits kernel either with the kernel based on LRD or the intersection kernel. Overall, the reported accuracy rates are higher than the state of the art accuracy rate. The best performance (91.3%) is achieved by the KRR classifier based on combining the presence bits kernel with

Method	5-fold CV
Ensemble model (Tetreault et al., 2012)	90.1%
KRR and $\hat{k}^{0/1}_{5-8}$	91.2%
KRR and $\hat{k}^{\cap}_{5-8}$	90.5%
KRR and $\hat{k}^{LRD}_{5-8}$	81.8%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{LRD}_{5-8}$	**91.3%**
KRR and $\hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	90.1%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	90.9%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	90.6%
KDA and $\hat{k}^{0/1}_{5-8}$	90.5%
KDA and $\hat{k}^{\cap}_{5-8}$	90.5%
KDA and $\hat{k}^{LRD}_{5-8}$	82.3%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{LRD}_{5-8}$	90.8%
KDA and $\hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	90.4%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	91.0%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	90.8%

Table 3: Accuracy rates on ICLE corpus of various classification systems based on string kernels compared with a state of the art approach. The accuracy rates are reported for the same 5-fold CV procedure as in (Tetreault et al., 2012). The best accuracy rate is highlighted in bold.

the kernel based on LRD. This represents an 1.2% improvement over the state of the art accuracy rate of Tetreault et al. (2012). Two more systems are able to obtain accuracy rates greater than 91.0%. These are the KRR classifier based on the presence bits kernel (91.2%) and the KDA classifier based on the sum of the presence bits kernel and the intersection kernel (91.0%). The overall results on the ICLE corpus show that the string kernels approach can reach state of the art accuracy levels. It is worth mentioning the purpose of this experiment was to use the same approach determined to work well in the TOEFL11 corpus. To serve this purpose, the range of n-grams was not tuned on this data set. Furthermore, other classifiers were not tested in this experiment. Nevertheless, better results can probably be obtained by adding these aspects into the equation.

5.5 Cross-corpus Experiment

In this experiment, various systems based on KRR or KDA are trained on the TOEFL11 corpus and tested on the TOEFL11-Big corpus. The kernel based on LRD was not included in this experiment since it is more computationally expensive. Therefore, only the presence bits kernel and the intersection kernel were evaluated on the TOEFL11-Big corpus. The results are summarized in Table 4. The same regularization parameters determined to work well on the TOEFL11 development set were used.

Method	Test
Ensemble model (Tetreault et al., 2012)	35.4%
KRR and $\hat{k}^{0/1}_{5-8}$	66.7%
KRR and $\hat{k}^{\cap}_{5-8}$	67.2%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	**67.7%**
KRR and $a_1\hat{k}^{0/1}_{5-8} + a_2\hat{k}^{\cap}_{5-8}$	**67.7%**
KDA and $\hat{k}^{0/1}_{5-8}$	65.6%
KDA and $\hat{k}^{\cap}_{5-8}$	65.7%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	66.2%
KDA and $a_1\hat{k}^{0/1}_{5-8} + a_2\hat{k}^{\cap}_{5-8}$	66.2%

Table 4: Accuracy rates on TOEFL11-Big corpus of various classification systems based on string kernels compared with a state of the art approach. The systems are trained on the TOEFL11 corpus and tested on the TOEFL11-Big corpus. The best accuracy rate is highlighted in bold. The weights a_1 and a_2 from the weighted sums of kernels are computed by kernel alignment.

The most interesting fact is that all the proposed systems are at least 30% better than the state of the art system. Considering that the TOEFL11-Big corpus contains 87 thousand samples, the 30% improvement is significant without any doubt. Diving into details, it can be observed that the results obtained by KRR are higher than those obtained by KDA. However, both methods perform very well compared to the state of the art. Again, kernel combinations are better than each of their individual kernels alone.

It is important to mention that the significant performance increase is not due to the learning method (KRR or KDA), but rather due to the string kernels that work at the character level. It is not only the case that string kernels are language independent, but for the same reasons they can also be topic independent. Since the topics (prompts) from TOEFL11 are different from the topics from TOEFL11-Big, it becomes clear that a method that uses words as features is strongly affected, since the distribution of words per topic can be completely different. But mistakes that reveal the native language can be captured by character n-grams that can appear more often even in different topics. The results indicate that this is also the case of the approach based on string kernels, which seems to be more robust to such topic variations of the data set. The best system has an accuracy rate that is 32.3% better than the state of

the art system of Tetreault et al. (2012). Overall, the empirical results indicate that the string kernels approach can achieve significantly better results than other state of the art approaches.

6 Conclusions

A language-independent approach to native language identification was presented in this paper. The system works at the character level, making the approach completely language independent and linguistic theory neutral. The results obtained in all the three experiments were very good. The best system presented in this work is based on combining the intersection and the presence string kernels by kernel alignment and on deciding the class label either with KDA or KRR. The best system is 1.7% above the top scoring system of the 2013 NLI Shared Task. Furthermore, it has an impressive generalization capacity, achieving results that are 30% higher than the state of the art method in the cross-corpus experiment.

Despite the fact that the approach based on string kernels performed so well, it remains to be further investigated why this is the case and why such a simple approach can compete with far more complex approaches that take words, lemmas, syntactic information, or even semantics into account. It seems that there are generalizations to the kinds of mistakes that certain non-native English speakers make that can be captured by n-grams of different lengths. Interestingly, using a range of n-grams generates a large number of features including (but not limited to) stop words, stems of content words, word suffixes, entire words, and even n-grams of short words. Rather than doing feature selection before the training step, which is the usual NLP approach, the kernel classifier selects the most relevant features during training. With enough training samples, the kernel classifier does a better job of selecting the right features from a very high feature space. This may be one reason for why the string kernel approach works so well. To gain additional insights into why this technique is working well, the features selected by the classifier as being more discriminating can be analyzed in future work. This analysis would also offer some information about localized language transfer effects, since the features used by the proposed model are n-grams of lengths 5 to 8. As mentioned before, the features captured by the model typically include stems, function words, word prefixes and suffixes, which have the potential to generalize over purely word-based features. These features would offer insights into two kinds of language transfer effects, namely word choice (lexical transfer) and morphological differences.

Acknowledgments

The authors would like to thank Beata Beigman Klebanov, Nitin Madnani and Xinhao Wang from ETS for their helpful comments and suggestions. The author also thank the anonymous reviewers for their valuable insights which lead to improvements in the presentation of this work.

References

Daniel Blanchard, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. TOEFL11: A Corpus of Non-Native English. Technical report, Educational Testing Service Research Report No. RR–13–24.

Julian Brooke and Graeme Hirst. 2012. Robust, Lexicalized Native Language Identification. *Proceedings of COLING 2012*, pages 391–408, December.

Serhiy Bykh and Detmar Meurers. 2012. Native Language Identification using Recurring n-grams – Investigating Abstraction and Domain Dependence. *Proceedings of COLING 2012*, pages 425–440, December.

Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. *Machine Learning*, 20(3):273–297.

Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz S. Kandola. 2001. On kernel-target alignment. *Proceedings of NIPS*, pages 367–373, December.

Liviu P. Dinu. 2003. On the classification and aggregation of hierarchies with different constitutive elements. *Fundamenta Informaticae*, 55(1):39–50.

Dominique Estival, Tanja Gaustad, Son-Bao Pham, Will Radford, and Ben Hutchinson. 2007. Author profiling for English emails. *Proceedings of PACLING*, pages 263–272.

Sylviane Granger, Estelle Dagneaux, and Fanny Meunier. 2009. *The International Corpus of Learner English: Handbook and CD-ROM, version 2*. Presses Universitaires de Louvain, Louvain-la-Neuve, Belgium.

Cristian Grozea, Christian Gehl, and Marius Popescu. 2009. ENCOPLOT: Pairwise Sequence Matching in Linear Time Applied to Plagiarism Detection. In *3rd PAN Workshop. Uncovering Plagiarism, Authorship, and Social Software Misuse*, page 10.

Trevor Hastie and Robert Tibshirani. 2003. *The Elements of Statistical Learning*. Springer, corrected edition, July.

Radu Tudor Ionescu. 2013. Local Rank Distance. *Proceedings of SYNASC*, pages 221–228.

Scott Jarvis, Yves Bestgen, and Steve Pepper. 2013. Maximizing classification accuracy in native language identification. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 111–118, June.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Automatically Determining an Anonymous Author's Native Language. *Proceedings of ISI*, pages 209–217.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

Subhransu Maji, Alexander C. Berg, and Jitendra Malik. 2008. Classification using intersection kernel support vector machines is efficient. *Proceedings of CVPR*.

Shervin Malmasi and Mark Dras. 2014. Chinese Native Language Identification. *Proceedings of EACL*, 2:95–99, April.

Marius Popescu and Liviu P. Dinu. 2007. Kernel methods and string kernels for authorship identification: The federalist papers case. *Proceedings of RANLP*, September.

Marius Popescu and Cristian Grozea. 2012. Kernel methods and string kernels for authorship analysis. *CLEF (Online Working Notes/Labs/Workshop)*, September.

Marius Popescu and Radu Tudor Ionescu. 2013. The Story of the Characters, the DNA and the Native Language. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 270–278, June.

Marius Popescu. 2011. Studying translationese at the character level. *Proceedings of RANLP*, pages 634–639, September.

Conrad Sanderson and Simon Guenter. 2006. Short text authorship attribution via sequence kernels, markov chains and author unmasking: An investigation. *Proceedings of EMNLP*, pages 482–491, July.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Joel Tetreault, Daniel Blanchard, Aoife Cahill, and Martin Chodorow. 2012. Native Tongues, Lost and Found: Resources and Empirical Evaluations in Native Language Identification. *Proceedings of COLING 2012*, pages 2585–2602, December.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 48–57, June.

Andrea Vedaldi and Andrew Zisserman. 2010. Efficient additive kernels via explicit feature maps. *Proceedings of CVPR*, pages 3539–3546.

Formalizing Word Sampling for Vocabulary Prediction as Graph-based Active Learning

Yo Ehara*
National Institute of Information and Communications Technology
ehara@nict.go.jp

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

Hidekazu Oiwa
Issei Sato
Hiroshi Nakagawa
The University of Tokyo
{oiwa,sato}@r.dl.itc.u-tokyo.ac.jp
nakagawa@dl.itc.u-tokyo.ac.jp

Abstract

Predicting vocabulary of second language learners is essential to support their language learning; however, because of the large size of language vocabularies, we cannot collect information on the entire vocabulary. For practical measurements, we need to sample a small portion of words from the entire vocabulary and predict the rest of the words. In this study, we propose a novel framework for this sampling method. Current methods rely on simple heuristic techniques involving inflexible manual tuning by educational experts. We formalize these heuristic techniques as a graph-based non-interactive active learning method as applied to a special graph. We show that by extending the graph, we can support additional functionality such as incorporating domain specificity and sampling from multiple corpora. In our experiments, we show that our extended methods outperform other methods in terms of vocabulary prediction accuracy when the number of samples is small.

1 Introduction

Predicting the vocabulary of second language learners is essential to support them when they are reading. Educational experts have been continuously studying methods for measuring the size of a learner's vocabulary, i.e., the number of words the learner knows, over the decades (Meara and Buxton, 1987; Laufer and Nation, 1999). Ehara et al. (2012) formalized a more fine-grained measurement task called *vocabulary prediction*. The goal of this task is to predict whether a learner knows a given word based on only a relatively small portion of his/her vocabulary. This vocabulary prediction task can be further used for predicting the readability of texts. By predicting vocabulary unknown to readers and showing the meaning of those specific words to readers, Ehara et al. (2013) showed that the number of documents that learners can read increases.

Word sampling is essential for vocabulary prediction. Because of the large size of language vocabularies, we usually cannot collect information on the entire vocabulary. For practical measurements, we inevitably need to sample a small portion of words from the entire vocabulary and then predict the rest. We refer to this sampling technique as word sampling.

Word sampling can greatly affect the performance of vocabulary prediction. For example, if we consider only short everyday general domain words such as "cat" and "dog" as samples, the rest of the vocabulary is difficult to predict since learners likely know most of these words. To more accurately measure a learner's vocabulary, we ideally must sample words that are representative of the entire set of words. More specifically, we wish to sample words such that if a learner knows these words, he/she is likely to know the rest of the words in the given vocabulary, and vice versa.

To our knowledge, however, all current studies have relied on a simple heuristic method. In this heuristic method, educational experts first somehow create groups of words with the aim that the words in a group are of similar difficulty for learn-

*The main body of this work was done when the first author was a Ph.D. candidate in the University of Tokyo and the paper was later greatly revised when the first author was a JSPS (Japan Society for the Promotion of Science) research fellow (PD) at National Institute of Informatics. See http://yoehara.com/ for details.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1374–1384,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

ers. To create groups of words, the experts typically make use of word frequencies and sometimes manually reclassify words based on experience. Next, a fixed number of words are randomly sampled from each group via a uniform distribution. We call this approach *heuristic word sampling*.

In this study, we propose a novel framework that formalizes word sampling as *non-interactive* graph-based active learning based on weighted graphs. In our approach, nodes of a graph correspond to words, whereas the edge weights show how similar the difficulty levels of a word pair are. Unlike *interactive* active learning algorithms used in the NLP community, which use expert annotators' human labels for sampling nodes, non-interactive active learning algorithms exclude expert annotators' human labels from the protocol (Ji and Han, 2012; Gu and Han, 2012). Given a weighted graph and using only its structure, without human labels, these algorithms sample nodes that are important for classification with algorithms called *label propagation*. Excluding annotators' human labels from the protocol is beneficial for educational purposes since learners can share the same set of sampled words via, for example, printed handouts.

Formalizing the current methods as non-interactive graph-based active learning enables us to extend the sampling methods with additional functionality that current methods cannot handle without applying burdensome manual heuristics because we can flexibly design the weighted graphs fed to the active learning algorithms. In our framework, this extension is achieved by extending the graph, namely, our framework can handle *domain specificity* and *multiple corpora*.

Domains are important when one wants to measure the vocabulary of learners. For example, consider measuring non-native English speakers taking computer science graduate courses. We may want to measure their English vocabulary with an emphasis on computer science rather than their general English vocabulary. However, such an extension is impossible via current methods, and thus it is desirable to sample algorithms to be able to handle domain specificity. Our framework can incorporate domain specificity between words in the form of edges between such words.

Handling multiple corpora is important when we cannot single out which corpus we should rely on. The current technique used by educational experts to handle multiple corpora is to heuristically integrate multiple frequency lists from multiple corpora into a single list of words; however, such manual integration is burdensome. Thus, automatic integration is desirable. Our framework converts multiple corpora into graphs, merges these graphs together, and then samples from the merged graph.

Our contributions as presented in this paper are summarized as follows:

1. We formalize word sampling for vocabulary prediction as graph-based active learning.

2. Based on this formalization, we can perform more flexible word sampling that can handle domain specificity and multiple corpora.

The remaining parts of this paper are organized as follows. In §2, we explain the problem setting in detail. We first explain how existing heuristic word sampling works and how it relies on the *cluster assumption* from the viewpoint of graphs. Then, we introduce existing graph-based non-interactive active learning methods. In §3, we show that the existing heuristic word sampling is merely a special case of a non-interactive active learning method (Gu and Han, 2012). Precisely, the existing sampling is identical to the case where a special graph called a "multi-complete graph" is fed to a non-interactive active learning method. Since this method can take any weighted graphs other than this special graph, this immediately leads to a way of devising new sampling methods by modifying graphs. §4 explains exactly how we can modify graphs for improving active learning. §5 evaluates the proposed method both quantitatively and qualitatively, and §6 concludes our paper.

2 Problem Setting

2.1 Heuristic Word Sampling

A simple vocabulary estimation technique introduced by educational experts is to use the frequency rank of words in a corpus based on the assumption that learners using words with similar frequency ranks have a similar vocabulary (Laufer and Nation, 1999). In accordance with this assumption, they first group words by frequency ranks in a corpus and then assume that words in each group have a similar vocabulary status. For example, they sampled words as follows:

1. Rank words by frequency in a corpus.

2. Group words with frequency ranks from 1 to $1,000$ as Level 1000, words with frequency ranks from $1,001$ to $2,000$ as Level 2000, and so on.

3. Take 18 samples from Level 1000, another 18 samples from Level 2000, and so on.

The rationale behind this method is to treat high-ranked and low-ranked words separately rather than sample words from the entire vocabulary. After sampling words, this sampling method can be used for various measurements; for example, Laufer and Nation (1999) used this method to estimate the size of the learners' vocabulary by simply adding $1,000 * \frac{\text{Correctly answered words}}{18}$ for each level.

2.2 Cluster Assumption

In the previous subsection, we noted that existing word sampling methods rely on the assumption that *words with similar frequency ranks are known to learners whose familiar words are similar each other*. This assumption is known as the cluster assumption in the field of graph studies (Zhou et al., 2004).

To further describe the cluster assumption, we first define graphs. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes (vertices) $\mathcal{V}$ and a set of edges $\mathcal{E}$. Here, each node has a *label*, and each edge has a *weight*. A label denotes the category of its corresponding node. For example, in binary classification, a label is taken from $\{+1, -1\}$. A weight is a real value; when the weight of an edge is large, we describe the edge as being heavy.

The cluster assumption is an assumption that *heavily connected nodes in a graph should have similar labels*. In other words, the cluster assumption states that weights of edges and labels of nodes should be consistent.

We explain how the cluster assumption relates to our task. In our application, each node corresponds to a word. Labels of the nodes in a graph denote the vocabulary of a learner. If he/she knows a word, the label of the node corresponding to the word is $+1$; if not, the label is -1. The cluster assumption in our application is that the heavier the edge, the higher the similarity between users familiar with the two words.

In this manner, existing word sampling methods implicitly assume cluster assumption. This is therefore the underlying approach for reducing the word sampling problem into graph-based active learning. Since graphs allow for more flexible modeling by changing the weights of edges, we expect that more flexible word sampling will be enabled by graph-based active learning.

2.3 Label Propagation

Since the graph-based active learning algorithms are based on *label propagation* algorithms, we will explain them first. Basically, given a weighted graph, label propagation algorithms classify their nodes in a weakly supervised manner. While the graph-based active learning algorithm that we are trying to use (Gu and Han, 2012) does not use label propagation algorithms' outputs directly, it is tuned to be used with a state-of-the-art label propagation method called Learning with Local and Global Consistency (LLGC) (Zhou et al., 2004).

Label propagation algorithms predict the labels of nodes from a few manually supervised labels and graph weights. To this end, label propagation algorithms follow the following steps. First, humans label a small subset of the nodes in the graph. This subset of nodes is called the set of *labeled nodes*, and the remaining nodes are called *unlabeled nodes*. Second, label propagation algorithms propagate labels to the unlabeled nodes based on edge weights. The rationale behind label propagation algorithms lies in cluster assumption; as label propagation algorithms assume that two nodes connected by a heavily weighted edge should have similar labels, more heavily weighted edges should propagate more labels.

We formalize Learning with Local and Global Consistency (LLGC) (Zhou et al., 2004), one of the state-of-the-art label propagation methods. Here, for simplicity, suppose that we want to perform binary classification of nodes. Let N be the total number of nodes in a graph. Then, we denote labels of each node by $\mathbf{y} \stackrel{\text{def}}{=} (y_1, \ldots, y_N)^\top$. For unlabeled nodes, y_i is set to 0. For labeled nodes, y_i is set to $+1$ if the learner knows a word, -1 if not. We also introduce a label propagation (LP) *score vector* $\mathbf{f} = (f_1, \ldots, f_N)^\top$. This LP score vector is the output of label propagation and is real-valued. To obtain the classification result from this real-valued LP score vector for an unlabeled node (word) i, the learner is predicted to know the word i if $f_i > 0$, and he/she is predicted to be unfamiliar with the word if $f_i \leq 0$.

Next, we formally define a normalized graph-Laplacian matrix, which is used for penalization based on the cluster assumption. Let an $N \times N$ -sized square matrix $\mathbf{W}$ be a *weighted adjacency matrix* of $\mathcal{G}$. $\mathbf{W}$ is symmetric and non-negative definite; its diagonal elements $\mathbf{W}_{i,i} = 0$ and all other elements are non-negative[1]. The graph Laplacian of a normalized graph, known as a *normalized graph Laplacian* matrix, is defined as $\mathbf{L}^{\text{norm}}_{\mathbf{W}} \stackrel{\text{def}}{=} \mathbf{I} - \mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}}$. Here, $\mathbf{D}_{\mathbf{W}}$ is defined as a diagonal matrix whose diagonal element is $(\mathbf{D}_{\mathbf{W}})_{i,i} \stackrel{\text{def}}{=} \sum_{j=1}^{|\mathcal{V}|} \mathbf{W}_{i,j}$, and $\mathbf{I}$ denotes the identity matrix of the appropriate size. Note that a normalized graph Laplacian $\mathbf{L}^{\text{norm}}_{\mathbf{W}}$ depends on the weighted adjacency matrix $\mathbf{W}$.

Then, LLGC can be formalized as a simple optimization problem as shown in Equation 1.

$$\min_{\mathbf{f}} \|\mathbf{f} - \mathbf{y}\|_2^2 + \mu \mathbf{f}^\top \mathbf{L}^{\text{norm}}_{\mathbf{W}} \mathbf{f} \qquad (1)$$

Equation 1 consists of two terms. Intuitively, the first term tries to make the LP score vector, the final output $\mathbf{f}$, as close as possible to the given labels $\mathbf{y}$. The second term is designed to meet the cluster assumption: it penalizes the case where two nodes with heavy edges have very different LP scores. $\mu > 0$ is the only hyper-parameter of LLGC: it determines how strong the penalization based on the cluster assumption should be. Thus, in total, Equation 1 outputs an LP score vector $\mathbf{f}$ considering both the labeled input $\mathbf{y}$ and the cluster assumption of the given graph $\mathbf{W}$: the heavier an edge, the closer the scores of the two nodes connected by the edge becomes.

2.4 Graph-based active learning algorithms

An important categorization of graph-based active learning for applications is whether it is *interactive* or *non-interactive*. Here, interactive approaches use human labels during the learning process; they present a node for humans to label, and based on this label, the algorithms compute the next node to be presented to the humans. Thus, in interactive algorithms, human labeling and computations of the next node must run concurrently.

Non-interactive algorithms do not use human labels during the learning process. Given the entire graph, these algorithms sample important nodes for label propagation algorithms. Here, important nodes are the ones that minimize estimated classification error of label propagation when the nodes are labeled. Note that, unlike active learning used in the NLP community, non-interactive active learning algorithms exclude expert annotators' human labels from the protocol. While they exclude expert annotators, they are still regarded as active learning methods in the machine learning community since they try to choose such nodes that are beneficial for classification (Ji and Han, 2012; Gu and Han, 2012).

For educational purposes, *non-interactive* algorithms are preferred over *interactive* algorithms. The main drawback of interactive algorithms is that they must run concurrently with the human labeling. For our applications, this means that the vocabulary tests for vocabulary prediction must always be computerized. In contrast, non-interactive algorithms allow us to have vocabulary tests printed in the form of handouts, so we focus on non-interactive algorithms throughout this paper.

Compared with interactive algorithm studies, such as Zhu et al. (2003), graph-based non-interactive active learning algorithms have been introduced in recent years. There has been a seminal paper on non-interactive algorithms (Ji and Han, 2012). We used Gu and Han's algorithm because it reports higher accuracy for many tasks with competitive computation times over Ji and Han's algorithm (Gu and Han, 2012).

These active learning methods share two basic rules although their objective functions are different. First, these methods tend to select globally important nodes, also known as *hubs*. A notable example of global importance is the number of edges. Second, these methods tend to avoid sampling nodes that are heavily connected to previously sampled nodes. This is due to *cluster assumption*, the assumption that similar nodes should have similar labels, which suggests that it is redundant to select nodes close to previously sampled nodes; the labels of such nodes should be reliably predicted from the previously sampled nodes.

Gu and Han's algorithm, which is the algorithm we used, also follows these rules. In this algorithm, when considering the k-th sample, for every node i in the current set of not-yet-chosen nodes, a score $score(k, i)$ is calculated, and the node with the highest score is chosen. First, the score is de-

[1]While all elements of a non-negative definite matrix are not necessarily non-negative, we define all elements of $\mathbf{W}$ as non-negative here, following the definition of Zhou et al. (2004).

signed to be large if the i-th node is globally important. In the algorithm, the global importance of a node is measured by an eigenvalue decomposition of the normalized graph-Laplacian, $\mathbf{L}^{\text{norm}}$. Transformed from the graph's adjacency matrix, this matrix stores the graph's global information. Second, the score is designed to be smaller if the i-th node is close to one of the previously sampled nodes.

Score $score\,(k,i)$ is defined as follows. We perform eigenvalue decomposition beforehand. $\mathbf{L}^{\text{norm}}_{\mathbf{W}} = \mathbf{U}\Lambda\mathbf{U}^{\top}$, $\mathbf{u}_i$ is the transpose of the i-th row of $\mathbf{U}$, and λ_i is its corresponding eigenvalue.

$$score\,(k,i) \stackrel{\text{def}}{=} \frac{\left(\mathbf{H}_k^{-1}\mathbf{u}_i\right)^{\top}\mathbf{\Lambda}^{-1}\left(\mathbf{H}_k^{-1}\mathbf{u}_i\right)}{1+\mathbf{u}_i^{\top}\mathbf{H}_k^{-1}\mathbf{u}_i} \quad (2)$$

In Equation 2, $\mathbf{H}_k$ preserves information of the previous $k-1$ samples. First, $\mathbf{H}_0$ is a diagonal matrix whose i-th diagonal element is defined as $\frac{1}{(\mu\lambda_i+1)^2-1}$ where μ is a hyper-parameter. $\mathbf{H}_0$ weighs the score of globally important nodes through the eigenvalue decomposition. Second, $\mathbf{H}_k$ is updated such that the scores of the nodes distant from the previously taken samples are higher. The precise update formula of $\mathbf{H}_k$ follows. i_{k+1} is the index of the node sampled at $k+1$-th round. For the derivation of this formula, see Gu and Han (2012).

$$\mathbf{H}_{k+1}^{-1} = \mathbf{H}_k^{-1} - \frac{\left(\mathbf{H}_k^{-1}\mathbf{u}_{i_{k+1}}\right)\left(\mathbf{H}_k^{-1}\mathbf{u}_{i_{k+1}}\right)^{\top}}{1+\mathbf{u}_{i_{k+1}}^{\top}\mathbf{H}_k^{-1}\mathbf{u}_{i_{k+1}}} \quad (3)$$

Hyper-parameter μ determines how strong the cluster assumption should be; the larger the value, the more strongly the algorithm avoids selecting nodes near previously selected samples over the graph. Note that μ is inherited from the LLGC[2] algorithm (Zhou et al., 2004), i.e., the label propagation algorithm that Gu and Han's algorithm is based on. From the optimization viewpoint, μ determines the degree of penalization.

Remember that the $score$ has nothing to do with the LP scores described in §2.3. $score$ is used to choose nodes used for training in the graph-based non-interactive active learning. LP scores are later used for classification by label propagation algorithms that use the chosen training nodes. Throughout this paper, when we mean LP scores, we explicitly write "LP scores". All the other scores mean $score$.

[2]Learning with Local and Global Consistency.

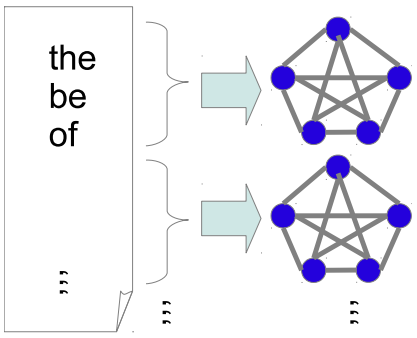


Figure 1: Converting frequency list into multiple-complete graph.

3 Formalizing heuristic word sampling as graph-based active learning

Figure 1 shows how to formalize a word frequency list into a multiple complete graph. The word frequency list is split into clusters, and each cluster forms a complete graph. Each node in a graph corresponds to a word. By gathering all the complete graphs, a multiple complete graph can be formed.

Multiple complete graph $\mathcal{G}_{T,n}$ is defined as a graph of T complete graphs, each of which consists of n nodes fully connected within the n nodes. An example of a multiple complete graph can be seen in Figure 2. We can define the $Tn \times Tn$ adjacency matrix for multiple complete graphs. $\mathbf{W}^{\text{complete}}_{\text{all}}$ is defined as follows:

$$\mathbf{W}^{\text{complete}}_{\text{all}} \stackrel{\text{def}}{=} \begin{pmatrix} \mathbf{W}^{\text{complete}} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{W}^{\text{complete}} \end{pmatrix} \quad (4)$$

$$\mathbf{W}^{\text{complete}} \stackrel{\text{def}}{=} \begin{pmatrix} 0 & 1 & \cdots & 1 & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ \vdots & 1 & \ddots & & \vdots \\ 1 & \vdots & & \ddots & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{pmatrix} \quad (5)$$

We can see that $\mathbf{W}^{\text{complete}}_{\text{all}}$ is a block-diagonal matrix where each block is a $n \times n$ matrix, $\mathbf{W}^{\text{complete}}$.

Heuristic word sampling can be rewritten into non-interactive active learning on graphs. Suppose there are T groups, each of which has n words, and we want to sample n_0 words from each. In

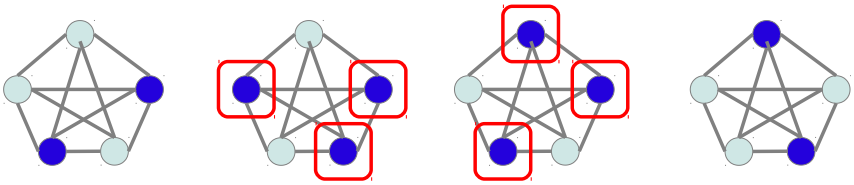


Figure 2: Example of multi-complete graph, where Theorem 3.1 holds true. Here, $T = 4$, $n = 5$, and $k = 10$; 10 light blue (light) nodes have already been sampled, and 10 blue (dark) nodes remain; the 11-th node is sampled uniformly randomly from the nodes within the red rectangles.

heuristic word sampling, for each group from T groups, n_0 words are sampled from the n words in the group uniformly randomly. Thus, there are Tn_0 words in total.

Since heuristic word sampling takes a node from each of the T groups, T concurrent sampling processes are involved. For simplicity, we further express the same sampling using only one sampling process from the entire graph as follows:

- For every round, we sample words uniformly randomly from the remaining words of the groups where the number of samples selected in previous rounds is least.

Figure 2 shows an example of this sampling process. Here, the second and third groups from the left are the groups in which the number of previously selected nodes is the least. This is because they have only two previously selected nodes, while the others have three. Thus, in the figure, the remaining words of the groups are the nodes with red rectangles. Randomly sampling one node from the nodes with red rectangles means sampling a node from the second or third group. We call the set of nodes in a graph from which samples will be taken in the next round a *seed pool*. Thus, in Figure 2, the set of nodes with red rectangles is the seed pool. Nodes that have already been sampled are taken out of the current seed pool.

Next, we more formally explain the seed pool concept. We start sampling nodes from a multiple complete graph via the algorithm presented by Gu and Han. The initial seed pool is set to all nodes in the graph, i.e., $\mathcal{V}$. We sample one node in each round; thus, $k \leq |\mathcal{V}|$ nodes are selected by the k-th round. Let $t \leq T$ be the index of the complete graph in the multiple complete graph. Then, the following theorem holds with ϵ being a small positive value that substitutes the 0 eigenvalues in the eigen decomposition.

Theorem 3.1 *Let* $0 < \epsilon < 1$ *and* $n \in \{2, 3, 4, \ldots\}$. *Then, among* T *complete graphs,* $k \mod T$ *complete graphs have* $\lfloor \frac{k}{T} \rfloor + 1$ *samples, and the remaining graphs have* $\lfloor \frac{k}{T} \rfloor$ *samples*[3]. *Moreover, the* $(k+1)$*-th sample is taken uniformly randomly from the remaining complete graphs.*

In Theorem 3.1, $\epsilon > 0$ is a substitute for the 0 eigenvalue of $\mathbf{L_W}$ [4]. Since ϵ is a substitute for the 0 eigenvalue, it is rational to assume $1 > \epsilon$. Also, remember that n is the number of nodes in one complete graph. The algorithm stops when $k = Tn_0 + 1$, i.e., at the $Tn_0 + 1$-th round when there are no remaining nodes to sample. Figure 2 shows an example of Theorem 3.1.

A proof of this theorem is presented in the supplementary material. Briefly, in a multiple complete graph, the score of a node depends only on the complete graph or the cluster that the node belongs to. Thus, we only have to consider one complete graph in which k is the number of nodes that have been already chosen. Then, mathematical induction proves that, within one complete graph, all the not-yet-chosen nodes have the same $score(k, i)$. Second, we have to show that the score always decreases by taking a sample, i.e., $score(k, i) > score(k+1, i)$. By a long but straightforward calculation, we can express $score(k, i)$ by using only μ, ϵ, n, and k. Then, by substituting the formula to $score(k, i)$, we obtain $score(k, i) - score(k+1, i) > 0$.

4 Extending Graphs

In the previous section, we explained how to formalize heuristic word sampling as active learning on multiple complete graphs. This formaliza-

[3] Here, both k and T are non-negative integers. Thus, $k\%T$ denotes the remainder of the division of k by T, and $\lfloor \frac{k}{T} \rfloor$ is the quotient of the division.

[4] In Gu and Han's algorithm, they substitute the 0 eigenvalue with a small positive value ϵ, and they set $\epsilon = 10^{-6}$.

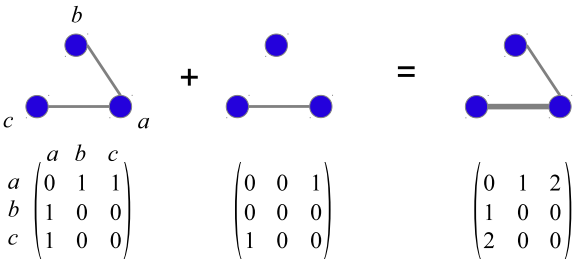


Figure 3: Example of merging two graphs.

tion can lead to better active learning by extending these graphs. In this section, we describe such graph extensions.

We extend graphs by *merging* graphs. Figure 3 shows how to merge graphs. We define "merging" two weighted graphs as creating a weighted graph whose adjacency matrix is the sum of the two adjacency matrices of the two weighted graphs. This suggests that an edge of the merged graph is simply the sum of the corresponding edges of the two weighted graphs.

The merged graph is expected to inherit the characteristics of its original graphs. Thus, applying graph-based active learning to the merged graph is expected to sample nodes in accordance with the characteristics of its original graphs. For example, if we merge a graph representing domain-specific relations and a multiple complete graph representing difficulty grouping of words, active learning from the resulting merged graph is expected to sample words considering both domain specificity and difficulty grouping of words.

For another example, suppose we merge two multiple complete graphs created from frequency lists from two different corpora. Then, active learning from the resulting merged graph is expected to sample words taking into account frequency lists from both corpora.

5 Evaluation

We evaluate our proposed method both quantitatively and qualitatively. In the quantitative evaluation, we measure the prediction accuracy of graphs. Note that the heuristic word sampling method is identical to using Gu and Han's algorithm with a multiple complete graph; however, our proposed graphs have enriched relations between words. In the qualitative evaluation, we explain in detail what words are appropriate as training examples for vocabulary prediction by presenting sampled examples.

5.1 Quantitative evaluation

To evaluate the accuracy of vocabulary prediction, we used the dataset that Ehara et al. (2010) and Ehara et al. (2012) used. This dataset was gleaned from questionnaires answered by 15 English as a second language (ESL) learners. Every learner was asked to answer how well he/she knew 11,999 English words. The data was collected in January 2009. One learner was unpaid, whereas the other 15 learners were paid. We used the data from the 15 paid learners since the data from the unpaid learner was noisy. Most of the learners were native Japanese speakers and graduate students. Because most of the learners in this dataset were native Japanese speakers, words from SVL 12,000 (SPACE ALC Inc., 1998) were used for the learners in this dataset. Note that SVL 12,000 is a collection of 12,000 words that are deemed important for Japanese learners of English, as judged by native English teachers.

Next, we required frequency lists for the words that appeared in the dataset. To create frequency lists, lemmatization is important because the number of word types depends on the method used to lemmatize the words. Note that in the field of vocabulary measurement, lemmatization is mainly performed by ignoring conjugation (Nation and Beglar, 2007). Lemmatizing the dataset resulted in a word list of 8,463 words. We adjusted the size of the word list to a round 8,000 by removing 463 randomly chosen words. Note that all constituent words were labeled by the 15 ESL learners.

We created the following four graphs by spanning edges among the $8,000$ words.

BNC multi-complete This graph corresponds to heuristic word sampling and served as our baseline. It is a multiple complete graph comprising eight complete graphs, each of which consisted of 1,000 words based on the sorted frequency list from the British National Corpus (BNC). We chose the BNC because the method presented by Nation and Beglar was based on it (Nation and Beglar, 2007). Note that all edge weights are set to 1.

BNC+domain To form this graph, edges representing domain specificity are added to the "BNC multi-complete" graph. For domain specificity, we used domain information

from WordNet 3.0.[5] First, we extracted 102 domain-specific words under the "computer" domain among the 8,000 words and created a complete graph consisting of these domain-specific words. The edge weights of the complete graph were set to 1. Next, we simply *merged*[6] the complete graph consisting of the domain-specific words with the "BNC multi-complete" graph.

BNC+COCA In addition to the "BNC multi-complete" graph, edges based on another corpus, the Corpus of Contemporary American English (COCA), were introduced. We first created the COCA multi-complete graph, a multiple complete graph consisting of eight complete graphs, each of which consisted of 1,000 words based on the sorted frequency list using COCA. The edge weights of the COCA multi-complete graph were set to 1. Next, we merged the BNC multi-complete and COCA multi-complete graphs to form the "BNC + COCA graph".

BNC+domain+COCA This graph is the graph produced by merging the "BNC + domain" and "BNC + COCA" graphs.

Note that our experiment setting differed from the usual label propagation setting used for semi-supervised learning because the purpose of our task differed. In the usual label propagation setting, the "test" nodes (data) are prepared separately from the training nodes to determine how accurately the algorithm can classify *forthcoming* or *unseen* nodes. However, in our setting, there were no such forthcoming words. Of course, there will always be words that do not emerge, even in a large corpus; however, such rare words are too difficult for language learners to identify, and many are proper nouns, which are not helpful for measuring the vocabulary of second language learners.

Therefore, our focus here is to measure how well the learners know a fixed set of words, that is, the given 8,000 words. Even if an algorithm can achieve high accuracy for words outside this fixed set, we have no way of evaluating it using the pooled annotations. Here, we want to measure, from a fixed number of samples (e.g., 50), how accurately an algorithm can predict a learner's vocabulary for the entire 8,000 words. Thus, we define accuracy to be the number of words that each algorithm finds correctly divided by the vocabulary size. We set hyper-parameter μ to 0.01 as Gu and Han (2012) did. Note that this hyper-parameter is reportedly not sensitive to accuracy (Zhou et al., 2011).

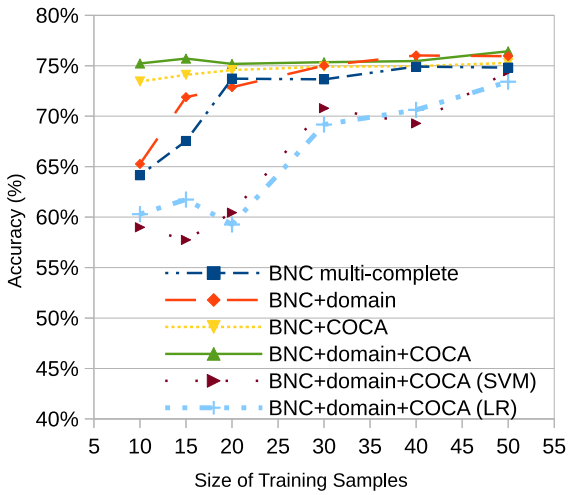


Figure 4: Results of our quantitative experiments. Vertical axis denotes accuracy, and horizontal axis shows number of samples, i.e., training words.

Figure 4 and Table 1 show the results of the experiment over the different datasets. The vertical axis in the figure denotes accuracy, whereas the horizontal axis denotes the number of samples, i.e., training words. Note that the accuracy is averaged over 15 learners and that LLGC is used for classification unless otherwise specified. For example, "BNC multi-complete" indicates that samples taken from the BNC multi-complete graph are used for training, and LLGC is used for classification. Note that "BNC + domain + COCA (SVM)" uses a support vector machine (SVM) for classification, and "BNC + domain + COCA (LR)" uses logistic regression (LR) for classification. Among many supervised machine learning methods, we chose SVM and LR because SVM is widely used in the NLP community, and LR was used for theoretical reasons (Ehara et al., 2012; Ehara et al., 2013).

SVM and LR require features of a word for classification while LLGC requires a weighted graph of words. Since the graph "BNC+domain+COCA" is made from three features, namely the word frequencies of BNC

[5]We used the NLTK toolkit http://nltk.org/ to extract the domain information.

[6]Definition of how to *merge* two graphs is in §4.

Table 1: Results of our quantitative experiments. LLGC is used for classification unless otherwise specified. Bold letters indicate top accuracy. Asterisks (*) indicate that values are statistically significant against baseline, heuristic sampling, i.e., "BNC multi-complete" (using sign test $p < 0.01$).

	10	15	20	30	40	50
BNC multi-complete	64.15 (%)	67.54	73.73	73.66	74.92	74.82
BNC+domain	65.27	71.88	72.88	75.02	**76.03** *	75.95
BNC+COCA	73.45	74.10	74.57	74.90	74.96	75.29
BNC+domain+COCA	**75.23** *	**75.71** *	**75.18** *	**75.35** *	75.47	**76.44** *
BNC+domain+COCA (SVM)	58.99	57.74	60.44	70.79	69.29	74.46
BNC+domain+COCA (LR)	60.29	61.74	59.27	69.17	70.63	73.42

and COCA corpora and whether a word is in the computer domain, we used these features for the features of SVM and LR in this experiment for a fair comparison. When using word frequencies for features, we used the logarithm of raw frequencies since it is reported to work well (Ehara et al., 2013). SVM and LR are also known to heavily depend on a hyper-parameter called C, which determines the strength of regularization. We tried $C = 0.1, 0.5, 1.0, 2.0, 5.0, 10.0, 20.0, 50.0,$ and 100.0 for each of SVM and LR where the size of training data is 50 and chose the C value that performs best. As a result, we set $C = 5.0$ for SVM and $C = 50.0$ for LR. Note that this setting is advantageous for SVM and LR compared to LLGC because the hyper-parameters of SVM and LR are tuned while LLGC's hyper-parameter remains untuned. For the implementation of SVM and LR, we used the "scikit-learn" package in Python [7].

We first observed that our proposed methods constantly outperform the baseline, heuristic word sampling, i.e., "BNC multi-complete" in Table 1. This indicates that we successfully obtained better accuracy by formalizing heuristic word sampling as active learning and extending graphs. In Table 1, the accuracy of the top-ranked methods (shown using **bold** letters) is statistically significantly better than the accuracy of "BNC multi-complete" (using the sign test $p < 0.01$).

We then observed that "BNC multi-complete" and "BNC + domain" show competitive accuracy with sample sizes from 10 to 20; furthermore, "BNC + domain" is slightly better than "BNC multi-complete" with sample sizes ranging from 30 to 50 (statistically significant $p < 0.01$ using sign test). Next, we note that there is a trade-off between domain and word frequency when choosing samples. More specifically, if we select too many words from the domain, the measurement of the general English ability of learners can be inaccurate; conversely, if we select too many words from the corpus-based word frequency list, while the general English ability of learners is accurately measured, we may obtain no information on the learner's vocabulary for the targeted domain. The competitive or slightly better accuracy of "BNC + domain" over "BNC multi-complete" shows that "BNC + domain" could successfully integrate domain information into the frequency-based groups without deteriorating measurements of general English ability.

We also observe that "BNC + COCA" greatly outperforms "BNC multi-complete" when the number of samples is 10. This shows that the integration of the two corpora, BNC and COCA (i.e., "BNC + COCA"), successfully increases the accuracy when there are only a small number of samples.

"BNC + domain + COCA" achieves the best accuracy of all the graphs except when the number of samples is 40. This indicates that the domain information and the information from the COCA corpus helped one another to improve the accuracy because "BNC + domain" and "BNC + COCA" introduce different types of domain information into "BNC multi-complete."

Finally, we observe that "BNC + domain + COCA (SVM)" and "BNC + domain + COCA (LR)" perform worse than LLGC over the same dataset for all sample sizes, particularly when the size of the training data is small. Since LLGC is a semi-supervised classifier while SVM and LR are not, SVM and LR perform poorly for small amounts of training data. This result shows that LLGC is appropriate for this task compared to SVM because, in this task, an increase in the size

[7] http://scikit-learn.org/stable/

Table 2: Computer-related samples in top 30 samples.

Name	Num. of Samples	Examples
BNC multi-complete	0	-
BNC+domain	5	input, client, field, background, register
BNC+COCA	0	-
BNC+domain +COCA	3	drive, client, command

of training data directly leads to an increased burden on the human learners.

5.2 Qualitative evaluation

In this subsection, we qualitatively evaluate our results to determine the types of nodes that are sampled when domain specificity is introduced. Specifically, we evaluate what words are selected as samples in the "BNC + domain" graph.

As noted above, in the "BNC + domain" graph, the computer science domain is introduced into "BNC multi-complete" to measure learners' vocabulary with a specific emphasis on the computer science domain. Thus, it is desirable that some words in the computer science domain are sampled from the "BNC + domain" graph; otherwise, we need to predict the learners' vocabulary for the computer science domain from general words rather than those in the computer science domain, which is extremely difficult.

Table 2 shows the number of words in the computer science domain sampled in the first 30 samples. Note that only "BNC + domain" and "BNC + domain + COCA" select samples from the computer science domain. This indicates that in the other two methods, to measure vocabulary with an emphasis on the computer science domain, we need to predict learners' vocabulary from the general words, which is almost impossible with only 30 samples. Furthermore, it is interesting to note that "BNC + domain" and "BNC + domain + COCA" select different samples from the computer science domain, except for the word "client," although originally the same computer science domain wordlist was introduced to both graphs.

Since "BNC + domain" achieves competitive or slightly better accuracy than "BNC multi-complete" in the quantitative analysis and the qualitative analysis, we conclude that our method can successfully introduce domain specificity into the sampling methodology without reducing accuracy.

6 Conclusion

In this study, we propose a novel sampling framework that measures the vocabulary of second language learners. We call existing sampling methods heuristic sampling. This approach to sampling ranks words from a single corpus by frequency and creates groups of 1,000 words. Next, tens of words are sampled from each group. This method assumes that the relative difficulty of all 1,000 words is the same.

In this paper, we introduce a novel sampling method by showing that the existing heuristic sampling approach is simply a special case of a graph-based active learning algorithm by Gu and Han (2012) applied to a special graph. We also propose a method to extend this graph to enable us to handle domain specificity of words and multiple corpora, which are difficult or impossible to handle using current methods.

We evaluate our method both quantitatively and qualitatively. In our quantitative evaluation, the proposed method achieves higher prediction accuracy compared with the current approach to vocabulary prediction. This suggests that our proposed method can successfully make use of domain specificity and multiple corpora for predicting vocabulary. In our qualitative evaluation, we examine the words sampled by our proposed method and observe that targeted domain-specific words are successfully sampled.

For our future work, because the graph used in this paper was constructed manually, we plan to automatically create a graph suitable for active learning and classification. There are several algorithms that create graphs from feature-based representations of words, but these have never been used for active learning of this task.

Acknowledgments

This work was supported by the Grant-in-Aid for JSPS Fellows (JSPS KAKENHI Grant Number 12J09575).

References

Yo Ehara, Nobuyuki Shimizu, Takashi Ninomiya, and Hiroshi Nakagawa. 2010. Personalized reading support for second-language web documents by collective intelligence. In *Proceedings of the 15th international conference on Intelligent user interfaces (IUI 2010)*, pages 51–60, Hong Kong, China. ACM.

Yo Ehara, Issei Sato, Hidekazu Oiwa, and Hiroshi Nakagawa. 2012. Mining words in the minds of second language learners: learner-specific word difficulty. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, Mumbai, India, December.

Yo Ehara, Nobuyuki Shimizu, Takashi Ninomiya, and Hiroshi Nakagawa. 2013. Personalized reading support for second-language web documents. *ACM Transactions on Intelligent Systems and Technology*, 4(2).

Quanquan Gu and Jiawei Han. 2012. Towards active learning on graphs: An error bound minimization approach. In *Proceedings of the IEEE International Conference on Data Mining (ICDM) 2012*.

Ming Ji and Jiawei Han. 2012. A variance minimization criterion to active learning on graphs. In *Proceedings of the 15th international conference on Artificial Intelligence and Statistics (AISTATS)*.

Batia Laufer and Paul Nation. 1999. A vocabulary-size test of controlled productive ability. *Language testing*, 16(1):33–51.

Paul Meara and Barbara Buxton. 1987. An alternative to multiple choice vocabulary tests. *Language Testing*, 4(2):142–154.

Paul Nation and David Beglar. 2007. A vocabulary size test. *The Language Teacher*, 31(7):9–13.

SPACE ALC Inc. 1998. Standard vocabulary list 12,000.

Dengyong Zhou, Oliver Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Proceedings in 18th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 321–328.

Xueyuan Zhou, Mikhail Belkin, and Nathan Srebro. 2011. An iterated graph laplacian approach for ranking on manifolds. In *Proceedings of 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 877–885.

Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. 2003. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*.

Language Transfer Hypotheses with Linear SVM Weights

Shervin Malmasi
Centre for Language Technology
Macquarie University
Sydney, NSW, Australia
shervin.malmasi@mq.edu.au

Mark Dras
Centre for Language Technology
Macquarie University
Sydney, NSW, Australia
mark.dras@mq.edu.au

Abstract

Language transfer, the characteristic second language usage patterns caused by native language interference, is investigated by Second Language Acquisition (SLA) researchers seeking to find overused and underused linguistic features. In this paper we develop and present a methodology for deriving ranked lists of such features. Using very large learner data, we show our method's ability to find relevant candidates using sophisticated linguistic features. To illustrate its applicability to SLA research, we formulate plausible language transfer hypotheses supported by current evidence. This is the first work to extend Native Language Identification to a broader linguistic interpretation of learner data and address the automatic extraction of underused features on a per-native language basis.

1 Introduction

It has been noted in the linguistics literature since the 1950s that speakers of particular languages have characteristic production patterns when writing in a second language. This language transfer phenomenon has been investigated independently in a number of fields from different perspectives, including qualitative research in Second Language Acquisition (SLA) and more recently though predictive computational models in NLP.

Motivated by the aim of improving foreign language teaching and learning, such analyses are often done manually in SLA, and are difficult to perform for large corpora. Smaller studies yield poor results due to the sample size, leading to extreme variability (Ellis, 2008). Recently, researchers have noted that NLP has the tools to use large amounts of data to automate this analysis, using complex feature types. This has motivated studies in Native Language Identification (NLI), a subtype of text classification where the goal is to determine the native language (L1) of an author using texts they have written in a second language (L2) (Tetreault et al., 2013).

Despite the good results in predicting L1s, few attempts have been made to interpret the features that distinguish L1s. This is partly because no methods for an SLA-oriented feature analysis have been proposed; most work focuses on testing feature types using standard machine learning tools.

The overarching contribution of this work is to develop a methodology that enables the transformation of the NLI paradigm into SLA applications that can be used to link these features to their underlying linguistic causes and explanations. These candidates can then be applied in other areas such as remedial SLA strategies or error detection.

2 Related Work

SLA research aims to find distributional differences in language use between L1s, often referred to as **overuse**, the extensive use of some linguistic structures, and **underuse**, the underutilization of particular structures, also known as *avoidance* (Gass and Selinker, 2008). While there have been some attempts in SLA to use computational approaches on small-scale data,[1] these still use fairly elementary techniques and have several shortcomings, including in the manual approaches to annotation and the computational artefacts derived from these.

Conversely, NLI work has focused on automatic learner L1 classification using Machine Learning with large-scale data and sophisticated linguistic features (Tetreault et al., 2012). Here, feature ranking could be performed with relevancy methods such as the F-score:

[1] E.g. Chen (2013), Lozanó and Mendikoetxea (2010) and Diéz-Bedmar and Papp (2008).

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1385–1390,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

$$F(j) \equiv \frac{\left(\bar{x}_j^{(+)} - \bar{x}_j\right)^2 + \left(\bar{x}_j^{(-)} - \bar{x}_j\right)^2}{\frac{1}{n_+-1}\sum_{i=1}^{n_+}\left(x_{i,j}^{(+)} - \bar{x}_j^{(+)}\right)^2 + \frac{1}{n_--1}\sum_{i=1}^{n_-}\left(x_{i,j}^{(-)} - \bar{x}_j^{(-)}\right)^2} \quad (1)$$

The F-score (Fisher score) measures the ratio between the intraclass and interclass variance in the values of feature j, where x represents the feature values in the negative and positive examples.[2] More discriminative features have higher scores.

Another alternative method is Information Gain (Yang and Pedersen, 1997). As defined in equation (2), it measures the entropy gain associated with feature t in assigning the class label c.

$$\begin{aligned} G(t) \;=\; & -\; \textstyle\sum_{i=1}^{m} \Pr(c_i) \log \Pr(c_i) \\ & +\; \Pr(t) \textstyle\sum_{i=1}^{m} \Pr(c_i|t) \log \Pr(c_i|t) \\ & +\; \Pr(\bar{t}) \textstyle\sum_{i=1}^{m} \Pr(c_i|\bar{t}) \log \Pr(c_i|\bar{t}) \end{aligned} \quad (2)$$

However, these methods are limited: they do not provide ranked lists per-L1 class, and more importantly, they do not explicitly capture underuse.

Among the efflorescence of NLI work, a new trend explored by Swanson and Charniak (2014) aims to extract lists of candidate language transfer features by comparing L2 data against the writer's L1 to find features where the L1 use is mirrored in L2 use. This allows the detection of obvious effects, but Jarvis and Crossley (2012) note (p. 183) that many transfer effects are "too complex" to observe in this manner. Moreover, this method is unable to detect underuse, is only suitable for syntactic features, and has only been applied to very small data (4,000 sentences) over three L1s. Addressing these issues is the focus of the present work.

3 Experimental Setup

3.1 Corpus

We use TOEFL11, the largest publicly available corpus of English L2 texts (Blanchard et al., 2013), containing 11 L1s with 1,100 texts each.[3]

3.2 Features

Adaptor grammar collocations Per Wong et al. (2012), we utilize an adaptor grammar to discover arbitrary length n-gram collocations. We explore both the pure part-of-speech (POS) n-grams as well as the more promising mixtures of POS and function words. We derive two adaptor grammars where each is associated with a different set of vocabulary: either pure POS or the mixture of POS and function words. We use the grammar proposed by Johnson (2010) for capturing topical collocations:

$Sentence \rightarrow Doc_j$	$j \in 1, \ldots, m$
$Doc_j \rightarrow \_j$	$j \in 1, \ldots, m$
$Doc_j \rightarrow Doc_j\ Topic_i$	$i \in 1, \ldots, t;$
	$j \in 1, \ldots, m$
$\underline{Topic_i} \rightarrow Words$	$i \in 1, \ldots, t$
$\overline{Words} \rightarrow Word$	
$Words \rightarrow Words\ Word$	
$Word \rightarrow w$	$w \in V_{pos};$
	$w \in V_{pos+fw}$

V_{pos} contains 119 distinct POS tags based on the Brown tagset and V_{pos+fw} is extended with 398 function words. The number of topics t is set to 50. The inference algorithm for the adaptor grammars are based on the Markov Chain Monte Carlo technique made available by Johnson (2010).[4]

Stanford dependencies We use Stanford dependencies as a syntactic feature: for each text we extract all the basic dependencies returned by the Stanford Parser (de Marneffe et al., 2006). We then generate all the variations for each of the dependencies (grammatical relations) by substituting each lemma with its corresponding POS tag. For instance, a grammatical relation of `det(knowledge, the)` yields the following variations: `det(NN, the)`, `det(knowledge, DT)`, and `det(NN, DT)`.

Lexical features Content and function words are also considered as two feature types related to learner's vocabulary and spelling.

3.3 Extracting Linear SVM Feature Weights

Using the extracted features, we train linear Support Vector Machine (SVM) models for each L1. We use a one-vs-rest approach to find features most relevant to each native language. L2-regularization is applied to remove noisy features and reduce the size of the candidate feature list. More specifically, we employ the LIBLINEAR SVM package (Fan et al., 2008)[5] as it is well-suited to text classification tasks with large numbers of features and texts as is the case here.

[2] See Chang and Lin (2008) for more details.

[3] Over 4 million tokens in 12,100 texts.

[4] http://web.science.mq.edu.au/%7Emjohnson/Software.htm

[5] http://www.csie.ntu.edu.tw/%7Ecjlin/liblinear/

In training the models for each feature, the SVM weight vector[6] is calculated according to (3):

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \quad (3)$$

After training, the positive and negative weights are split into two lists and ranked by weight. The positive weights represent overused features, while features whose absence (i.e. underuse) is indicative of an L1 class will have large negative weights. This yields two candidate language transfer feature lists per L1.

4 Results

We now turn to an analysis of the output from our system to illustrate its applicability for SLA research. Table 1 lists some elements from the underuse and overuse lists for various L1s. The lists are of different feature types. They have been chosen to demonstrate all feature types and also a variety of different languages. For reasons of space, only several of the top features are analysed here.

Hindi L1 writers are distinguished by certain function words including *hence*, *thus*, and *etc*, and a much higher usage rate of male pronouns. It has been observed in the literature (Sanyal, 2007, for example) that the English spoken in India still retains characteristics of the English that was spoken during the time of the Raj and the East India Company that have disappeared from other English varieties, so it sounds more formal to other speakers, or retains traces of an archaic business correspondence style; the features noted fit that pattern.

The second list includes content words overused by Arabic L1 learners. Analysis of content words here, and for other L1s in our data, reveals very frequent misspellings which are believed to be due to orthographic or phonetic influences (Tsur and Rappoport, 2007; Odlin, 1989). Since Arabic does not share orthography with English, we believe most of these are due to phonetics. Looking at items 1, 3 and 5 we can see a common pattern: the English letter *u* which has various phonetic realizations is being replaced by a vowel that more often represents that sound. Items 2 and 5 are also phonetically similar to the intended words.

For Spanish L1 authors we provide both underuse and overuse lists of syntactic dependencies. The top 3 overuse rules show the word *that* is very often used as the subject of verbs. This is almost certainly a consequence of the prominent syntactic role played by the Spanish word *que* which, depending on the context, is equivalent to the English words *whom*, *who*, *which*, and most commonly, *that*. The fourth rule shows they often use *this* as a determiner for plural nouns. A survey of the corpus reveals many such errors in texts of Spanish learners, e.g. *this actions* or *this emissions*. The fifth rule shows that the adjectival modifier of a plural noun is often being incorrectly pluralised to match the noun in number as would be required in Spanish, for example, *differents subjects*.

Turning to the underused features in Spanish L1 texts, we see that four related features rank highly, showing that *these* is not commonly used as a determiner for plural nouns and *which* is rarely used as a subject. The final feature shows that *no* is avoided as a determiner. This may be because while *no* mostly has the same role in Spanish as it does in English, it cannot be used as a determiner; *ningún* must be used instead. We hypothesize that this construction is being avoided as placing *no* before a noun in Spanish is ungrammatical. This example demonstrates that our two list methodology can not only help identify overused structures, but also uncovers the related constructs that are being underutilized at their expense.

The final list in Table 1 is of underused Adaptor Grammar patterns by Chinese learners. The first three features show that these writers significantly underuse determiners, here *an*, *other* and *these* before nouns. This is not unexpected since Chinese learners' difficulties with English articles are well known (Robertson, 2000). More interestingly, we find underuse of features like *even if* and *might*, along with others not listed here such as *could* VB[7] plus many other variants related to the subjunctive mood. One explanation is that linguistic differences between Chinese and English in expressing counterfactuals could cause them to avoid such constructions in L2 English. Previous research in this area has linked the absence of subjunctive linguistic structures in Chinese to different cognitive representations of the world and consequences for thinking counterfactually (Bloom, 2014), although this has been disputed (Au, 1983; Garbern Liu, 1985).

Adaptor Grammars also reveal frequent use of the "existential there"[8] in German L1 data while

[6] See Burges (1998) for a detailed explanation.

[7] e.g. *could be*, *could have*, *could go* and other variants

[8] e.g. *There is/are ...*, as opposed to the locative *there*.

Overuse			Underuse	
Hindi	Arabic	Spanish	Spanish	Chinese
#2: thus	#2: anderstand	#1: nsubj(VBP,that)	#2: det(NNS,these)	#12: *an* NN
#4: hence	#4: mony	#2: nsubj(VBZ,that)	#3: nsubj(VBZ,which)	#16: *other* NN
#22: his	#6: besy	#3: nsubj(VB,that)	#6: nsubj(VB,which)	#18: *these* NNS
#30: etc	#15: diffrent	#4: det(NNS,this)	#7: nsubj(VBP,which)	#19: *even if*
#33:rather	#38: seccessful	#25: amod(NNS,differents)	#10: det(NN,no)	#68: *might*

Table 1: Example transfer candidates and rankings from the overuse/underuse lists for various L1s and features types, in order: Hindi function words, Arabic content words, Spanish dependencies (2) and Chinese Adaptor Grammars.

English	**Spanish**	**English**	**Spanish**
diferent	diferente	conclution	conclusión
consecuence	consecuencia	desagree	Neg. affix *des-*
responsability	responsabilidad	especific	específico
oportunity	oportunidad	necesary	necesario

Table 2: Highly ranked English misspellings of Spanish learners and their Spanish cognates.

they are highly underused in French L1 data. The literature supports our data: The German equivalent *es gibt* is common while French use is far more constrained (Cappelle and Loock, 2013).

Lexical analysis also revealed Spanish–English orthographic transfer, listed in Table 2. This list includes many cognates, in contrast with the Arabic L1 data where most misspellings were phonetic in nature.

We also observe other patterns which remain unexplained. For instance, Chinese, Japanese and Korean speakers make excessive use of phrases such as *however*, *first* and *second*. One possibility is that this relates to argumentation styles that are possibly influenced by cultural norms. More broadly, this effect could also be teaching rather than transfer related. For example, it may be case that a widely-used text book for learning English in Korea happens to overuse this construction.

Some recent findings from the 2013 NLI Shared Task found that L1 Hindi and Telugu learners of English had similar transfer effects and their writings were difficult to distinguish. It has been posited that this is likely due to shared culture and teaching environments (Malmasi et al., 2013).

Despite some clearcut instances of overuse,[9] more research is required to determine the causal factors. We hope to expand on this in future work using more data.

[9]More than half of the Korean scripts contained a sentence-initial *however*.

5 Discussion and Conclusion

Using the proposed methodology, we generated lists of linguistic features overused and underused by English learners of various L1 backgrounds. Through an analysis of the top items in these ranked lists, we demonstrated the high applicability of the output by formulating plausible language transfer hypotheses supported by current evidence. We also showcased the method's generalizability to numerous linguistic feature types.

Our method's output consists of two ranked lists of linguistic features: one for overuse and the other for underuse, something which had not been addressed by research to date. We also found Adaptor Grammar collocations to be highly informative for this task.

This work, an intersection of NLP, Machine Learning and SLA, illustrates how the various disciplines can complement each other by bringing together theoretical, experimental and computational issues. NLP provides accurate and automated tagging of large corpora with sophisticated features not available in corpus linguistics, e.g. with state-of-the-art dependency parsing. Sophisticated machine learning techniques then enable the processing of large quantities of data (thousands of times the size of manual studies) in a way that will let SLA researchers explore a variety of assumptions and theoretical analyses. And conversely, NLP can benefit from the long-term study and language acquisition insights from SLA.

In terms of NLI, this work is the first attempt to expand NLI to a broad linguistic interpretation of the data, including feature underuse. NLI systems achieve classification accuracies of over 80% on this 11-class task, leading to theoretical questions about the features that make them so effective. This work also has a backwards link in this regard by providing qualitative evidence about the underpinning linguistic theories that make NLI work.

The work presented here has a number of applications; chief among them is the development of tools for SLA researchers. This would enable them to not just provide new evidence for previous findings, but to also perform semi-automated data-driven generation of new and viable hypotheses. This, in turn, can help reduce expert effort and involvement in the process, particularly as such studies expand to more corpora and emerging language like Chinese (Malmasi and Dras, 2014b) and Arabic (Malmasi and Dras, 2014a).

The brief analysis included here represents only a tiny portion of what can be achieved with this methodology. We included but a few of the thousands of features revealed by this method; practical SLA tools based on this would have a great impact on current research.

In addition to language transfer hypotheses, such systems could also be applied to aid development of pedagogical material within a needs-based and data-driven approach. Once language use patterns are uncovered, they can be assessed for teachability and used to create tailored, L1-specific exercises and teaching material.

From the examples discussed in Section 4 these could include highly specific and targeted student exercises to improve spelling, expand vocabulary and enrich syntactic knowledge — all relative to their mother tongue. Such exercises can not only help beginners improve their fundamental skills and redress their errors but also assist advanced learners in moving closer to near-nativeness.

The extracted features and their weights could also be used to build statistical models for grammatical error detection (Leacock et al., 2014). Contrary to the norm of developing error checkers for native writers, such models could be specifically targeted towards learners or even particular L1–L2 pairs which could be useful in Computer-Assisted Language Learning (CALL) systems.

One limitation here is that our features may be corpus-dependent as they are all exam essays. This can be addressed by augmenting the data with new learner corpora, as they become available. While a strength here is that we compared each L1 against others, a paired comparison only against native texts can be insightful too.

There are several directions for future work. The first relates to clustering the data within the lists. Our intuition is that there might be coherent clusters of related features, with these clusters characterising typical errors or idiosyncrasies, that are predictive of a particular L1. As shown in our results, some features are highly related and may be caused by the same underlying transfer phenomena. For example, our list of overused syntactic constructs by Spanish learners includes three high ranking features related to the same transfer effect. The use of unsupervised learning methods such as Bayesian mixture models may be appropriate here. For parse features, tree kernels could help measure similarity between the trees and fragments (Collins and Duffy, 2001).

Another avenue is to implement weight-based ranking methods to further refine and re-rank the lists, potentially by incorporating the measures mentioned in Section 2 to assign weights to features. As the corpus we used includes learner proficiency metadata, it may also be possible to create proficiency-segregated models to find the features that characterise errors at each language proficiency level. Finally, the use of other linguistic features such as Context-free Grammar phrase structure rules or Tree Substitution Grammars could provide additional insights.

In addition to these further technical investigations, we see as a particularly useful direction the development of an SLA research tool to conduct a large SLA study with a wide range of experts. We believe that this study makes a contribution to this area and hope that it will motivate future work.

References

Terry Kit-Fong Au. 1983. Chinese and English counterfactuals: the Sapir-Whorf hypothesis revisited. *Cognition*, 15(1):155–187.

Daniel Blanchard, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. TOEFL11: A Corpus of Non-Native English. Technical report, Educational Testing Service.

Alfred H Bloom. 2014. *The linguistic shaping of thought: A study in the impact of language on thinking in China and the West*. Psychology Press.

Christopher JC Burges. 1998. A tutorial on Support Vector Machines for Pattern Recognition. *Data mining and knowledge discovery*, 2(2):121–167.

Bert Cappelle and Rudy Loock. 2013. Is there interference of usage constraints?: A frequency study of existential there is and its French equivalent il ya in translated vs. non-translated texts. *Target*, 25(2):252–275.

Yin-Wen Chang and Chih-Jen Lin. 2008. Feature ranking using linear svm. *Causation and Prediction Challenge Challenges in Machine Learning, Volume 2*, page 47.

Meilin Chen. 2013. Overuse or underuse: A corpus study of English phrasal verb use by Chinese, British and American university students. *International Journal of Corpus Linguistics*, 18(3).

Michael Collins and Nigel Duffy. 2001. Convolution Kernels for Natural Language. In *Advances in Neural Information Processing Systems*, pages 625–632.

Marie-Catherine de Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, pages 449–454, Genoa, Italy.

María Belén Diéz-Bedmar and Szilvia Papp. 2008. The use of the English article system by Chinese and Spanish learners. *Language and Computers*, 66(1):147–176.

Rod Ellis. 2008. *The Study of Second Language Acquisition, 2nd edition*. Oxford University Press, Oxford, UK.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Lisa Garbern Liu. 1985. Reasoning counterfactually in Chinese: Are there any obstacles? *Cognition*, 21(3):239–270.

Susan M. Gass and Larry Selinker. 2008. *Second Language Acquisition: An Introductory Course*. Routledge, New York.

Scott Jarvis and Scott Crossley, editors. 2012. *Approaching Language Transfer Through Text Classification: Explorations in the Detection-based Approach*. Multilingual Matters, Bristol, UK.

Mark Johnson. 2010. PCFGs, Topic Models, Adaptor Grammars and Learning Topical Collocations and the Structure of Proper Names. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1148–1157, Uppsala, Sweden, July. Association for Computational Linguistics.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2014. Automated grammatical error detection for language learners. *Synthesis Lectures on Human Language Technologies*, 7(1):1–170.

Cristobal Lozanó and Amaya Mendikoetxea. 2010. Interface conditions on postverbal subjects: A corpus study of L2 English. *Bilingualism: Language and Cognition*, 13(4):475–497.

Shervin Malmasi and Mark Dras. 2014a. Arabic Native Language Identification. In *Proceedings of the Arabic Natural Language Processing Workshop (co-located with EMNLP 2014)*, Doha, Qatar, October. Association for Computational Linguistics.

Shervin Malmasi and Mark Dras. 2014b. Chinese Native Language Identification. *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, April.

Shervin Malmasi, Sze-Meng Jojo Wong, and Mark Dras. 2013. NLI Shared Task 2013: MQ Submission. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 124–133, Atlanta, Georgia, June. Association for Computational Linguistics.

Terence Odlin. 1989. *Language Transfer: Cross-linguistic Influence in Language Learning*. Cambridge University Press, Cambridge, UK.

Daniel Robertson. 2000. Variability in the use of the English article system by Chinese learners of English. *Second Language Research*, 16(2):135–172.

Jyoti Sanyal. 2007. *Indlish: The Book for Every English-Speaking Indian*. Viva Books Private Limited.

Ben Swanson and Eugene Charniak. 2014. Data Driven Language Transfer Hypotheses. *EACL 2014*, page 169.

Joel Tetreault, Daniel Blanchard, Aoife Cahill, and Martin Chodorow. 2012. Native Tongues, Lost and Found: Resources and Empirical Evaluations in Native Language Identification. In *Proceedings of COLING 2012*, pages 2585–2602, Mumbai, India, December. The COLING 2012 Organizing Committee.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 48–57, Atlanta, Georgia, June. Association for Computational Linguistics.

Oren Tsur and Ari Rappoport. 2007. Using classifier features for studying the effect of native language on the choice of written second language words. In *Proc. Workshop on Cognitive Aspects of Computat. Language Acquisition*, pages 9–16.

Sze-Meng Jojo Wong, Mark Dras, and Mark Johnson. 2012. Exploring Adaptor Grammars for Native Language Identification. In *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, pages 699–709.

Yiming Yang and Jan O Pedersen. 1997. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420.

Predicting Dialect Variation in Immigrant Contexts Using Light Verb Constructions

A. Seza Doğruöz
Netherlands Institute for Advanced Study
Wassenaar, Netherlands
a.s.dogruoz@gmail.com

Preslav Nakov
Qatar Computing Research Institute
Tornado Tower
floor 10, P.O. Box 5825, Doha, Qatar
pnakov@qf.org.qa

Abstract

Languages spoken by immigrants change due to contact with the local languages. Capturing these changes is problematic for current language technologies, which are typically developed for speakers of the standard dialect only. Even when dialectal variants are available for such technologies, we still need to predict which dialect is being used. In this study, we distinguish between the immigrant and the standard dialect of Turkish by focusing on Light Verb Constructions. We experiment with a number of grammatical and contextual features, achieving over 84% accuracy (56% baseline).

1 Introduction

Human languages are in constant evolution, driven in part by contact with other languages (Uriel, 1953; Thomason, 2008). In immigrant contexts, bilingual and multilingual speakers act as agents of change by transmitting borrowed words and expressions across languages (Grosjean, 2014). Depending on social factors such as duration and intensity of contact with the local languages, large-scale spread of borrowed elements could lead to differences between the contact and non-contact dialects of the same language (Winford, 2005). For example, Spanish spoken by immigrants in USA sounds different in comparison to Spanish spoken in South America (Corvalán, 2003).

In this study, we focus on the immigrant dialect of Turkish as spoken in the Netherlands (NL-Turkish), which differs from Turkish spoken in Turkey (TR-Turkish). In contact situations, it is common for verbs to be borrowed across languages and integrated as nominal complements of Light Verb Constructions (LVCs) (Edwards and Gardner-Chloros, 2007; Butt, 2010). NL-Turkish LVCs are changing due to Dutch influence (Doğruöz and Backus, 2007; Doğruöz and Backus, 2009; Doğruöz and Gries, 2012). However, assessing Dutch influence is not always easy since NL-Turkish LVCs still co-exist with the TR-Turkish LVCs. This study aims to automatically identify the features that can distinguish between NL-Turkish and TR-Turkish LVCs.

Our study would benefit Machine Translation systems targeting dialectal variation. It differs from studies concerning the well-established dialectal variations of Arabic, e.g., Levantine, Gulf, Egyptian, Maghrebi (Salloum and Habash, 2012), EU vs. Brazilian Portuguese (Marujo et al., 2011) or Turkish vs. Tatar (Altintas and Cicekli, 2002). In contrast, we are interested in developing language technologies for *immigrant* dialects, which are often understudied and lack written resources due to their unofficial status. When immigrant speakers face communication difficulties (e.g., bureaucratic affairs with the local officials, teacher-parent meetings, doctor-patient conversations) in the local languages (e.g., Dutch) of the host country, they are often provided with translation equivalents in the standard dialect (e.g., TR-Turkish) of their native languages. However, these translations ignore the evolution of the immigrant dialect.[1] By identifying the differences between two dialects of the same variety, we aim to improve Machine Translation systems targeting immigrant speakers. Our contributions are the following:

- We are the first to predict on-going dialect variation in immigrant contexts as opposed to studying established dialect variations.
- We are also the first to compare bilingual LVCs with the monolingual ones across two dialects of the same language.

[1] One of the authors failed the driving test in the Netherlands due to the dialect variation in the Turkish translation.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1391–1395,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

- Our comparison of grammatical versus contextual features reveals context to be much more important.
- We experiment with LVCs extracted from natural spoken data rather than relying on isolated occurences, out of context.

2 Method

We follow Baldwin and Kim (2010) and Butt (2010) in their definitions of LVCs, which state that there is a unity between the nominal and the verbal complements, but the meaning of the verb is somewhat bleached. In this study, we focus on Turkish LVCs with the verbal complements of *yapmak*/*etmek*, which both can be translated as "make/do". LVCs with these verbal complements are undergoing change in NL-Turkish (Doğruöz and Backus, 2009).

We experiment with the following features to predict NL-Turkish vs. TR-Turkish LVCs.

2.1 Nominal Features

In addition to traditional LVCs (e.g. [*ütü yapmak*] "iron do" (to iron) with both complements of Turkish origins), there is also foreign influence on Turkish LVCs. Section 2.1.1 describes the foreign influence on both NL-Turkish and TR-Turkish nominal complements based on their etymological origins.

2.1.1 Influence on Nominal Complements

Dutch Influence In example (1), the Dutch verb *overplaats* is nominalized through the infinitive marker (*-en*) and precedes the Turkish verb *yapmak* to form a Turkish-Dutch bilingual LVC.

Example 1:
O arkadaş [**overplaats-en yap-ıl-acak-tı**.]
That friend [replace-inf[2] do-pass-fut-past].
That friend would have been replaced.

In addition to borrowing nominalized Dutch verbs to form bilingual LVCs, Dutch LVCs are also translated as a chunk into NL-Turkish. These translated LVCs sound unconventional to TR-Turkish speakers (Doğruöz and Gries, 2012). In example (2), the LVC [*sınav yapmak*] "exam do" is a literal translation of the Dutch [*examen doen*] "exam-pl do", which is used to describe how students take high school exams to graduate.

[2]acc: accusative, fut:future, inf:infinitive, past:past tense, part: participle, pres: present tense, pl: plural, poss: possesive, prog:progressive tense, sg: singular

In a similar context, TR-Turkish speakers would have used [*sınav-a girmek*] "exam enter" instead. These LVCs are also classified as having their origins in another language.

Example 2:
Üç gündür [**sınav yap-ıyor-uz**].
Three day [exam do-prog-1pl].
We are having exams for the last three days.

Other Foreign Influences Although Dutch influence is clearly present in NL-Turkish LVCs, TR-Turkish LVCs are also not free of foreign influence. We have come across Arabic, Persian, French and English influences on Turkish LVCs with nominalized foreign verbs or literally translated LVCs as chunks. Example (3) illustrates how a borrowed Arabic verb (*hitap*, "address") is integrated as a nominal complement into a Turkish LVC [*hitap etmek*] "address do".

Example 3:
Hoca-m diye [**hitap edi-yo-z**] biz.
Teacher-poss.1sg like [address do-prog-1pl] we.
We address (him) as the teacher.

Example (4) illustrates how an English LVC [*do sports*] is borrowed into Turkish as a chunk [*spor yapmak*] "sports do".

Example 4:
Yazın [**spor yap-ıyo-z**].
summer spor do-prog-1pl
We do sports in summer.

We have identified the etymological origins of LVCs in both corpora using an online etymological dictionary.[3] Although LVCs of Dutch origin only occur in NL-Turkish, LVCs borrowed from other languages (e.g., Arabic, English, French) occur both in NL-Turkish and in TR-Turkish.

2.1.2 Case Marking

We also came across Turkish [N V] constructions with "*yapmak*" and "*etmek*" where the nominal complement acts as the object of the verb.

Turkish marks the direct objects with accusative case marking if they are definite (Enç, 1991). In example (5), the nominal element is the object of the verb, and thus it has the accusative marker.

Example 5:
Ben kendi [**iş-im-i yap-ıyor-um.**]
I own [work-poss.1sg-acc do-prog-1sg].
I do my own work.

[3]http://www.nisanyansozluk.com/

However, indefinite objects of the verb are left unmarked for case. In example (6), *yapmak* takes an indefinite object (*food*) as the complement. The boundary between [N V] constructions with indefinite nominal objects and LVCs are somewhat blurry. In both cases, the meaning of the verbal complement is bleached out and the nominal complement weighs heavier than the verbal one. We will not dwell further on this subtle distinction, but we plan future work on this topic following Cook et al. (2007) and Vincze et al. (2013).

Example 6:
Bazen [**yemek yap-ar-dı-m**]
Sometimes [food do-pres-past-1sg]
I used to sometimes prepare food.

Since Dutch does not mark objects of the verb morphologically, NL-Turkish speakers have difficulty (e.g., unnecessary addition or omission of case markers) in determining the definiteness of the nominal complements in [N V] constructions (Doğruöz and Backus, 2009). Therefore, we expect this feature to differentiate well between NL-Turkish and TR-Turkish [N V] constructions and LVCs with *yapmak/etmek* as verbal complements.

2.2 Verbal Complements

2.2.1 Finiteness

The verbs in LVCs are assumed to be flexible for inflection (Baldwin and Kim, 2010). However, we know little about how fineteness contributes to the formation of LVCs. To the best of our knowledge, finiteness has not been tested as a feature for identifying LVCs earlier. Therefore, we encoded the finiteness on *yapmak/etmek* as a binary (yes/no) feature in both data sets. Example (7) illustrates a non-finite LVC where the verb stem (*et*) is accompanied with an infinitive marker (*-mek*).

Example 7:
Misafir-ler-e [**ikram et-mek**] için al-dı-k
Guest-pl-dat [serve do-inf.] for buy-past-1pl
We bought (it) to serve the guests.

2.2.2 Type

NL-Turkish speakers could use other light verbs than TR-Turkish speakers for the same LVC construction. In example (8), the NL-Turkish speaker uses [*doğum etmek*] "birth do" instead of [*doğum yapmak*] "birth do", which is commonly preferred by TR-Turkish speakers. To capture this difference between the two dialects, we include the verb type as a feature as well.

Example 8:
Orda kadın [**doğum et-ti**].
There lady [birth do-past].
The lady gave birth there.

2.3 Word Order in LVCs

To the best of our knowledge, the influence of word order in LVCs has not been investigated as a feature. Although Turkish has a relatively flexible constituent order, object-verb (OV) is the most frequent word order for both NL-Turkish and TR-Turkish (Doğruöz and Backus, 2007). NL-Turkish speakers have adopted Dutch word order verb-object (VO) for some syntactic constructions, but we know little about the word order variation for LVCs. Encoding the word order of LVCs as a binary feature (OV vs. VO) could give us clues about differences or similarities of LVC use in NL-Turkish and in TR-Turkish. In example (9), the nominal complement (*one thing*) follows the verbal complement instead of preceding it as seen in earlier examples.

Example 9:
[**Yap-acak bir şey**] yok.
[Do-part. one thing] exist.not
There is nothing to do.

2.4 Context

So far, most studies were carried out ignoring the context of LVCs but focusing on their inherent grammatical features (e.g., lexical, syntactic, semantic or morphological). However, the context of an utterance could potentially provide additional useful cues. Since our data comes from natural conversations, we also experimented with the contextual information (words surrounding LVCs) as a feature for both data sets.

3 Data

Our data comes from spoken NL-Turkish (46 speakers from the Netherlands, 74,461 words) and TR-Turkish (22 speakers from Turkey, 28,731 words) corpora collected by one of the authors. LVC's are automatically extracted from the data using their stem forms ("yap-", "et-" without the infinitive -mEk). Table 1 illustrates the frequency of [N V] constructions with *etmek* and *yapmak* in both data sets.

	# etmek	# yapmak	# Total
NL-Turkish	449	543	992
TR-Turkish	527	755	1282
Total	976	1298	

Table 1: Distribution of *etmek* and *yapmak*.

4 Experiments

Our aim is to build a classifier that can determine whether a particular utterance containing an LVC (with the verbs *yapmak* or *etmek*) is uttered by an NL-Turkish or a TR-Turkish speaker.

We make use the following features in our classifier: (1) words from the context of the LVCs, (2) type of the light verb (*yapmak* or *etmek*), (3) the nominal complements, (4) finiteness of the verb (finite/non-finite), (5) case marking on the nominal complement (yes/no), (6) word order (VO/OV), and (7) etymological origins of the nominal complement (Arabic/Dutch/French/English/Persian/Turkish/mixed).

For the contextual features, we experiment with two models: (a) we distinguish between a word extracted from the context to the left or to the right of the verb (*yapmak* or *etmek*) in the feature space, and (b) we do not make a distinction in terms of context. The reason to experiment with option (a) is due to the potential importance of the word order. While the word order variation is already modeled through feature (6), we also include the context as an additional feature to test its effect. On the down side, adding context doubles the feature space size and could lead to data sparseness issues. For the context words, we did not filter out stopwords since they are part of natural speech.

For our experiments, we used an SVM classifier as implemented in LibSVM. We used a linear kernel; more complex kernels did not help. We report results for a 5-fold cross-validation.

5 Results

Table 2 illustrates the results of our experiments. All models outperform the majority class baseline of always predicting TR-Turkish (which is 56.38% accuracy) by a sizable margin. Furthermore, splitting the context into left/right yields approximately 1.5% absolute drop in accuracy.

	Split the Context?	
Features	**Left vs. Right**	**No Split**
Baseline	56.38	
Full model	**82.81**	**84.30**
no context	70.67	
no nominal complements	82.19	83.64
no info about etymol. origin	82.10	83.99
no finiteness	83.03	84.35
no case marking info	82.76	84.43
no word order info	82.89	84.43
no verb type	82.94	84.39

Table 2: Cross-validation accuracy (5 folds).

The lower part of the table shows the results when turning off each of the feature types. The context seems to be the most important feature since its exclusion leads to a drop from low-to-mid eighties to about 70% accuracy. Except the nominal complements and the information about etymological origins, most other features seem to have marginal impact on accuracy. Excluding the two features (nominal complements and etymological origins) lead to approximately 0.5% absolute drop in accuracy. The impact of the last four features in the table is tiny; excluding some of them even leads to a tiny improvement.

Overall, we can conclude that by far the most important features are the context features (without the left/right context split). The other useful features are the nominal complements and the information about the etymological origin of the borrowed LVCs. The remaining four linguistic features seem to be largely irrelevant.

6 Conclusion and Future Work

Language technologies are usually developed for standard dialects, ignoring the linguistic differences in other dialects such as those in immigrant contexts. One of the reasons for this is the difficulty of assessing and predicting linguistic differences across dialects. This is similar to efforts to translate well-established Arabic dialects (Bakr et al., 2008; Sawaf, 2010), or to adapt between Brazilian and European Portuguese (Marujo et al., 2011), Czech–Slovak (Hajič et al., 2000), Spanish–Portuguese (Nakov and Ng, 2009; Nakov and Ng, 2012), Turkish–Crimean Tatar (Altintas and Cicekli, 2002), Irish–Scottish Gaelic (Scannell, 2006), Bulgarian–Macedonian (Nakov and Tiedemann, 2012), Malay–Indonesian (Wang et al., 2012) or Mandarin–Cantonese (Zhang, 1998).

In this work, we have built a classifier that uses LVCs to differentiate between two different Turkish dialects: standard and immigrant. The results indicate that contextual features are most useful for this task. Although this requires further investigation, we can explain it by the thousands of features context generates: each contextual word is a feature. Thus, it is very hard for our grammatical features to compete against contextual features but they do have an impact.

We are planning to extend our study to dialects in other immigrant settings (e.g., Turkish in Germany) and to other types of multiword expressions (e.g., [N N] compounds).

References

Kemal Altintas and Ilyas Cicekli. 2002. A machine translation system between a pair of closely related languages. In *Proceedings of the 17th International Symposium on Computer and Information Sciences*, ISCIS '02, pages 192–196, Orlando, FL, USA.

Hitham Abo Bakr, Khaled Shaalan, and Ibrahim Ziedan. 2008. A hybrid approach for converting written Egyptian colloquial dialect into diacritized Arabic. In *Proceedings of the 6th International Conference on Informatics and Systems*, INFOS '08, Cairo, Egypt.

Timothy Baldwin and Su Nam Kim, 2010. *In Nitin Indurkhya and Fred J. Damerau (eds.), Handbook of Natural Language Processing*, chapter Multiword expressions, pages 267–292. CRC Press, Boca Raton, USA, second edition.

Miriam Butt, 2010. *In Mengistu Amberber, Brett Baker, and Mark Harvey (eds.), Complex predicates: cross-linguistic perspectives on event structure*, chapter The light verb jungle: still hacking away, pages 48–78. Cambridge University Press.

Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2007. Pulling their weight: Exploiting syntactic forms for the automatic identification of idiomatic expressions in context. In *Proceedings of MWE '07*, pages 41–48, Prague, Czech Republic.

Carmen Silva Corvalán. 2003. Otra mirada a la expresión del sujeto como variable sintáctica. *Lengua, variación y contexto: Estudios dedicados a Humberto López Morales*, 2:849–860.

A. Seza Doğruöz and Ad Backus. 2007. Postverbal elements in immigrant Turkish: Evidence of change? *International Journal of Bilingualism*, 11(2):185–220.

A. Seza Doğruöz and Ad Backus. 2009. Innovative constructions in Dutch Turkish: An assessment of ongoing contact-induced change. *Bilingualism: Language and Cognition*, 12(01):41–63.

A. Seza Doğruöz and Stefan Gries. 2012. Spread of on-going changes in an immigrant language: Turkish in the Netherlands. *Review of Cognitive Linguistics*, 10(2).

Malcolm Edwards and Penelope Gardner-Chloros. 2007. Compound verbs in codeswitching: Bilinguals making do? *International Journal of Bilingualism*, 11(1):73–91.

Mürvet Enç. 1991. The semantics of specificity. *Linguistic Inquiry*, 22(1):1–25.

François Grosjean. 2014. Bicultural bilinguals. *International Journal of Bilingualism*, pages 1–15.

Jan Hajič, Jan Hric, and Vladislav Kuboň. 2000. Machine translation of very close languages. In *Proceedings of ANLP '00*, pages 7–12, Seattle, WA, USA.

Luís Marujo, Nuno Grazina, Tiago Luís, Wang Ling, Luísa Coheur, and Isabel Trancoso. 2011. BP2EP - adaptation of Brazilian Portuguese texts to European Portuguese. In *Proceedings of EAMT '11*, pages 129–136, Leuven, Belgium.

Preslav Nakov and Hwee Tou Ng. 2009. Improved statistical machine translation for resource-poor languages using related resource-rich languages. In *Proceedings of EMNLP '09*, pages 1358–1367, Singapore.

Preslav Nakov and Hwee Tou Ng. 2012. Improving statistical machine translation for a resource-poor language using related resource-rich languages. *J. Artif. Intell. Res. (JAIR)*, 44:179–222.

Preslav Nakov and Jörg Tiedemann. 2012. Combining word-level and character-level models for machine translation between closely-related languages. In *Proceedings of ACL '12*, Jeju Island, Korea.

Wael Salloum and Nizar Habash. 2012. Elissa: A dialectal to standard Arabic machine translation system. In *Proceedings of COLING '12*, pages 385–392, Mumbai, India.

Hassan Sawaf. 2010. Arabic dialect handling in hybrid machine translation. In *Proceedings of AMTA '10*, Denver, Colorado.

Kevin Scannell. 2006. Machine translation for closely related language pairs. In *Proceedings of the LREC 2006 Workshop on Strategies for developing machine translation for minority languages*, pages 103–107, Genoa, Italy.

Sarah Thomason. 2008. Social and linguistic factors as predictors of contact-induced change. *Journal of language contact*, 2(1):42–56.

Weinreich Uriel. 1953. Languages in contact: Findings and problems. *Publications of the Linguistic Circle of New York*, vol. 1.

Veronika Vincze, István Nagy, and Richárd Farkas. 2013. Identifying English and Hungarian light verb constructions: A contrastive approach. In *Proceedings of ACL '13*, pages 255–261, Sofia, Bulgaria.

Pidong Wang, Preslav Nakov, and Hwee Tou Ng. 2012. Source language adaptation for resource-poor machine translation. In *Proceedings of EMNLP-CoNLL '12*, pages 286–296, Jeju Island, Korea.

Donald Winford. 2005. Contact-induced changes: Classification and processes. *Diachronica*, 22(2):373–427.

Xiaoheng Zhang. 1998. Dialect MT: a case study between Cantonese and Mandarin. In *Proceedings of the COLING '98*, pages 1460–1464, Montreal, Quebec, Canada.

Device-Dependent Readability for Improved Text Understanding

A-Yeong Kim **Hyun-Je Song** **Seong-Bae Park** **Sang-Jo Lee**
School of Computer Science and Engineering
Kyungpook National University
Daegu, 702-701, Korea
{aykim,hjsong,sbpark}@sejong.knu.ac.kr, sjlee@knu.ac.kr

Abstract

Readability is used to provide users with high-quality service in text recommendation or text visualization. With the increasing use of hand-held devices, reading device is regarded as an important factor for readability. Therefore, this paper investigates the relationship between readability and reading devices such as a smart phone, a tablet, and paper. We suggest readability factors that are strongly related with the readability of a specific device by showing the correlations between various factors in each device and human-rated readability. Our experimental results show that each device has its own readability characteristics, and thus different weights should be imposed on readability factors according to the device type. In order to prove the usefulness of the results, we apply the device-dependent readability to news article recommendation.

1 Introduction

Readability is a function that maps a given text into a readability score by considering "how easily the text is read and understood" (Richards et al., 1992; Zamanian and Heydari, 2012). Normally, the readability score is formulated as a combination of various factors. These factors reflect the easiness and understanding of the text and include text presentation format, font size, average ratio of annotated images, and sentence length (Hasegawa et al., 2008; Kitson, 1927; Ma et al., 2012; Öquist, 2006). Therefore, readability can be used to provide satisfiable services in text recommendation or text visualization.

The study on readability has begun in the education field to measure the level of a text. With the success of using readability in education (François and Fairon, 2012; Heilman et al., 2008; Ma et al., 2012), readability has been used in a range of domains recently. For example, in document retrieval, readability is used to provide documents to non-expert users so that they can read the retrieved documents easily (Jameel et al., 2012; Yan et al., 2006). In text mining, readability has been employed to analyze the characteristics of text. Especially, Hillbom showed the differences in readability between broadsheet newspapers and tabloids that share a similar political stance (Hillbom, 2009).

There is one important issue of readability that has not been studied in natural language processing. It is a reading device. That is, previous studies focused only on text printed on paper. However, with the increasing use of hand-held devices, people in these days use various reading devices such as a tablet and a smart phone as well as a paper. Readability score can be different according to the device type, because each device has its own idiosyncrasy. For example, assume that a system recommends the same news article to both user A who reads it in her smart phone and user B who reads it on paper. Although both users read the same article, user A might believe that her article is more difficult to read than user B because of the screen size of her smart phone.

This paper explores the relationship between reading devices and readability. For this purpose, we first investigate whether readability changes according to device type or not. Then, we analyze which readability factors are affected by reading devices. To see the relationship between readability factors and devices, various well-known readability factors are computed for news articles collected from an Internet portal. At the same time, the readability of each article is also manually rated. When the readability is rated manually, it is done three times for different reading devices of a smart phone, a tablet, and paper. The factors that affect the readability actually in each device are found out through the correlations between the factors and the manually-labeled readability. Some factors are important to the readability of smart phone, but insignificant to that of paper. Therefore, we discover the importance of each readability factor for each device by analyzing the correlations.

The usefulness of the device-dependent readability is proven by applying it to news article recommendation. That is, different importance weights for readability factors are considered according to device type when recommending news articles. Our experimental results show that the performance of news article recommendation gets best when the device used for reading news articles is identical to the device used for measuring readability. Therefore, it is essential to consider different importance weights according to device type

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1396–1404,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

in news article recommendation. It also proves that the proposed device-dependent readability reflects the characteristics of reading devices well.

The rest of this paper is organized as follows. We first review related studies on readability. Next, we introduce various readability factors and propose the device-dependent readability. Then, the news article recommendation using the device-dependent readability is explained. This recommendation is prepared to prove the usefulness of the device-dependent readability. In the experiments, we present the experimental results on the relationship between reading devices and readability. We also describe the experiments on news recommendation using the device-dependent readability and present their results. Finally, we summarize our research.

2 Related work

The history of readability studies began in the 1800s. Early studies focused on the frequency of easy words, sentence length, and word length (Huldén, 2004). Flesch designed a formula to calculate "reading ease" using only the average word length and sentence length (Flesch, 1948). He adjusted the relative importance between word length and sentence length using 100 words selected randomly from a corpus. This formula is called the Flesch-Kincaid formula, and is generally used in measuring the readability of a textbook (Kincaid et al., 1975). Dale and Chall (1949) defined a list of 3,000 easy words. Then, they used the average sentence length and the percentage of words not included in the list. These studies simply used superficial factors, and thus do not reflect syntactic factors.

Recent studies on readability use various factors including syntactic ones, and combine them to produce a highly predictive model of readability. François and Faircon (2012) proposed a readability formula with 46 textual factors for French as a foreign language. The factors represent lexical, syntactic, and semantic characteristics of sentences, and the specificities of French. They are extracted from 28 French Foreign Language (FFL) textbooks written for adults learning FFL. On the other hand, Pitler and Nenkova (2008) showed the relation between readability factors and readability. They used human ratings from the Wall Street Journal corpus, and computed the correlations between the readability factors and the average human ratings. According to their results, the average number of verb phrases in a sentence, the number of words in an article, the likelihood of the vocabulary, and the likelihood of the discourse relations are highly correlated with human ratings. However, these studies did not consider the reading devices, but focused on how well a text is written. Since the readability can be differentiated according to reading device, a reading device should be considered when computing the readability of a given text.

To the best of our knowledge, there are few studies on the readability on mobile devices that do not consider language-related aspects. Most studies on mobile devices focused on the development of new text format and layout to help users read documents easily. Öquist (2006) proposed a new text presentation format called the dynamic Rapid Serial Visual Presentation. According to his experimental results, this format helps to reduce eye movements. On the other hand, Hasegawa et al. (2008) evaluated the readability of documents on mobile devices with regard to screen and font size. They reported that the readability is improved when the characters are vertically enlarged. Readability on mobile devices is not reflected only by the visualization factors, but also by textual factors. Therefore, this paper explores the readability factors that reflect the lexical and grammatical complexity of text and are affected by reading devices.

3 Readability Factors

Table 1 lists the readability factors used in this paper. Basically, they are based on the factors proposed by Pitler and Nenkova (2008). However, some factors are excluded and some new factors are added. This is because some of their factors are computationally infeasible and language-dependent. As a result, we have thirteen readability factors. These readability factors are divided into four types: superficial, lexical, syntactic factors, and lexical cohesion.

3.1 Superficial Factors

Superficial factors were used in most early readability studies (Dale and Chall, 1949; Flesch, 1948; Kincaid et al., 1975), and reflect the construction of a text. We investigate four factors: text length (`TL`), sentence length (`SL`), average number of words per sentence (`WS`), and average number of characters per word (`CW`). Since longer text is perceived as "harder-to-read" than short one, these factors are all reciprocally related with readability.

The first two factors are related to length. `TL` counts the number of characters in a text, whereas `SL` computes the number of sentences. When a writer attempts to write many topics in a text, she tends to use many kinds of words simultaneously. As a result, the text becomes longer and more complex. Such long length of text disturbs a reader's comprehension of the text, and then it is more difficult for the reader to read the text (Heilman et al., 2008).

`WS` counts the average number of words per sentence, and `CW` reflects the average number of characters per word. When they are large, the sentence is difficult to read, which leads to difficulties in understanding the text. Especially, `CW` reflects compound nouns and technical words. For instance, compound nouns in Korean are usually long, because there is no spacing between words in a compound noun. For example, let us consider a compound noun, "*Daehanmingukjungboo*," which means the Korean government. Actually this compound noun consists of two independent nouns.

Type of Factors	Abbr.	Description
Superficial factors	TL	The number of characters in a text
	SL	The number of sentences in a text
	WS	Average number of words per sentence
	CW	Average number of characters per word
Lexical factor	LL	Article likelihood estimated by language model
Syntactic factors	PTD	Average parse tree depths per sentence
	NP	Average number of noun phrases per sentence
	VP	Average number of verb phrases per sentence
	SBAR	Average number of subordinate clauses per sentence
Lexical cohesion	COS	Average cosine similarity between pairs of adjacent sentences
	WO	Average word overlap between pairs of adjacent sentences
	NPO	Average word overlap over noun and pronoun only
	PRP	Average number of pronouns per sentence

Table 1: Description of readability factors

One is "*Daehanminguk*" meaning Korea and the other is "*Jungboo*" meaning a government. The two are concatenated to form a compound noun and become a long single word. In addition, many difficult words such as domain-specific terms tend to be long. Such lengthy words make it difficult to read a text.

3.2 Lexical Factor

Lexical factor determines whether a given text consists of frequent words. Texts that express a new trend in various fields often use many newly coined words. Such neologisms make it difficult to read and understand a text. Therefore, an easily-understandable text is composed of widely-used words rather than unusual words.

In order to compute the use of frequent words in a text, a unigram language model is used as in the work of Pitler and Nenkova (2008). In this model, the log likelihood of text t is computed by

$$\sum_{w \in t} C(w) \cdot \log P(w|B). \quad (1)$$

where $P(w|B)$ is the probability of a word w according to a background corpus B, and $C(w)$ is the number of times that w appears in t.

This factor examines the familiarity of the words used in the text. The more frequently a word appears in the background corpus, the more familiar it is regarded. The frequency of a word w is then reflected into $P(w|B)$ computed from the independent background corpus B. Therefore, the factor LL is positively related with readability.

3.3 Syntactic Factors

Syntactic factors reflect sentence complexity directly that affects human processing of a sentence. We consider the average parse tree depth per sentence (PTD), the average number of noun phrases per sentence (NP), the average number of verb phrases per sentence (VP), and the average number of subordinate clauses per sentence (SBAR) as syntactic factors. These four factors were defined by Schwarm and Ostendorf (2005).

A reader regards a text as difficult when the sentences in the text have large parse tree depths or many subordinate clauses. Thus, PTD and SBAR are related negatively with readability. On the other hand, the relationship of NP and VP to readability are not one way. The large number of noun phrases in a text requires a reader to remember more items (Barzilay and Lapata, 2008; Pitler and Nenkova, 2008). However, it also makes the text more interesting. The texts written for adults actually contain more entities than those written for children (Barzilay and Lapata, 2008). The same is true for VP. The large number of verb phrases in a sentence makes the sentence more complex. However, people feel that a text is more easier to comprehend when related clauses are grouped together (Bailin and Grafstein, 2001).

3.4 Lexical Cohesion

Lexical cohesion denotes how the sentences in a text are semantically connected. People usually bring continuous sentences into their mind at the same time, and interpret them as a single unit (Okazaki et al., 2005). In other words, a reader prefers text whose sentences are smoothly connected to text whose sentences are independent of one another. Therefore, sentence continuity plays a primary role in understanding an entire text.

In the classic study of cohesion, various uses of cohesive elements such as pronouns, definite articles, and topic continuity have been discussed (Halliday and Hasan, 1976). This paper uses the average cosine similarity (COS), word overlap (WO), word overlap over just nouns and pronouns (NPO) between pairs of adjacent sentences, and the average number of pronouns per sentence (PRP). COS, WO, and NPO are superficial measures of topic continuity, whereas PRP is an indicative feature of sentence continuity. High values for these factors imply that the sentences in the text are related somehow. Therefore, these factors are believed to be related positively with readability.

3.5 Measurement of Readability

When a reading device d is given, the readability of text t, represented as $R(t|d)$, is formulated as a combination of readability factors with their corresponding weight in the device. We assume that $w_{i|d}$, the weight of a readability factor f_i, is dependent on the reading device d. Following the previous work of Pitler and Nenkova (2008), we also assume that each readability factor affects readability independently. Therefore, readability is calculated as a weighted linear sum of all readability factors. That is, $R(t|d)$ is computed by

$$R(t|d) = \sum_{i \in \{1,2,...,M\}} w_{i|d} \cdot f_i(t) \quad (2)$$

where M is the number of readability factors.

Each weight $w_{i|d}$ is determined from a set of news articles T. We collected a large number of news articles from an Internet news portal. The readability of each article was manually labeled. This is done three times, since we have three different devices of a smart phone, a tablet, and paper. Since human rating of each article $t \in T$ is available for each device, $w_{i|d}$'s can be estimated by linear regression. These weights are different according to the devices.

4 News Article Recommendation by Device-Dependent Readability

The fact that the weights $w_{i|d}$ in Equation (2) are different for each device d implies that the readability measurement should be different depending on the device type. In order to see the usefulness of this device-dependent readability, we apply it to news article recommendation. News article recommendation aims to provide a user with news articles that interest the user. Thus, it selects a few articles that meet user preference from a gigantic amount of news events. Various methods have reported notable results in news article recommendation (Das et al., 2007; Li et al., 2010; Liu et al., 2010). In addition, with the recent interest in hand-held devices, the demand for news recommendation on hand-held devices is increasing. However, there has been, at least as far as we know, no study on the readability of hand-held devices.

Device-dependent readability is reflected into news article recommendation through a re-ranking framework. Figure 1 depicts the overall process of suggesting news articles for a specific device with the device-dependent readability. The point of this figure is to measure how appropriate a news article is for a specific reading device. For this, a news recommendation system first chooses a set of news articles from a news repository based on its own criterion. Then, we re-rank them by the device-dependent readability to obtain the final set of ranked news articles for the device.

Formally, a news article recommendation ranks a set of articles, $\mathbf{A} = \{a_1, a_2, ..., a_m\}$, where a_i represents the i-th article. The order between ranks $a_1 \succ a_2 \succ ... \succ a_m$ should be satisfied by the criterion of the recommendation system. That is, assuming that the system has a score function $score(a_i)$, $score(a_i) > score(a_j)$ has to be met if $a_i \succ a_j$. Then, the top $k(k \leq m)$ articles of $\mathbf{A}$ by the score function are suggested as appropriate news articles. After that, the selected articles are re-ranked by another criterion, the device-dependent readability. That is, the final rank of an article within the selected set is determined by another function, $rerank$. Since this function has to reflect the device-dependent readability, it takes two parameters. One is an article, and the other is a device type. The re-rank function is modeled as

$$\begin{aligned} rerank(a, d) &= R(a|d) \\ &= \sum_{i \in \{1,2,...,M\}} w_{i|d} \cdot f_i(a). \end{aligned} \quad (3)$$

As a result, the readability-based re-ranking module suggests the news articles based on how easily the articles are read on a specific reading device. Note that even the same article would be ranked differently according to the device type because the article is re-ranked by the device-dependent readability. At last, the top $k^*(k^* \leq k)$ re-ranked articles among them are suggested as final news articles.

	Min	Max	Average
Article length	68	610	346.5
# of sentences	1	14	6.24
# of words per sentence	8	33	16.93
# of words per article	17	178	99.34

Table 2: Statistics of the news article data

5 Experiments

5.1 Experiments on Readability Factors

5.1.1 Experiment Settings

For the experiments of analyzing relationship between readability factors and readability, we collected a Korean news corpus from Naver News[1]. This corpus contains news articles from June 10, 2013 to June 25, 2013. We selected 74 articles randomly from the corpus which were used for readability formula and showing the relationships between readability factors. All selected articles belong to one of three categories: 'Politics', 'Entertainment', and 'Sports'. A set of these 74 news articles becomes T, and is used to compute the weights in Equation (2). Table 2 describes a simple statistics of the selected news articles. The shortest article consists of 68 characters, whereas the longest one has 610 characters. The average length of article is 346.5. The shortest article is written in one sentence, and the longest has 14 sentences. One article has approximately 6.24 sentences on average. In addition, the

[1] A Korean news portal of which web address is `http://news.naver.com`.

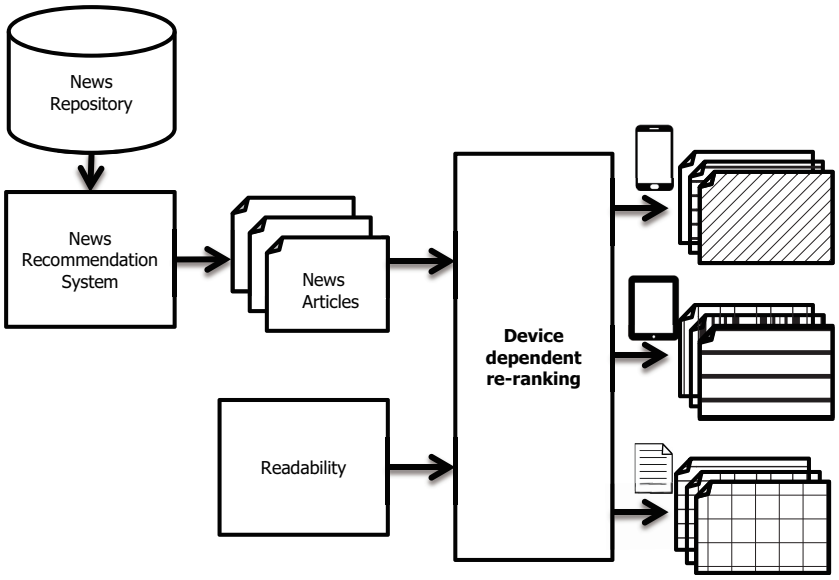


Figure 1: Overall process of re-ranking news articles based on device-dependent readability

number of words per sentence ranges from 8 to 33, and the average is 16.93. The minimum number of words in an article is 17, and the maximum number of words is 178. An article is composed of 99.34 words on average.

In order to compute the lexical factor LL by Equation (1), a background corpus B is required. Since this corpus should be independent from the news articles explained above, the Naver News is adopted again to generate B. For the background corpus B, we collected news articles from January 1, 2013 to September 6, 2013, but excluded the articles from June 10 to June 25, because they are already used. This corpus consists of 298,729 articles with 3,264,104 distinct words.

The readability score for each article was manually labeled by three undergraduate students. To investigate the relationship between reading devices and readability, each article was read using three different reading devices. The Galaxy Note 1 with a 5-inch screen is used as the smart phone, Galaxy Tab 10.1 with a 10.1-inch screen is used as the tablet, and A4-size paper is used for the paper. That is, the human annotators read and rated 74 articles per device. The order of the devices where the annotators evaluated readability is smart phone, tablet, and paper. This order was maintained for all the experiments. All aspects but content texts were under control. For instance, font = "Gothic, 12 pt" (this is most commonly used font and size that most Korean web pages and textbooks use), font color = "black", alignment = "both" were used for all three devices. In addition, the non-content aspects were exactly same for devices because the annotators of readability and the recommended articles shared the reading devices. Although these aspects affect readability and many previous studies already proved it, it is not our concern. We only attempt to capture how readability is affected by the content in different types of devices.

Reading device	Min	Max	Average
Smart phone	1.67	5	3.423 ± 0.741
Tablet	1.33	5	3.531 ± 0.837
Paper	2	5	3.360 ± 0.594

Table 3: Readability scores given by human annotators

Human annotators can remember the content of news articles when they read articles with three devices. The human annotators were asked to read and evaluate many articles within a relatively short period. Therefore, before the main experiments, we performed a pilot experiment on the memory effects of previously read articles and verified it empirically. We hired three undergraduate students who were not involved in our main experiments. The students read the same 250 articles four times, and these also come from Naver News corpus which are not included the previous 74 articles. After their first reading, they read the articles again in 3, 7, and 14 days later. After 3 days, two students remembered the articles somewhat, but one student remembered them vaguely. Since they almost forgot the articles after 7 days, we placed 7 days interval between devices.

The readability score of an article was rated by the annotators using the questions in the work of Pitler and Nenkova (2008). We use only two of the questions, while they used four questions for the annotators. Their questions are intended to measure the extent of how well a text is written, how it fits together, how easy it is to understand, and how interesting it is. We can consider "well-written" and "fit-together" as a syntactic perspective, whereas "easy to understand" and "interesting" belong to a content perspective. For such a

Smart phone		Tablet		Paper	
Factor	Value	Factor	Value	Factor	Value
SL	-0.394	SL	-0.370	NP	0.298
TL	-0.293	WS	0.321	WS	0.278
WS	0.288	LL	0.253	LL	0.268
LL	0.249	NP	0.240	VP	0.244

Table 4: Pearson correlation coefficients of important readability factors

reason, four questions can be summarized in two questions. The two questions used are

- How well-written is this article?
- How interesting is this article?

For these two questions, each annotator assigns a score between 1 and 5 to each article. Here, 1 point means that the article is worst and 5 point implies that it is best. A readability score of one human annotator is composed with the average of two questions (well-written, interesting). We used the average of three human annotators' readability scores in our experiments. Table 3 shows the readability scores of the articles for each device. According to this table, the readability score ranges from 1.67 to 5 for the smart phone, 1.33 to 5 for the tablet, and ranges from 2 to 5 for the paper. The average readability is 3.423 for the smart phone, 3.531 for the tablet, and 3.360 for the paper. To see the inter-judge agreement among annotators, the Kappa coefficient (Fleiss, 1971) is used. The Kappa values for the 'smart phone', 'tablet', and 'paper' are 0.342, 0.333, and 0.361, respectively. All these values correspond to *fair agreement*.

5.1.2 Experimental Results

In order to see the importance of each factor in a specific device, we adopt the Pearson correlation coefficients between readability factors and reading devices. Table 4 lists the four most important factors in each device and their Pearson correlation coefficients. Especially, p-value is smaller than 0.05 for all factors in this table.

For the smart phone, SL, the number of sentences in a text, is the most important readability factor. Its correlation with the smart phone is -0.394. TL, the number of characters, is the second important factor and has a negative correlation of -0.293. These results imply that readers are negatively sensitive to the length of an article because of the small display size of a smart phone. That is, in the smart phone, longer articles are recognized as difficult to read compared to shorter ones. The number of words per sentence, WS, is the third important factor with correlation of 0.288. The log-likelihood of an article, LL, is also positively related with the readability, which proves that widely-used words make it easy to understand an article. The top three factors are superficial with regard to text length. Therefore, the superficial factors are more important than other types of factors for the smart phone.

SL is the most critical readability factor even for the tablet. It affects readability with high correlation of -0.370. The second important factor is WS with correlation of 0.321. Both of these factors are superfical. The third important factor, LL, is positively related with readability as expected. The fourth factor that affects readability is the number of noun phrases, NP. It is natural for NP to be positively related with the readability.

Finally, for the paper, NP is most strongly related to readability with correlation of 0.298. The second important factor is WS, whose correlation is 0.278. LL is the third important factor and shows a positive relationship. Note that WS and LL are important readability factors for all devices. The next important readability factor for the paper is the average number of verb phrases (VP). The articles with many noun phrases and verb phrases are perceived as easier-to-read for the paper. Note that the importance of superficial factors is limited for the paper. We expected that WS is negatively related, but, it is positively related with readability for all three devices. The reason for this could be that the annotators thought the articles with higher WS are more interesting.

The important factors for the smart phone are different from those for the paper. On the other hand, the tablet shares many factors with both the smart phone and the paper. Because the screen size of a tablet is similar to the size of an A4 paper, the tablet and the paper share readability factors. However, length-related factors play a more important role than syntactic factors in the smart phone because a smart phone has a smaller screen.

5.2 Experiments on News Recommendation

5.2.1 Experiment Settings

Experiments for news article recommendation were performed to see the effectiveness of device-dependent readability. The process of news recommendation with device-dependent readability is as follows. For a specific device,

1. Select top-k news articles from a news repository by the criterion of the recommendation system.
2. Re-rank the k articles by the readability of the device using Equation (3).
3. Select top-k^* news articles by the new rank.
4. Human annotators read and rate the k^* articles with the device.
5. Compare the ranks of k^* articles by device-dependent readability with those by human ratings.

Since we have three types of devices, this process is performed three times with a different device.

The news articles from September 10, 2013 to September 12, 2013 collected from Naver News were

	Min	Max	Average
Article length	277	6,077	990.68
# of sentences	4	199	22.85
# of words per sentence	4	100	15.73
# of words per article	71	2,034	301.61

Table 5: Statistics of news data for recommendation

Reading device	Min	Max	Average
Smart phone	1	5	3.513 ± 0.962
Tablet	1	5	3.344 ± 0.852
Paper	1	5	3.250 ± 0.907

Table 6: Scores of news articles by human annotators in news recommendation

used as the news repository. The number of times that a news article was actually read by its anonymous readers at the portal site is used as the criterion for the recommendation system. Since this criterion is provided on a daily basis and news articles were collected for three days, the process explained above is performed three times. The top twenty articles were selected by the criterion every day. That is, $k = 20$. Table 5 shows the statistics of the total 60 articles. The shortest article consists of 277 characters, and the longest article has 6,077 characters. On average, an article is written with 990.68 characters. The minimum number of sentences in an article is 4, and the maximum number of sentences is 199. An article is composed of 22.85 sentences on average. The average number of words in a sentence is 15.73, whereas a sentence length ranges from 4 to 100 words. The shortest article has 71 words, and the longest article has 2,034 words. One article has approximately 301.61 words on average.

Three human annotators labeled the scores of the news articles manually. The annotators were the same persons who labeled the readability scores. Similar to the previous experiments, 7 days intervals was placed among devices to reduce the memory effect. The same two questions used in the previous section were used again for this experiment. The annotators assigned a score between 1 and 5 to every article for each question. The final score of an article was obtained by averaging six scores (two questions from three annotators). Table 6 summarizes the scores of the articles by the human annotators. As shown in this table, the article scores vary for all reading devices. The average scores for smart phone, tablet, and paper are 3.513, 3.344, and 3.250 respectively. The Kappa value for the 'smart phone' is 0.402, and that for both the 'tablet' and the 'paper' is 0.393. Thus, the value of 'smart phone' falls into *moderate agreement*, whereas those of the 'tablet' and 'paper' correspond to *fair agreement*. The performance of the news article recommendation is evaluated with the Normalized Discounted Cumulative Gain at top P (NDCG@P) (Järvelin and Kekäläinen, 2002).

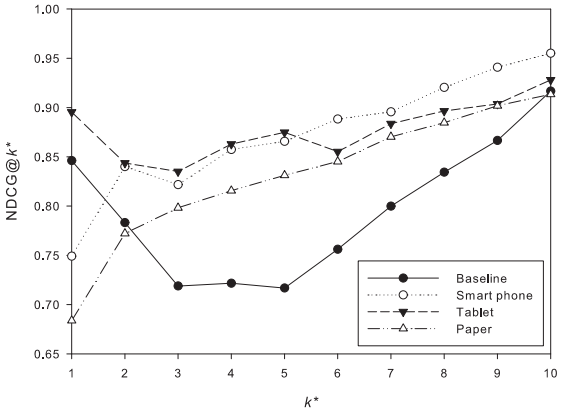


Figure 2: NDCG@k^* scores with various k^* for the *smart phone*.

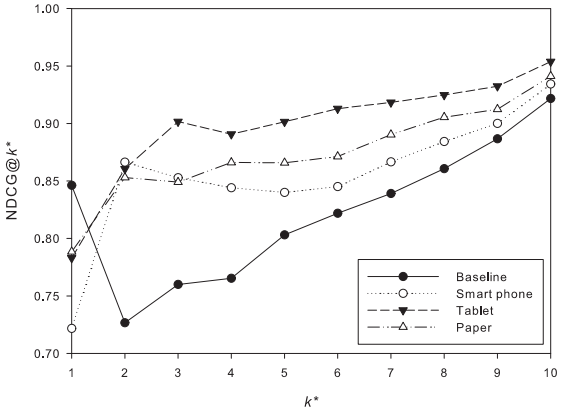


Figure 3: NDCG@k^* scores with various k^* for the *tablet*.

5.2.2 Experimental Results

For the a baseline criterion, we use the news article recommendation system in Naver, which recommends news article by the number of article hits. Figures 2 to 4 show the NDCG@k^* scores with $1 \leq k^* \leq 10$ for the three devices. Each graph in these figures compares the performance of various devices when the readability for a specific device is used. That is, Figure 2 depicts the NDCG@k^* scores for the recommended news articles when the articles are shown in the smart phone, the tablet, and the paper respectively. In computing their NDCG@k^* scores, the news articles are re-ranked by readability for the smart phone. Therefore, in this figure we expect that the NDCG@k^* score for using the smart phone is higher than those for using the tablet and paper. In the same way, Figure 3 and Figure 4 compare the NDCG@k^* scores when the readabilities for the tablet and paper are used.

In all three graphs, the best news recommendation performance is achieved when the device used to read

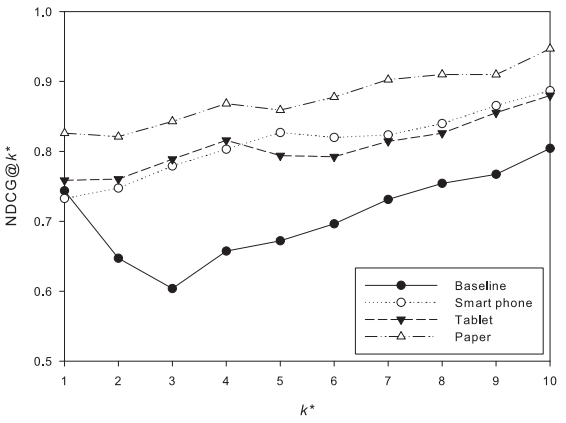


Figure 4: NDCG@k^* scores with various k^* for the *paper*.

news articles is the same as the device used for readability. In Figure 2, the use of the smart phone outperforms those of other devices when $k^* \geq 6$. This proves that the quality of highly ranked news articles is much better for the smart phone than for other devices, when the readability for smart phone is used.

Figure 3 shows the NDCG@k^* scores for using various devices when the news articles are re-ranked by readability for the tablet. In this figure, the use of the tablet as a reading device is better than using the smart phone or the paper. The performance difference is largest at $k^* = 3$. The difference becomes smaller as k^* increases up to 10, but the performance of tablet is still higher than those of others. In Figure 2 and 3, when $k^* = 1$, the baseline outperforms other devices. We believe this happens because the baseline chooses news articles by user-hit. Therefore, many articles recommended by the baseline are interesting because people tend to click more often when an article is interesting. As noted, readability reflects users' interests, which leads to high performance of the baseline. The performance of paper is best in Figure 4, since the articles are re-ranked by the readability for paper. Paper outperforms all other devices for all k^*s. Note that the performances of the baseline are always lowest regardless of reading device.

From all results above, we can infer that the use of device-dependent readability is helpful to news article recommendation. This is because the readability factors that affect the readers of news articles are different according to the reading device. Therefore, it is important to reflect the characteristics of a reading device when recommending news articles.

6 Conclusion

In this paper, we have proposed a device-dependent readability. Since a reading device is one of the most important features of readability, different weights have been assigned to the readability factors according to device type. We have shown that the important readability factors are distinct according to the reading device by investigating the correlation between the readability factors and the reading device. Through the correlation, we found that tablet shares many important factors with both smart phone and paper.

The experiments on the news articles collected from an Internet portal proved that readability is actually affected by the reading device. In addition, the validity of the device-dependent readability was shown by applying it to the news article recommendation. The news articles were first ranked by the criterion of the recommendation system. Then, they were re-ranked by the device-dependent readability. Our experiments showed that the recommendation performance of the re-ranked articles gets best when the device used for readability is the same as the reading device. These two types of experiments proved the importance and effectiveness of the device-dependent readability.

Acknowledgments

This work was supported by the IT R&D program of MSIP/KEIT (10044494, WiseKB: Big data based self-evolving knowledge base and reasoning platform) and the Industrial Strategic Technology Development Program (10035348, Development of a Cognitive Planning and Learning Model for Mobile Platforms) funded by the Ministry of Knowledge Economy(MKE, Korea).

References

Alan Bailin and Ann Grafstein. 2001. The linguistic assumptions underlying readability formulae: A critique. *Language & Communication*, 21(3):285–301.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Edgar Dale and Jeanne Chall. 1949. The concept of readability. *Elementary English*, 26(1):19–26.

Abhinandan Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, pages 271–280.

Joseph Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378–382.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of Applied Psychology*, 32(3):221–233.

Thomas François and Cédrick Fairon. 2012. An AI readability formula for French as a foreign language. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 466–477.

Michael Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman Group Ltd.

Satoshi Hasegawa, Kazuhiro Fujikake, Masako Omori, and Masaru Miyao. 2008. Readability of characters on mobile phone liquid crystal displays. *International Journal of Occupational Safety and Ergonomics (JOSE)*, 14(3):293–304.

Michael Heilman, Kevyn Collins-Thompson, and Maxine Eskenazi. 2008. An analysis of statistical models and features for reading difficulty prediction. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pages 71–79.

Kristina Hillbom. 2009. *Newspaper Readability: a Broadsheet vs. a Tabloid*. Ph.D. thesis, University of Gävle.

Måns Huldén. 2004. Linguistic complexity in two major american newspapers and the associated press newswire, 1900–2000. Master's thesis, Åbo Akademi University.

Shoaib Jameel, Wai Lam, and Xiaojun Qian. 2012. Ranking text documents based on conceptual difficulty using term embedding and sequential discourse cohesion. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 145–152.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

J. Peter Kincaid, Robert Fishburne Jr., Richard Rogers, and Brad Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, DTIC Document.

Harry Kitson. 1927. *The mind of the buyer*. MacMillan Company.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670.

Jiahui Liu, Peter Dolan, and Elin R. Pedersen. 2010. Personalized news recommendation based on click behavior. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*, pages 31–40.

Yi Ma, Eric Fosler-Lussier, and Robert Lofthus. 2012. Ranking-based readability assessment for early primary children's literature. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 548–552.

Naoaki Okazaki, Yutaka Matsuo, and Mitsuru Ishizuka. 2005. Improving chronological ordering of sentences extracted from multiple newspaper articles. *ACM Transactions on Asian Language Information Processing (TALIP)*, 4(3):321–339.

Gustav Öquist. 2006. *Evaluating readability on mobile devices*. Ph.D. thesis, Uppsala University.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 186–195.

Jack Richards, John Platt, Heidi Platt, and Christophe Candlin. 1992. *Longman Dictionary of Language Teaching and Applied Linguistics*, volume 78. Longman London.

Sarah Schwarm and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 523–530.

Xin Yan, Dawei Song, and Xue Li. 2006. Concept-based document readability in domain specific information retrieval. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 540–549.

Mostafa Zamanian and Pooneh Heydari. 2012. Readability of texts: State of the art. *Theory and Practice in Language Studies*, 2(1):43–53.

Predicting Chinese Abbreviations with Minimum Semantic Unit and Global Constraints

Longkai Zhang Li Li Houfeng Wang Xu Sun
Key Laboratory of Computational Linguistics (Peking University)
Ministry of Education, China
zhlongk@qq.com, {li.l,wanghf,xusun}@pku.edu.cn

Abstract

We propose a new Chinese abbreviation prediction method which can incorporate rich local information while generating the abbreviation globally. Different to previous character tagging methods, we introduce the minimum semantic unit, which is more fine-grained than character but more coarse-grained than word, to capture word level information in the sequence labeling framework. To solve the "character duplication" problem in Chinese abbreviation prediction, we also use a substring tagging strategy to generate local substring tagging candidates. We use an integer linear programming (ILP) formulation with various constraints to globally decode the final abbreviation from the generated candidates. Experiments show that our method outperforms the state-of-the-art systems, without using any extra resource.

1 Introduction

Abbreviation is defined as a shortened description of the original fully expanded form. For example, "NLP" is the abbreviation for the corresponding full form "Natural Language Processing". The existence of abbreviations makes it difficult to identify the terms conveying the same concept in the information retrieval (IR) systems and machine translation (MT) systems. Therefore, it is important to maintain a dictionary of the prevalent original full forms and the corresponding abbreviations.

Previous works on Chinese abbreviation generation focus on the sequence labeling method, which give each character in the full form an extra label to indicate whether it is kept in the abbreviation. One drawback of the character tagging strategy is that Chinese characters only contain limited amount of information. Using character-based method alone is not enough for Chinese abbreviation generation. Intuitively we can think of a word as the basic tagging unit to incorporate more information. However, if the basic tagging unit is word, we need to design lots of tags to represent which characters are kept for each unit. For a word with n characters, we should design at least 2^n labels to cover all possible situations. This reduces the generalization ability of the proposed model. Besides, the Chinese word segmentation errors may also hurt the performance. Therefore we propose the idea of "Minimum Semantic Unit" (MSU) which is the minimum semantic unit in Chinese language. Some of the MSUs are words, while others are more fine-grained than words. The task of selecting representative characters in the full form can be further broken down into selecting representative characters in the MSUs. We model this using the MSU-based tagging method, which can both utilize semantic information while keeping the tag set small.

Meanwhile, the sequence labeling method performs badly when the "character duplication" phenomenon exists. Many Chinese long phrases contain duplicated characters, which we refer to as the "character duplication" phenomenon. There is no sound criterion for the character tagging models to decide which of the duplicated character should be kept in the abbreviation and which one to be skipped. An example is "北京航空航天大学"(Beijing University of Aeronautics and Astronautics) whose abbreviation is "北航". The character "航" appears twice in the full form and only one is kept in the abbreviation. In these cases, we can break the long phase into local substrings. We can find the representative characters in the substrings instead of the long full form and let the decoding phase to integrate useful information globally. We utilize this sub-string based approach and obtain this local tagging information by labeling

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1405–1414,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

on the sub-string of the full character sequence.

Given the MSU-based and substring-based methods mentioned above, we can get a list of potential abbreviation candidates. Some of these candidates may not agree on keeping or skipping of some specific characters. To integrate their advantages while considering the consistency, we further propose a global decoding strategy using Integer Linear Programming(ILP). The constraints in ILP can naturally incorporate 'non-local' information in contrast to probabilistic constraints that are estimated from training examples. We can also use linguistic constraints like "adjacent identical characters is not allowed" to decode the correct abbreviation in examples like the previous "北航" example.

Experiments show that our Chinese abbreviation prediction system outperforms the state-of-the-art systems. In order to reduce the size of the search space, we further propose pruning constraints that are learnt from the training corpus. Experiment shows that the average number of constraints is reduced by about 30%, while the top-1 accuracy is not affected.

The paper is structured as follows. Section 1 gives the introduction. In section 2 we describe our method, including the MSUs, the substring-based tagging strategy and the ILP decoding process. Experiments are described in section 3. We also give a detailed analysis of the results in section 3. In section 4 related works are introduced, and the paper is concluded in the last section.

2 System Architecture

2.1 Chinese Abbreviation Prediction

Chinese abbreviations are generated by selecting representative characters from the full forms. For example, the abbreviation of "北京大学" (Peking University) is "北大" which is generated by selecting the first and third characters, see TABLE 1. This can be tackled from the sequence labeling point of view.

Full form	北	京	大	学
Status	Keep	Skip	Keep	Skip
Result	北		大	

Table 1: The abbreviation "北大" of the full form "北京大学" (Peking University)

From TABLE 1 we can see that Chinese abbreviation prediction is a problem of selecting representative characters from the original full form[1]. Based on this assumption, previous works mainly focus on this character tagging schema. In these methods, the basic tagging unit is the Chinese character. Each character in the full form is labeled as 'K' or 'S', where 'K' means the current character should be kept in abbreviation and 'S' means the current character should be skipped.

However, a Chinese character can only contain limited amount of information. Using character-based method alone is not enough for Chinese abbreviation generation. We introduce an MSU-based method, which models the process of selecting representative characters given local MSU information.

2.2 MSU Based Tagging

2.2.1 Minimum Semantic Unit

Because using the character-based method is not enough for Chinese abbreviation generation, we may think of word as the basic tagging unit to incorporate more information intuitively. In English, the abbreviations (similar to acronyms) are usually formed by concatenating initial letters or parts of a series of words. In other words, English abbreviation generation is based on words in the full form. However, in Chinese, word is not the most suitable abbreviating unit. Firstly, there is no natural boundary between Chinese words. Errors from the Chinese word segmentation tools will accumulate to harm the performance of abbreviation prediction. Second, it is hard to design a reasonable tag set when the length of a possible Chinese word is very long. The second column of TABLE 2 shows different ways of selecting representative characters of Chinese words with length 3. For a Chinese compound word with 3 characters, there are 6 possible ways to select characters. In this case we should have at least 6 kinds of tags to cover all possible situations. The case is even worse for words with more complicated structures. A suitable abbreviating unit should be smaller than word.

We propose the "Minimum Semantic Unit (MSU)" as the basic tagging unit. We define MSU as follows:

1. A word whose length is less or equal to 2 is an MSU.

[1] A small portion of Chinese abbreviations are not generated from the full form. For example, the abbreviation of "山东"(Shan Dong Province) is "鲁". However, we can use a look-up table to get this kind of abbreviations.

Full form	SK Label	MSUs
幼儿园(nursery)	幼/K 儿/S 园/S	幼儿+园
补贴费(allowance)	补/S 贴/K 费/S	补贴+费
信用卡(Credit card)	信/S 用/S 卡/K	信用+卡
水电站(Hydropower Station)	水/K 电/K 站/S	水+电+站
参议院(Senate)	参/K 议/S 院/K	参+议+院
音乐团(Music group)	音/S 乐/K 团/K	音乐+团

Table 2: Representing characters of Chinese words with length 3 (K for keep and S for skip) and the corresponding MSUs

2. A word whose length is larger than 2, but does not contain any MSUs with length equal to 2. For example, "火车站"(Railway Station) is not an MSU because the first two characters "火车"(Train) can form an MSU.

By this definition, all 6 strings in TABLE 2 are often thought as a word, but they are not MSUs in our view. Their corresponding MSU forms are shown in TABLE 2.

We collect all the MSUs from the benchmark datasets provided by the second International Chinese Word Segmentation Bakeoff[2]. We choose the Peking University (PKU) data because it is more fine-grained than all other corpora. Suppose we represent the segmented data as L (In our case L is the PKU word segmentation data), the MSU selecting algorithm is shown in TABLE 3.

For a given full form, we first segment it using a standard word segmenter to get a coarse-grained segmentation result. Here we use the Stanford Chinese Word Segmenter [3]. Then we use the MSU set to segment each word using the strategy of "Maximum Forward Matching"[4] to get the fine-grained MSU segmentation result.

2.2.2 Labeling strategy

For MSU-based tagging, we use a labeling method which uses four tags, "KSFL". "K" stands for "Keep the whole unit", "S" stands for "Skip the whole unit", "F" stands for "keep the First character of the unit", and Label "L" stands for "keep the Last character of the unit". An example is shown in TABLE 4.

The "KSFL" tag set is also applicable for MSUs whose length is greater than 2 (an example is "巧克力/chocolate"). By examining the corpus we find that such MSUs are either kept of skipped in the final abbreviations. Therefore, the labels of these long MSUs are either 'K' or 'S'. Empirically, this assumption holds for MSUs, but does not hold for words[5].

"国家语言文字工作委员会" (The abbreviation is "国家语委")	
KSFL	国家/K 语言/F 文字/S 工作/S 委员/F 会/S

Table 4: The abbreviation "国家语委" of "国家语言文字工作会" (National Linguistics Work Committee) based on MSU tagging.

2.2.3 Feature templates

The feature templates we use are as follows. See TABLE 5.

1. Word X_i $(-2 \leq i \leq 2)$
2. POS tag of word X_i $(-2 \leq i \leq 2)$
3. Word Bigrams (X_i, X_{i+1}) $(-2 \leq i \leq 1)$
4. Type of word X_i $(-2 \leq i \leq 2)$
5. Length of word X_i $(-2 \leq i \leq 2)$

Table 5: Feature templates for unit tagging. X represents the MSU sequence of the full form. X_i represents the ith MSU in the sequence.

Templates 1, 2 and 3 express word uni-grams and bi-grams. In MSU-based tagging, we can utilize the POS information, which we get from the Stanford Chinese POS Tagger[6]. In template 4, the type of word refers to whether it is a number, an English word or a Chinese word. Because the basic tagging unit is MSU, which carries word information, we can use many features that are infeasible in character-based tagging.

[2] http://www.sighan.org/bakeoff2005/

[3] http://nlp.stanford.edu/software/segmenter.shtml

[4] In Chinese, "Forward" means from left to right.

[5] In table 2, all examples are partly kept.

[6] http://nlp.stanford.edu/software/tagger.shtml

Init:
Let $MSUSet$ = empty set
For each word w in L:
  If $Length(w) \leq 2$
    Add w to $MSUSet$
  End if
End for
For each word w in L:
  If $Length(w) > 2$ and no word x in $MSUSet$ is a substring of w
    Add w to $MSUSet$
  End if
End for
Return $MSUSet$

Table 3: Algorithm for collecting MSUs from the PKU corpus

2.2.4 Sequence Labeling Model

The MSU-based method gives each MSU an extra indicative label. Therefore any sequence labeling model is appropriate for the method. Previous works showed that Conditional Random Fields (CRFs) can outperform other sequence labeling models like MEMMs in abbreviation generation tasks (Sun et al., 2009; Tsuruoka et al., 2005). For this reason we choose CRFs model in our system.

For a given full form's MSU list, many candidate abbreviations are generated by choosing the k-best results of the CRFs. We can use the forward-backward algorithm to calculate the probability of a specified tagging result. To reduce the searching complexity in the ILP decoding process, we delete those candidate tagged sequences with low probability.

2.3 Substring Based Tagging

As mentioned in the introduction, the sequence labeling method, no matter character-based or MSU-based, perform badly when the "character duplication" phenomenon exists. When the full form contains duplicated characters, there is no sound criterion for the sequence tagging strategy to decide which of the duplicated character should be kept in the abbreviation and which one to be skipped. On the other hand, we can tag the substrings of the full form to find the local representative characters in the substrings of the long full form. Therefore, we propose the sub-string based approach to given labeling results on sub-strings. These results can be integrated into a more accurate result using ILP constraints, which we will describe in the next section.

Another reason for using the sub-string based methods is that long full forms contain more characters and are much easier to make mistakes during the sequence labeling phase. Zhang et al. (2012) shows that if the full form contains less than 5 characters, a simple tagger can reach an accuracy of 70%. Zhang et al. (2012) also shows that if the full form is longer than 10 characters, the average accuracy is less than 30%. The numerous potential candidates make it hard for the tagger to choose the correct one. For the long full forms, although the whole sequence is not correctly labeled, we find that if we only consider its short substrings, we may find the correct representative characters. This information can be integrated into the decoding model to adjust the final result.

We use the MSU-based tagging method in the sub-string tagging. The labeling strategy and feature templates are the same to the MSU-based tagging method. In practice, enumerating all subsequences of a given full form is infeasible if the full form is very long. For a given full form, we use the boundary MSUs to reduce the possible sub-sequence set. For example, "中国科学院"(Chinese Academy of Science) has 5 subsequences: "中国", "中国科学", "科学", "科学院" and "院".

2.4 ILP Formulation of Decoding

Given the MSU-based and sub-sequence-based methods mentioned above as well as the prevalent character-based methods, we can get a list of potential abbreviation candidates and abbreviated substrings. We should integrate their advantages while keeping the consistency between each

candidate. Therefore we further propose a global decoding strategy using Integer Linear Programming(ILP). The constraints in ILP can naturally incorporate 'non-local' information in contrast to probabilistic constraints that are estimated from training examples. We can also use linguistic constraints like "adjacent identical characters is not allowed" to decode the correct abbreviation in examples like the "北航" example in section 1.

Formally, given the character sequence of the full form $c = c_1...c_l$, we keep Q top-ranked MSU-based tagging results T=($T_1, ..., T_Q$) and M tagged substrings S=($S_1, ..., S_M$) using the methods described in previous sections. We also use N top-ranked character-based tagging results R=($R_1, ..., R_N$) based on the previous character-based works. We also define the set $U = S \cup R \cup T$ as the union of all candidate sequences. Our goal is to find an optimal binary variable vector solution $\vec{v} = \vec{x}\vec{y}\vec{z} = (x_1, ..., x_M, y_1, ..., y_N, z_1, ..., z_Q)$ that maximizes the object function:

$$\lambda_1 \sum_{i=1}^{M} score(S_i) \cdot x_i + \lambda_2 \sum_{i=1}^{N} score(R_i) \cdot y_i$$

$$+\lambda_3 \sum_{i=1}^{Q} score(T_i) \cdot z_i$$

subject to constrains in TABLE 6. The parameters λ_1, λ_2, λ_3 controls the preference of the three parts, and can be decided using cross-validation.

Constraint 1 indicates that x_i, y_i, z_i are all boolean variables. They are used as indicator variables to show whether the corresponding tagged sequence is in accordance with the final result.

Constraint 2 is used to guarantee that at most one candidate from the character-based tagging is preserved. We relax the constraint to allow the sum to be zero in case that none of the top-ranked candidate is suitable to be the final result. If the sum equals zero, then the sub-sequence based tagging method will generate a more suitable result. Constrain 3 has the same utility for the MSU-based tagging.

Constraint 4, 5, 6 are inter-method constraints. We use them to guarantee that the labels of the preserved sequences of different tagging methods do not conflict with each other. Constraint 7 is used to guarantee that the labels of the preserved sub-strings do not conflict with each other.

Constraint 8 is used to solve the "character duplicate" problem. When two identical characters are kept adjacently, only one of them will be kept. Which one will be kept depends on the global decoding score. This is the advantage of ILP against traditional sequence labeling methods.

2.5 Pruning Constraints

The efficiency of solving the ILP decoding problem depends on the number of candidate tagging sequences N and Q, as well as the number of subsequences M. Usually, N and Q is less than 10 in our experiment. Therefore, M influences the time complexity the most. Because we use the boundary of MSUs instead of enumerating all possible subsequences, the value of M can be largely reduced.

Some characters are always labeled as "S" or "K" once the context is given. We can use this phenomenon to reduce the search space of decoding. Let c_i denote the i_{th} character relative to the current character c_0 and t_i denote the tag of c_i. The context templates we use are listed in TABLE 7.

Uni-gram Contexts	c_0, c_{-1}, c_1
Bi-gram Contexts	$c_{-1}c0$, $c_{-1}c_1$, c_0c_1

Table 7: Context templates used in pruning

With respect to a training corpus, if a context C relative to c_0 always assigns a certain tag t to c_0, then we can use this constraint in pruning. We judge the degree of "always" by checking whether $\frac{count(C \wedge t_0 = t)}{count(C)} > threshold$. The threshold is a non-negative real number under 1.0.

3 Experiments

3.1 Data and Evaluation Metric

We use the abbreviation corpus provided by Institute of Computational Linguistics (ICL) of Peking University in our experiments. The corpus is similar to the corpus used in Sun et al. (2008, 2009); Zhang et al. (2012). It contains $8,015$ Chinese abbreviations, including noun phrases, organization names and some other types. Some examples are presented in TABLE 8. We use 80% abbreviations as training data and the rest as testing data. In some cases, a long phrase may contain more than one abbreviation. For these cases, the corpus just keeps their most commonly used abbreviation for each full form.

The evaluation metric used in our experiment is the top-K accuracy, which is also used by Tsuruoka et al. (2005), Sun et al. (2009) and

1. $x_i \in \{0,1\}$, $y_i \in \{0,1\}$, $z_i \in \{0,1\}$
2. $\sum_{i=1}^{N} y_i \leq 1$
3. $\sum_{i=1}^{Q} z_i \leq 1$
4. $\forall R_i \in R$, $S_j \in S$, if R_i and S_j have a same position but the position gets different labels, then $y_i + x_j \leq 1$
5. $\forall T_i \in T$, $S_j \in S$, if T_i and S_j have a same position but the position gets different labels, then $z_i + x_j \leq 1$
6. $\forall R_i \in R$, $T_j \in T$, if R_i and T_j have a same position but the position gets different labels, then $x_i + z_j \leq 1$
7. $\forall S_i, S_j \in S$ if S_i and S_j have a same position but the position gets different labels, then $z_i + z_j \leq 1$
8. $\forall S_i, S_j \in S$ if the last character S_i keeps is the same as the first character S_j keeps, then $z_i + z_j \leq 1$

Table 6: Constraints for ILP

Type	Full form	Abbreviation
Noun Phrase	优秀稿件(Excellent articles)	优稿
Organization	作家协会(Writers' Association)	作协
Coordinate phrase	受伤死亡(Injuries and deaths)	伤亡
Proper noun	传播媒介(Media)	传媒

Table 8: Examples of the corpus (Noun Phrase, Organization, Coordinate Phrase, Proper Noun)

Zhang et al. (2012). The top-K accuracy measures what percentage of the reference abbreviations are found if we take the top N candidate abbreviations from all the results. In our experiment, top-10 candidates are considered in re-ranking phrase and the measurement used is top-1 accuracy (which is the accuracy we usually refer to) because the final aim of the algorithm is to detect the exact abbreviation.

CRF++[7] , an open source linear chain CRF tool, is used in the sequence labeling part. For ILP part, we use lpsolve[8], which is also an open source tool. The parameters of these tools are tuned through cross-validation on the training data.

3.2 Results

TABLE 9 shows the top-K accuracy of the character-based and MSU-based method. We can see that the MSU-based tagging method can utilize word information, which can get better performance than the character-based method. We can also figure out that the top-5 candidates include the reference abbreviation for most full forms. Therefore reasonable decoding by considering all possible labeling of sequences may improve the performance. Although the MSU-based methods only outperforms character-based methods by 0.75% for top-1 accuracy, it is much better when considering top-2 to top-5 accuracy (+2.5%). We further select the top-ranked candidates for ILP decoding. Therefore the MSU-based method can further improve the performance in the global decoding phase.

K	char-based	MSU-based
1	0.5714	0.5789
2	0.6879	0.7155
3	0.7681	0.7819
4	0.8070	0.8283
5	0.8333	**0.8583**

Table 9: Top-K ($K \leq 5$) results of character-based tagging and MSU-based tagging

We then use the top-5 candidates of character-based method and MSU-based method, as well as the top-2 results of sub-sequence labeling in the ILP decoding phase. Then we select the top-ranked candidate as the final abbreviation of each instance. TABLE 10 shows the results. We can see that the accuracy of our method is 61.0%, which improved by +3.89% compared to the character-based method, and +3.14% compared to the MSU-based method.

We find that the ILP decoding phase do play an important role in generating the right an-

[7] http://crfpp.sourceforge.net/

[8] http://lpsolve.sourceforge.net/5.5/

Method	Top-1 Accuracy
Char-based	0.5714
MSU-based	0.5789
ILP Result	**0.6103**

Table 10: Top-1 Accuracy after ILP decoding

swer. Some reference abbreviations which are not picked out by either tagging method can be found out after decoding. TABLE 11 shows the example of the organization name "高等学校统一招生办公室" (Higher Education Admissions Office). Neither the character-based method nor the MSU-based method finds the correct answer "高招办", while after ILP decoding, "高招办" becomes the final result. TABLE 12 and TABLE 13 give two more examples.

True Result	高招办
Char-based	高办
MSU-based	高统办
ILP Decoding	高招办

Table 11: Top-1 result of "高等学校统一招生办公室" (Higher Education Admissions Office)

True Result	超值
Char-based	物值
MSU-based	物超值
ILP Decoding	超值

Table 12: Top-1 result of "物超价值" (Articles exceed the value)

True Result	声光视效
Char-based	声光效
MSU-based	声效果
ILP Decoding	声光视效

Table 13: Top-1 result of "声音灯光视觉效果" (Visual effects of sound and lights)

3.3 Improvements Considering Length

Full forms that are longer than five characters are long terms. Long terms contain more characters, which is much easier to make mistakes. Figure 1 shows the top-1 accuracy respect to the term length using different tagging methods and using ILP decoding. The x-axis represents the length of the full form. The y-axis represents top-1 accuracy. We find that our method works especially better than pure character-based or MSU-based approach when the full form is long. By decoding using ILP, both local and global information are incorporated. Therefore many of these errors can be eliminated.

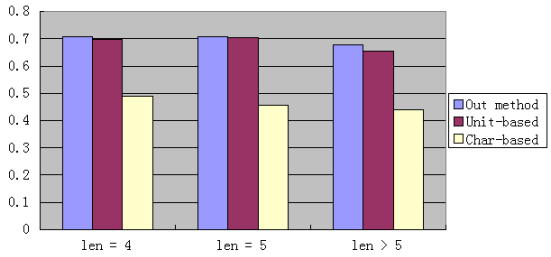


Figure 1: Top-1 accuracy of different methods considering length

3.4 Effect of pruning

As discussed in previous sections, if we are able to pre-determine that some characters in a certain context should be kept or skipped, then the number of possible boolean variable x can be reduced. TABLE 14 shows the differences. To guarantee a high accuracy, we set the threshold to be 0.99. When the original full form is partially tagged by the pruning constraints, the number of boolean variables per full form is reduced from 34.4 to 25.5. By doing this, we can improve the prediction speed over taking the raw input.

From TABLE 14 we can also see that the top-1 accuracy is not affected by these pruning constraints. This is obvious, because CRF itself has a strong modeling ability. The pruning constraints cannot improve the model accuracy. But they can help eliminate those false candidates to make the ILP decoding faster.

	Accuracy	Average length	Time(s)
raw	0.6103	34.4	12.5
pruned	0.6103	25.5	7.1

Table 14: Comparison of testing time of raw input and pruned input

3.5 Compare with the State-of-the-art Systems

We also compare our method with previous methods, including Sun et al. (2009) and Zhang et al. (2012). Because we use a different corpus, we re-implement the system Sun et al. (2009), Zhang

et al. (2012) and Sun et al. (2013), and experiment on our corpus. The first two are CRF+GI and DPLVM+GI in Sun et al. (2009), which are reported to outperform the methods in Tsuruoka et al. (2005) and Sun et al. (2008). For DPLVM we use the same model in Sun et al. (2009) and experiment on our own data. We also compare our approach with the method in Zhang et al. (2012). However, Zhang et al. (2012) uses different sources of search engine result information to re-rank the original candidates. We do not use any extra web resources. Because Zhang et al. (2012) uses web information only in its second stage, we use "BIEP"(the tag set used by Zhang et al. (2012)) to denote the first stage of Zhang et al. (2012), which also uses no web information. TABLE 15 shows the results of the comparisons. We can see that our method outperforms all other methods which use no extra resource. Because Zhang et al. (2012) uses extra web resource, the top-1 accuracy of Zhang et al. (2012) is slightly better than ours.

Method	Top-1 Accuracy
CRF+GI	0.5850
DPLVM+GI	0.5990
BIEP	0.5812
Zhang et al. (2012)	**0.6205**
Our Result	**0.6103**

Table 15: Comparison with the state-of-the-art systems

4 Related Work

Previous research mainly focuses on "abbreviation disambiguation", and machine learning approaches are commonly used (Park and Byrd, 2001; HaCohen-Kerner et al., 2008; Yu et al., 2006; Ao and Takagi, 2005). These ways of linking abbreviation pairs are effective, however, they cannot solve our problem directly. In many cases the full form is definite while we don't know the corresponding abbreviation.

To solve this problem, some approaches maintain a database of abbreviations and their corresponding "full form" pairs. The major problem of pure database-building approach is obvious. It is impossible to cover all abbreviations, and the building process is quit laborious. To find these pairs automatically, a powerful approach is to find the reference for a full form given the context, which is referred to as "abbreviation generation".

There is research on heuristic rules for generating abbreviations Barrett and Grems (1960); Bourne and Ford (1961); Taghva and Gilbreth (1999); Park and Byrd (2001); Wren et al. (2002); Hearst (2003). Most of them achieved high performance. However, hand-crafted rules are time consuming to create, and it is not easy to transfer the knowledge of rules from one language to another.

Recent studies of abbreviation generation have focused on the use of machine learning techniques. Sun et al. (2008) proposed a supervised learning approach by using SVM model. Tsuruoka et al. (2005); Sun et al. (2009) formalized the process of abbreviation generation as a sequence labeling problem. In Tsuruoka et al. (2005) each character in the full form is associated with a binary value label y, which takes the value S (Skip) if the character is not in the abbreviation, and value P (Preserve) if the character is in the abbreviation. Then a MEMM model is used to model the generating process. Sun et al. (2009) followed this schema but used DPLVM model to incorporate both local and global information, which yields better results. Sun et al. (2013) also uses machine learning based methods, but focuses on the negative full form problem, which is a little different from our work.

Besides these pure statistical approaches, there are also many approaches using Web as a corpus in machine learning approaches for generating abbreviations.Adar (2004) proposed methods to detect such pairs from biomedical documents. Jain et al. (2007) used web search results as well as search logs to find and rank abbreviates full pairs, which show good result. The disadvantage is that search log data is only available in a search engine backend. The ordinary approaches do not have access to search engine internals. Zhang et al. (2012) used web search engine information to rerank the candidate abbreviations generated by statistical approaches. Compared to their approaches, our method uses no extra resource, but reaches comparable results.

ILP shows good results in many NLP tasks. Punyakanok et al. (2004); Roth and Yih (2005) used it in semantic role labeling (SRL). Martins et al. (2009) used it in dependency parsing. (Zhao and Marcus, 2012) used it in Chinese word segmentation. (Riedel and Clarke, 2006) used ILP

in dependency parsing. However, previous works mainly focus on the constraints of avoiding boundary confliction. For example, in SRL, two argument of cannot overlap. In CWS, two Chinese words cannot share a same character. Different to their methods, we investigate on the conflict of labels of character sub-sequences.

5 Conclusion and Future work

We propose a new Chinese abbreviation prediction method which can incorporate rich local information while generating the abbreviation globally. We propose the MSU, which is more coarse-grained than character but more fine-grained than word, to capture word information in the sequence labeling framework. Besides the MSU-based method, we use a substring tagging strategy to generate local substring tagging candidates. We use an ILP formulation with various constraints to globally decode the final abbreviation from the generated candidates. Experiments show that our method outperforms the state-of-the-art systems, without using any extra resource. This method is not limited to Chinese abbreviation generation, it can also be applied to similar languages like Japanese.

The results are promising and outperform the baseline methods. The accuracy can still be improved. Potential future works may include using semi-supervised methods to incorporate unlabeled data and design reasonable features from large corpora. We are going to study on these issues in the future.

Acknowledgments

This research was partly supported by National Natural Science Foundation of China (No.61370117,61333018,61300063),Major National Social Science Fund of China(No.12&ZD227), National High Technology Research and Development Program of China (863 Program) (No. 2012AA011101), and Doctoral Fund of Ministry of Education of China (No. 20130001120004). The contact author of this paper, according to the meaning given to this role by Key Laboratory of Computational Linguistics, Ministry of Education, School of Electronics Engineering and Computer Science, Peking University, is Houfeng Wang. We thank Ke Wu for part of our work is inspired by his previous work at KLCL.

References

Adar, E. (2004). Sarad: A simple and robust abbreviation dictionary. *Bioinformatics*, 20(4):527–533.

Ao, H. and Takagi, T. (2005). Alice: an algorithm to extract abbreviations from medline. *Journal of the American Medical Informatics Association*, 12(5):576–586.

Barrett, J. and Grems, M. (1960). Abbreviating words systematically. *Communications of the ACM*, 3(5):323–324.

Bourne, C. and Ford, D. (1961). A study of methods for systematically abbreviating english words and names. *Journal of the ACM (JACM)*, 8(4):538–552.

HaCohen-Kerner, Y., Kass, A., and Peretz, A. (2008). Combined one sense disambiguation of abbreviations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 61–64. Association for Computational Linguistics.

Hearst, M. S. (2003). A simple algorithm for identifying abbreviation definitions in biomedical text.

Jain, A., Cucerzan, S., and Azzam, S. (2007). Acronym-expansion recognition and ranking on the web. In *Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on*, pages 209–214. IEEE.

Martins, A. F., Smith, N. A., and Xing, E. P. (2009). Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 342–350. Association for Computational Linguistics.

Park, Y. and Byrd, R. (2001). Hybrid text mining for finding abbreviations and their definitions. In *Proceedings of the 2001 conference on empirical methods in natural language processing*, pages 126–133.

Punyakanok, V., Roth, D., Yih, W.-t., and Zimak, D. (2004). Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th international conference on Compu-*

tational Linguistics, page 1346. Association for Computational Linguistics.

Riedel, S. and Clarke, J. (2006). Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 129–137. Association for Computational Linguistics.

Roth, D. and Yih, W.-t. (2005). Integer linear programming inference for conditional random fields. In *Proceedings of the 22nd international conference on Machine learning*, pages 736–743. ACM.

Sun, X., Li, W., Meng, F., and Wang, H. (2013). Generalized abbreviation prediction with negative full forms and its application on improving chinese web search. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 641–647, Nagoya, Japan. Asian Federation of Natural Language Processing.

Sun, X., Okazaki, N., and Tsujii, J. (2009). Robust approach to abbreviating terms: A discriminative latent variable model with global information. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 905–913. Association for Computational Linguistics.

Sun, X., Wang, H., and Wang, B. (2008). Predicting chinese abbreviations from definitions: An empirical learning approach using support vector regression. *Journal of Computer Science and Technology*, 23(4):602–611.

Taghva, K. and Gilbreth, J. (1999). Recognizing acronyms and their definitions. *International Journal on Document Analysis and Recognition*, 1(4):191–198.

Tsuruoka, Y., Ananiadou, S., and Tsujii, J. (2005). A machine learning approach to acronym generation. In *Proceedings of the ACL-ISMB Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*, pages 25–31. Association for Computational Linguistics.

Wren, J., Garner, H., et al. (2002). Heuristics for identification of acronym-definition patterns within text: towards an automated construction of comprehensive acronym-definition dictionaries. *Methods of information in medicine*, 41(5):426–434.

Yu, H., Kim, W., Hatzivassiloglou, V., and Wilbur, J. (2006). A large scale, corpus-based approach for automatically disambiguating biomedical abbreviations. *ACM Transactions on Information Systems (TOIS)*, 24(3):380–404.

Zhang, L., Li, S., Wang, H., Sun, N., and Meng, X. (2012). Constructing Chinese abbreviation dictionary: A stacked approach. In *Proceedings of COLING 2012*, pages 3055–3070, Mumbai, India. The COLING 2012 Organizing Committee.

Zhao, Q. and Marcus, M. (2012). Exploring deterministic constraints: from a constrained english pos tagger to an efficient ilp solution to chinese word segmentation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1062, Jeju Island, Korea. Association for Computational Linguistics.

Using Structured Events to Predict Stock Price Movement: An Empirical Investigation

Xiao Ding[†]*, **Yue Zhang**[‡], **Ting Liu**[†], **Junwen Duan**[†]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{xding, tliu, jwduan}@ir.hit.edu.cn
[‡]Singapore University of Technology and Design
yue_zhang@sutd.edu.sg

Abstract

It has been shown that news events influence the trends of stock price movements. However, previous work on news-driven stock market prediction rely on shallow features (such as bags-of-words, named entities and noun phrases), which do not capture structured entity-relation information, and hence cannot represent complete and exact events. Recent advances in Open Information Extraction (Open IE) techniques enable the extraction of structured events from web-scale data. We propose to adapt Open IE technology for event-based stock price movement prediction, extracting structured events from large-scale public news without manual efforts. Both linear and nonlinear models are employed to empirically investigate the hidden and complex relationships between events and the stock market. Large-scale experiments show that the accuracy of S&P 500 index prediction is 60%, and that of individual stock prediction can be over 70%. Our event-based system outperforms bags-of-words-based baselines, and previously reported systems trained on S&P 500 stock historical data.

1 Introduction

Predicting stock price movements is of clear interest to investors, public companies and governments. There has been a debate on whether the market can be predicted. The Random Walk Theory (Malkiel, 1973) hypothesizes that prices are determined randomly and hence it is impossible to outperform the market. However, with advances of AI, it has been shown empirically that stock price movement is predictable (Bondt and Thaler, 1985; Jegadeesh, 1990; Lo and MacKinlay, 1990; Jegadeesh and Titman, 1993). Recent work (Das and Chen, 2007; Tetlock, 2007; Tetlock et al., 2008; Si et al., 2013; Xie et al., 2013; Wang and Hua, 2014) has applied Natural Language Processing (NLP) techniques to help analyze the effect of web texts on stock market prediction, finding that events reported in financial news are important evidence to stock price movement prediction.

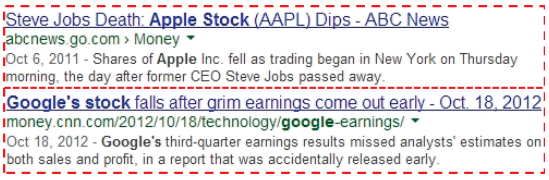


Figure 1: Example news for *Apple Inc.* and *Google Inc.*

As news events affect human decisions and the volatility of stock prices is influenced by human trading, it is reasonable to say that events can influence the stock market. Figure 1 shows two pieces of financial news about *Apple Inc.* and *Google Inc.*, respectively. Shares of *Apple Inc.* fell as trading began in New York on Thursday morning, the day after its former CEO Steve Jobs passed away. Google's stock fell after grim earnings came out. Accurate extraction of events from financial news may play an important role in stock market prediction. However, previous work represents news documents mainly using simple features, such as bags-of-words, noun phrases, and named entities (Lavrenko et al., 2000; Kogan et al., 2009; Luss and d'Aspremont, 2012; Schumaker and Chen, 2009). With these unstructured features, it is difficult to capture key events embedded in financial news, and even more difficult to model the impact of events on stock market prediction. For example, representing the event "Apple has sued Samsung Electronics for copying 'the look and feel'

*This work was done while the first author was visiting Singapore University of Technology and Design

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1415–1425,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

of its iPad tablet and iPhone smartphone." using term-level features {"Apple", "sued", "Samsung", "Electronics", "copying", ...} alone, it can be difficult to accurately predict the stock price movements of *Apple Inc.* and *Samsung Inc.*, respectively, as the unstructured terms cannot indicate the actor and object of the event.

In this paper, we propose using structured information to represent events, and develop a prediction model to analyze the relationship between events and the stock market. The problem is important because it provides insights into understanding the underlying mechanisms of the influence of events on the stock market. There are two main challenges to this method. On the one hand, how to obtain structured event information from large-scale news streams is a challenging problem. We propose to apply Open Information Extraction techniques (Open IE; Banko et al. (2007); Etzioni et al. (2011); Fader et al. (2011)), which do not require predefined event types or manually labeled corpora. Subsequently, two ontologies (i.e. VerbNet and WordNet) are used to generalize structured event features in order to reduce their sparseness. On the other hand, the problem of accurately predicting stock price movement using structured events is challenging, since events and the stock market can have complex relations, which can be influenced by hidden factors. In addition to the commonly used linear models, we build a deep neural network model, which takes structured events as input and learn the potential relationships between events and the stock market.

Experiments on large-scale financial news datasets from Reuters[1] (106,521 documents) and Bloomberg[2] (447,145 documents) show that events are better features for stock market prediction than bags-of-words. In addition, deep neural networks achieve better performance than linear models. The accuracy of S&P 500 index prediction by our approach outperforms previous systems, and the accuracy of individual stock prediction can be over 70% on the large-scale data.

Our system can be regarded as one step towards building an expert system that exploits rich knowledge for stock market prediction. Our results are helpful for automatically mining stock price related news events, and for improving the accuracy of algorithm trading systems.

[1]http://www.reuters.com/

[2]http://www.bloomberg.com/

2 Method

2.1 Event Representation

We follow the work of Kim (1993) and design a structured representation scheme that allows us to extract events and generalize them. Kim defines an event as a tuple (O_i, P, T), where $O_i \subseteq O$ is a set of *objects*, P is a *relation* over the objects and T is a time interval. We propose a representation that further structures the event to have *roles* in addition to relations. Each event is composed of an **action** P, an **actor** O_1 that conducted the action, and an **object** O_2 on which the action was performed. Formally, an event is represented as $E = (O_1,\ P,\ O_2,\ T)$, where P is the action, O_1 is the actor, O_2 is the object and T is the timestamp (T is mainly used for aligning stock data with news data). For example, the event "Sep 3, 2013 - Microsoft agrees to buy Nokia's mobile phone business for $7.2 billion." is modeled as: (Actor = *Microsoft*, Action = *buy*, Object = *Nokia's mobile phone business*, Time = *Sep 3, 2013*).

Previous work on stock market prediction represents events as a set of individual terms (Fung et al., 2002; Fung et al., 2003; Hayo and Kutan, 2004; Feldman et al., 2011). For example, "Microsoft agrees to buy Nokia's mobile phone business for $7.2 billion." can be represented by {"Microsoft", "agrees", "buy", "Nokia's", "mobile", ...} and "Oracle has filed suit against Google over its ever-more-popular mobile operating system, Android." can be represented by {"Oracle", "has", "filed", "suit", "against", "Google", ...}. However, terms alone might fail to accurately predict the stock price movement of *Microsoft*, *Nokia*, *Oracle* and *Google*, because they cannot indicate the actor and object of the event. To our knowledge, no effort has been reported in the literature to empirically investigate structured event representations for stock market prediction.

2.2 Event Extraction

A main contribution of our work is to extract and use structured events instead of bags-of-words in prediction models. However, structured event extraction can be a costly task, requiring predefined event types and manual event templates (Ji and Grishman, 2008; Li et al., 2013). Partly due to this, the bags-of-words-based document representation has been the mainstream method for a long time. To tackle this issue, we resort to Open IE, extracting event tuples from wide-coverage data with-

out requiring any human input (e.g. templates). Our system is based on the system of Fader et al. (2011) and the work of Ding et al. (2013); it does not require predefined target event types and labeled training examples. Given a natural language sentence obtained from news texts, the following procedure is used to extract structured events:

1. **Event Phrase Extraction.** We extract the predicate verb P of a sentence based on the dependency parser of Zhang and Clark (2011), and then find the longest sequence of words P_v, such that P_v starts at P and satisfies the syntactic and lexical constraints proposed by Fader et al. (2011). The content of these two constraints are as follows:
   - **Syntactic constraint:** every multi-word event phrase must begin with a verb, end with a preposition, and be a contiguous sequence of words in the sentence.
   - **Lexical constraint:** an event phrase should appear with at least a minimal number of distinct argument pairs in a large corpus.
2. **Argument Extraction.** For each event phrase P_v identified in the step above, we find the nearest noun phrase O_1 to the left of P_v in the sentence, and O_1 should contain the subject of the sentence (if it does not contain the subject of P_v, we find the second nearest noun phrase). Analogously, we find the nearest noun phrase O_2 to the right of P_v in the sentence, and O_2 should contain the object of the sentence (if it does not contain the object of P_v, we find the second nearest noun phrase).

An example of the extraction algorithm is as follows. Consider the sentence,

Instant view: Private sector adds 114,000 jobs in July: ADP.

The predicate verb is identified as "adds", and its subject and object "sector" and "jobs", respectively. The structured event is extracted as (*Private sector*, *adds*, *114,000 jobs*).

2.3 Event Generalization

Our goal is to train a model that is able to make predictions based on various expressions of the same event. For example, "Microsoft swallows Nokia's phone business for $7.2 billion" and "Microsoft purchases Nokia's phone business" report the same event. To improve the accuracy of our prediction model, we should endow the event extraction algorithm with generalization capacity. To this end, we leverage knowledge from two well-known ontologies, WordNet (Miller, 1995) and VerbNet (Kipper et al., 2006). The process of event generalization consists of two steps. First, we construct a morphological analysis tool based on the WordNet stemmer to extract lemma forms of inflected words. For example, in "Instant view: Private sector adds 114,000 jobs in July.", the words "adds" and "jobs" are transformed to "add" and "job", respectively. Second, we generalize each verb to its class name in VerbNet. For example, "add" belongs to the *multiply* class. After generalization, the event (*Private sector*, *adds*, *114,000 jobs*) becomes (*private sector*, *multiply_class*, *114,000 job*). Similar methods on event generalization have been investigated in Open IE based event causal prediction (Radinsky and Horvitz, 2013).

2.4 Prediction Models

1. Linear model. Most previous work uses linear models to predict the stock market (Fung et al., 2002; Luss and d'Aspremont, 2012; Schumaker and Chen, 2009; Kogan et al., 2009; Das and Chen, 2007; Xie et al., 2013). To make direct comparisons, this paper constructs a linear prediction model by using Support Vector Machines (SVMs), a state-of-the-art classification model. Given a training set $(d_1,\ y_1)$, $(d_2,\ y_2)$, ..., $(d_N,\ y_N)$, where $n \in [1,\ N]$, d_n is a news document and $y_i \in \{+1,\ -1\}$ is the output class. d_n can be news titles, news contents or both. The output Class +1 represents that the stock price will increase the next day/week/month, and the output Class -1 represents that the stock price will decrease the next day/week/month. The features can be bag-of-words features or structured event features. By SVMs, $y = \arg\max\{Class + 1,\ Class - 1\}$ is determined by the linear function $\boldsymbol{w} \cdot \boldsymbol{\Phi}(d_n, y_n)$, where $\boldsymbol{w}$ is the feature weight vector, and $\boldsymbol{\Phi}(d_n, y_n)$ is a function that maps d_n into a *M*-dimensional feature space. Feature templates will be discussed in the next subsection.

2. Nonlinear model. Intuitively, the relationship between events and the stock market may be more complex than linear, due to hidden and indirect

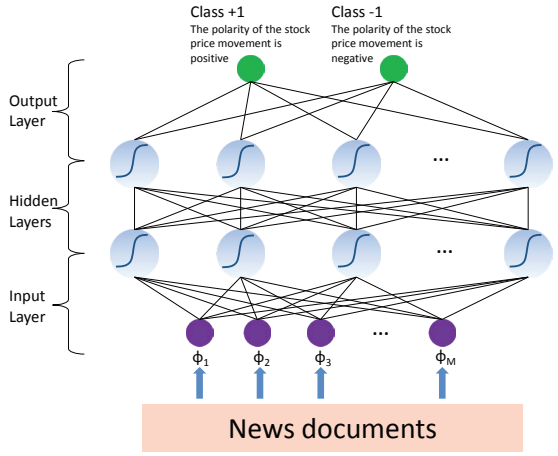


Figure 2: Structure of the deep neural network model

relationships. We exploit a deep neural network model, the hidden layers of which is useful for learning such hidden relationships. The structure of the model with two hidden layers is illustrated in Figure 2. In all layers, the sigmoid activation function σ is used.

Let the values of the neurons of the output layer be y_{cls} ($cls \in \{+1, -1\}$), its input be net_{cls}, and $\mathbf{y_2}$ be the value vector of the neurons of the last hidden layer; then:

$$y_{cls} = f(net_{cls}) = \sigma(\boldsymbol{w_{cls}} \cdot \mathbf{y_2}) \quad (1)$$

where $\boldsymbol{w_{cls}}$ is the weight vector between the neuron *cls* of the output layer and the neurons of the last hidden layer. In addition,

$$\begin{aligned} y_{2k} &= \sigma(\boldsymbol{w_{2k}} \cdot \mathbf{y_1}) \quad (k \in [1, |y_2|]) \\ y_{1j} &= \sigma(\boldsymbol{w_{1j}} \cdot \mathbf{\Phi}(d_n)) \quad (j \in [1, |y_1|]) \end{aligned} \quad (2)$$

Here $\mathbf{y_1}$ is the value vector of the neurons of the first hidden layer, $\boldsymbol{w_{2k}} = (w_{2k1}, w_{2k2}, ..., w_{2k|y_1|})$, $k \in [1, |y_2|]$ and $\boldsymbol{w_{1j}} = (w_{1j1}, w_{1j2}, ..., w_{1jM})$, $j \in [1, |y_1|]$. w_{2kj} is the weight between the *k*th neuron of the last hidden layer and the *j*th neuron of the first hidden layer; w_{1jm} is the weight between the *j*th neuron of the first hidden layer and the *m*th neuron of the input layer $m \in [1, M]$; d_n is a news document and $\mathbf{\Phi}(d_n)$ maps d_n into a *M*-dimensional features space. News documents and features used in the nonlinear model are the same as those in the linear model, which will be introduced in details in the next subsection. The standard back-propagation algorithm (Rumelhart et al., 1985) is used for supervised training of the neural network.

	train	dev	test
number of instances	1425	178	179
number of events	54776	6457	6593
time interval	02/10/2006 - 18/16/2012	19/06/2012 - 21/02/2013	22/02/2013 - 21/11/2013

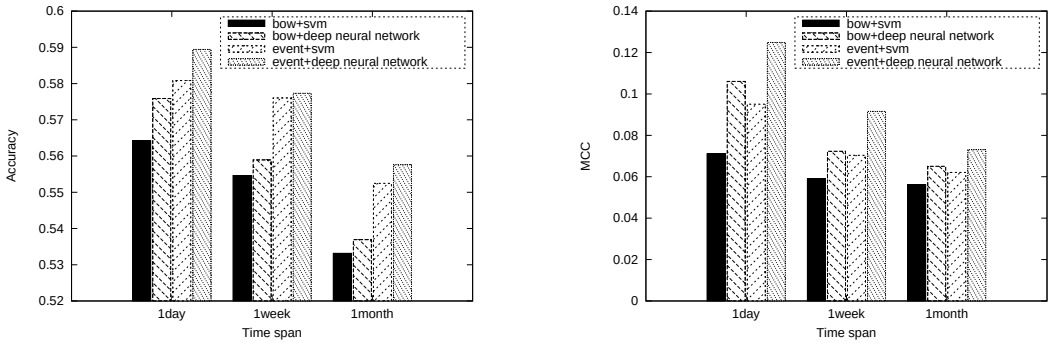
Table 1: Dataset splitting

2.5 Feature Representation

In this paper, we use the same features (i.e. document representations) in the linear and nonlinear prediction models, including bags-of-words and structured events.

(1) Bag-of-words features. We use the classic "TFIDF" score for bag-of-words features. Let L be the vocabulary size derived from the training data (introduced in the next section), and $freq(t_l)$ denote the number of occurrences of the *l*th word in the vocabulary in document d. $\mathbf{TF_l} = \frac{1}{|d|} freq(t_l)$, $\forall l \in [1, L]$, where $|d|$ is the number of words in the document d (stop words are removed). $\mathbf{TFIDF_l} = \frac{1}{|d|} freq(t_l) \times log(\frac{N}{|\{d: freq(t_l) > 0\}|})$, where N is the number of documents in the training set. The feature vector $\mathbf{\Phi}$ can be represented as $\mathbf{\Phi} = (\phi_1, \phi_2, ..., \phi_M) = (TFIDF_1, TFIDF_2, ..., TFIDF_M)$. The TFIDF feature representation has been used by most previous studies on stock market prediction (Kogan et al., 2009; Luss and d'Aspremont, 2012).

(2) Event features. We represent an event tuple (O_1, P, O_2, T) by the combination of elements (except for T) $(O_1, P, O_2, O_1 + P, P + O_2, O_1 + P + O_2)$. For example, the event tuple (Microsoft, buy, Nokia's mobile phone business) can be represented as (*#arg1*=Microsoft, *#action*=get_class, *#arg2*=Nokia's mobile phone business, *#arg1_action*=Microsoft get_class, *#action_arg2*=get_class Nokia's mobile phone business, *#arg1_action_arg2*=Microsoft get_class Nokia's mobile phone business). Structured events are more sparse than words, and we reduce sparseness by two means. First, verb classes (Section 2.3) are used instead of verbs for *P*. For example, "get_class" is used instead of the verb "buy". Second, we use back-off features, such as $O_1 + P$ ("Microsoft get_class") and $P + O_2$ ("get_class Nokia's mobile phone business"), to address the sparseness of O_1 and O_2. Note that the order of O_1 and O_2 is important for our task since they indicate the actor and object, respectively.



(a) Accuarcy (b) MCC

Figure 3: Overall development experiment results

3 Experiments

Our experiments are carried out on three different time intervals: short term (1 day), medium term (1 week) and long term (1 month). We test the influence of events on predicting the polarity of stock change for each time interval, comparing the event-based news representation with bag-of-words-based news representations, and the deep neural network model with the SVM model.

3.1 Data Description

We use publicly available financial news from Reuters and Bloomberg over the period from October 2006 to November 2013. This time span witnesses a severe economic downturn in 2007-2010, followed by a modest recovery in 2011-2013. There are 106,521 documents in total from Reuters News and 447,145 from Bloomberg News. News titles and contents are extracted from HTML. The timestamps of the news are also extracted, for alignment with stock price information. The data size is larger than most previous work in the literature.

We mainly focus on predicting the change of the Standard & Poor's 500 stock (S&P 500) index[3], obtaining indices and stock price data from Yahoo Finance. To justify the effectiveness of our prediction model, we also predict price movements of fifteen individual shares from different sectors in S&P 500. We automatically align 1,782 instances of daily trading data with news titles and contents from the previous day/the day a week before the stock price data/the day a month before the stock price data, 4/5 of which are used as the training data, 1/10 for development testing and 1/10 for testing. As shown in Table 1, the training, development and test set are split temporally, with the data from 02/10/2006 to 18/16/2012 for training, the data from 19/06/2012 to 21/02/2013 for development testing, and the data from 22/02/2013 to 21/11/2013 for testing. There are about 54,776 events in the training set, 6,457 events in the development set and 6,593 events in the test set.

[3]Standard & Poor's 500 is a stock market index based on the market capitalizations of 500 large companies having common stock listed on the NYSE or NASDAQ.

3.2 Evaluation Metrics

We use two assessment metrics. First, a standard and intuitive approach to measuring the performance of classifiers is accuracy. However, this measure is very sensitive to data skew: when a class has an overwhelmingly high frequency, the accuracy can be high using a classifier that makes prediction on the majority class. Previous work (Xie et al., 2013) uses an additional evaluation metric, which relies on the Matthews Correlation Cofficient (MCC) to avoid bias due to data skew (our data are rather large and not severely skewed, but we also use MCC for comparison with previous work). MCC is a single summary value that incorporates all 4 cells of a 2*2 confusion matrix (True Positive, False Positive, True Negative and False Negative, respectively). Given TP, TN, FP and FN:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (3)$$

3.3 Overall Development Results

We evaluate our four prediction methods (i.e. SVM with bag-of-word features (bow), deep neural network with bag-of-word features (bow),

		1 day	1 week	1 month
1 layer	Accuracy	58.94%	57.73%	55.76%
	MCC	0.1249	0.0916	0.0731
2 layers	Accuracy	59.60%	57.73%	56.19%
	MCC	0.1683	0.1215	0.0875

Table 2: Different numbers of hidden layers

	title	content	content + title	bloomberg title + title
Acc	59.60%	54.65%	56.83%	59.64%
MCC	0.1683	0.0627	0.0852	0.1758

Table 3: Different amounts of data

Google Inc.					
Company News		Sector News		All News	
Acc	MCC	Acc	MCC	Acc	MCC
67.86%	0.4642	61.17%	0.2301	55.70%	0.1135
Boeing Company					
Company News		Sector News		All News	
Acc	MCC	Acc	MCC	Acc	MCC
68.75%	0.4339	57.14%	0.1585	56.04%	0.1605
Wal-Mart Stores					
Company News		Sector News		All News	
Acc	MCC	Acc	MCC	Acc	MCC
70.45%	0.4679	62.03%	0.2703	56.04%	0.1605

Table 4: Individual stock prediction results

SVM with event features and deep neural network with event features) on three time intervals (i.e. 1 day, 1 week and 1 month, respectively) on the development dataset, and show the results in Figure 3. We find that:

(1) Structured event is a better choice for representing news documents. Given the same prediction model (SVM or deep neural network), the event-based method achieves consistently better performance than the bag-of-words-based method over all three time intervals. This is likely due to the following two reasons. First, being an extraction of predicate-argument structures, events carry the most essential information of the document. In contrast, bag-of-words can contain more irrelevant information. Second, structured events can directly give the actor and object of the action, which is important for predicting stock market.

(2) The deep neural network model achieves better performance than the SVM model, partly by learning hidden relationships between structured events and stock prices. We give analysis to these relationships in the next section.

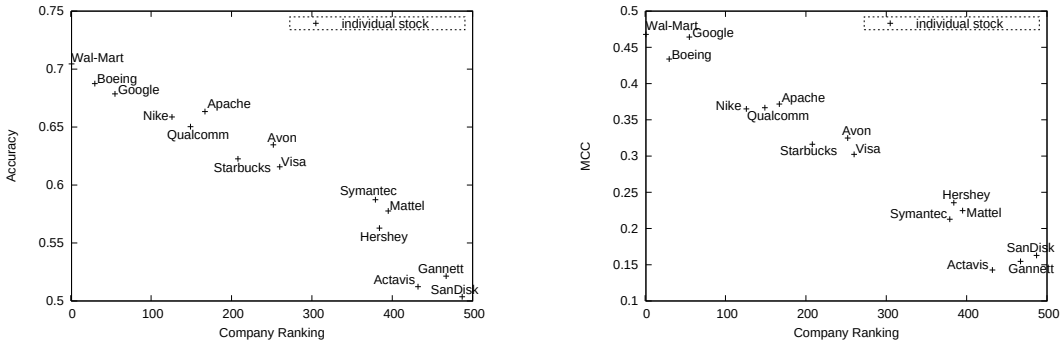
(3) Event information is a good indicator for short-term volatility of stock prices. As shown in Figure 3, the performance of daily prediction is better than weekly and monthly prediction. Our experimental results confirm the conclusion of Tetlock, Saar-Tsechansky, and Macskassy (2008) that there is a one-day delay between the price response and the information embedded in the news. In addition, we find that some events may cause immediate changes of stock prices. For example, former Microsoft CEO Steve Ballmer announced he would step down within 12 months on 23/08/2013. Within an hour, Microsoft shares jumped as much as 9 percent. This fact indicates that it may be possible to predict stock price movement on a shorter time interval than one day. However, we cannot access fine-grained stock price historical data, and this investigation will be left as future work.

3.4 Experiments with Different Numbers of Hidden Layers of the Deep Neural Network Model

Cybenko (1989) states that when every processing element utilizes the sigmoid activation function, one hidden layer is enough to solve any discriminant classification problem, and two hidden layers are capable to parse arbitrary output functions of input pattern. Here we conduct a development experiment by different number of hidden layers for the deep neural network model. As shown in Table 2, the performance of two hidden layers is better than one hidden layer, which is consistent with the experimental results of Sharda and Delen (2006) on the task of movie box-office prediction. It indicates that more hidden layers can explain more complex relations (Bengio, 2009). Intuitively, three or more hidden layers may achieve better performance. However, three hidden layers mean that we construct a five-layer deep neural network, which is difficult to train (Bengio et al., 1994). We did not obtain improved accuracies using three hidden layers, due to diminishing gradients. A deep investigation of this problem is out of the scope of this paper.

3.5 Experiments with Different Amounts of Data

We conduct a development experiment by extracting news titles and contents from Reuters and Bloomberg, respectively. While titles can give the central information about the news, contents may provide some background knowledge or details. Radinsky et al. (2012) argued that news titles are more helpful for prediction compared to news con-



(a) Accuarcy (b) MCC

Figure 4: Individual stock prediction experiment results

tents, and this paper mainly uses titles. Here we design a comparative experiment to analyze the effectiveness of news titles and contents. First, we use Reuters news to compare the effectiveness of news titles and contents, and then add Bloomberg news titles to investigate whether the amounts of data matters. Table 3 shows that using only news titles achieves the best performance. A likely reason is that we may extract some irrelevant events from news contents.

With the additional Bloomberg data, the results are not dramatically improved. This is intuitively because most events are reported by both Reuters news and Bloomberg news. We randomly select about 9,000 pieces of news documents from Reuters and Bloomberg and check the daily overlap manually, finding that about 60% of the news are reported by both Reuters and Bloomberg. The overlap of important news (news related to S&P 500 companies) is 80% and the overlap of unimportant news is 40%.

3.6 Individual Stock Prediction

In addition to predicting the S&P 500 index, we also investigate the effectiveness of our approach on the problem of individual stock prediction using the development dataset. We select three well-known companies, *Google Inc.*, *Boeing Company* and *Wal-Mart Stores* from three different sectors (i.e. Information Technology, Industrials and Consumer Staples, respectively) classified by the Global Industry Classification Standard (GICS). We use company news, sector news and all news to predict individual stock price movement, respectively. The experimental results are listed in Table 4.

The result of individual stock prediction by using only company news dramatically outperforms the result of S&P 500 index prediction. The main reason is that company-related events can directly affect the volatility of company shares. There is a strong correlation between company events and company shares. Table 4 also shows that the result of individual stock prediction by using sector news or all news does not achieve a good performance, probably because there are many irrelevant events in all news, which would reduce the performance of our prediction model.

The fact that the accuracy of these well-known stocks are higher than the index may be because there is relatively more news events dedicated to the relevant companies. To gain a better understanding of the behavior of the model on more individual stocks, we randomly select 15 companies (i.e. *Google Inc.*, *Boeing Company*, *Wal-Mart Stores*, *Nike Inc.*, *QUALCOMM Inc.*, *Apache Corporation*, *Starbucks Corp.*, *Avon Products*, *Visa Inc.*, *Symantec Corp.*, *The Hershey Company*, *Mattel Inc.*, *Actavis plc*, *Gannett Co.* and *SanDisk Corporation*) from S&P 500 companies. More specifically, according to the Fortune ranking of S&P 500 companies[4], we divide the ranked list into five parts, and randomly select three companies from each part. The experimental results are shown in Figure 4. We find that:

(1) All 15 individual stocks can be predicted with accuracies above 50%, while 60% of the stocks can be predicted with accuracies above 60%. It shows that the amount of company-related events has strong relationship with the volatility of

[4]http://money.cnn.com/magazines/fortune/fortune500/. The amount of company-related news is correlated to the fortune ranking of companies. However, we find that the trade volume does not have such a correlation with the ranking.

	S&P 500 Index Prediction		Individual Stock Prediction					
			Google Inc.		Boeing Company		Wal-Mart Stores	
	Accuracy	MCC	Accuracy	MCC	Accuracy	MCC	Accuracy	MCC
dev	59.60%	0.1683	67.86%	0.4642	68.75%	0.4339	70.45%	0.4679
test	58.94%	0.1649	66.97%	0.4435	68.03%	0.4018	69.87%	0.4456

Table 5: Final experimental results on the test dataset

company shares.

(2) With decreasing company fortune rankings, the accuracy and MCC decrease. This is mainly because there is not as much daily news about low-ranking companies, and hence one cannot extract enough structured events to predict the volatility of these individual stocks.

3.7 Final Results

The final experimental results on the test dataset are shown in Table 5 (as space is limited, we show the results on the time interval of one day only). The experimental results on the development and test datasets are consistent, which indicate that our approach has good robustness. The following conclusions obtained from development experiments also hold on the test dataset:

(1) Structured events are more useful representations compared to bags-of-words for the task of stock market prediction.

(2) A deep neural network model can be more accurate on predicting the stock market compared to the linear model.

(3) Our approach can achieve stable experiment results on S&P 500 index prediction and individual stock prediction over a large amount of data (eight years of stock prices and more than 550,000 pieces of news).

(4) The quality of information is more important than the quantity of information on the task of stock market prediction. That is to say that the most relevant information (i.e. news title *vs* news content, individual company news *vs* all news) is better than more, but less relevant information.

3.8 Analysis and Discussion

We use Figure 5 to demonstrate our analysis to the development experimental result of *Google Inc.* stock prediction, which directly shows the relationship between structured events and the stock market. The links between each layer show the magnitudes of feature weights in the model learned using the training set.

Three events, (Google, says bought stake in, China's XunLei), (Google, reveals stake in, Chinese social website) and (Capgemini, partners, Google apps software), have the highest link weights to the first hidden node (from the left). These three events indicate that Google constantly makes new partners and expands its business area. The first hidden node has high-weight links to Class +1, showing that Google's positive cooperation can lead to the rise of its stock price.

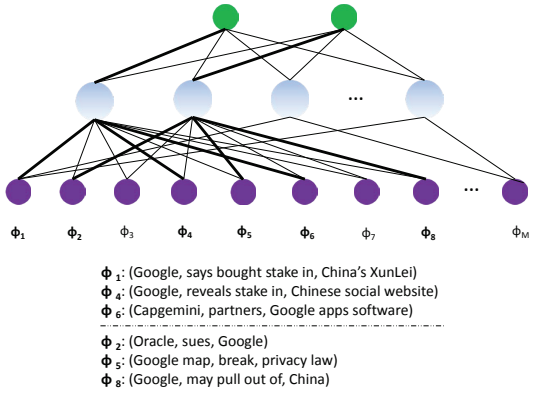


Figure 5: Prediction of *Google Inc.* (we only show structured event features since backoff features are less informative)

Three other events, (Oracle, sues, Google), (Google map, break, privacy law) and (Google, may pull out of, China), have high-weight links to the second hidden node. These three events show that Google was suffering questions and challenges, which could affect its reputation and further pull down its earnings. Correspondingly, the second hidden node has high-weight links to Class -1. These suggest that our method can automatically and directly reveal complex relationships between structured events and the stock market, which is very useful for investors, and can facilitate the research of stock market prediction.

Note that the event features used in our prediction model are generalized based on the algorithm introduced in Section 2.5. Therefore, though a specific event in the development test set might have never happened, its generalized form can be found in the training set. For example, "Google acquired social marketing company Wildfire In-

teractive" is not in the training data, but "Google get_class" ("get" is the class name of "acquire" and "buy" in VerbNet) can indeed be found in the training set, such as "Google bought stake in XunLei" on 04/01/2007. Hence although the full specific event feature does not fire, its back-offs fire for a correct prediction. For simplicity of showing the event, we did not include back-off features in Figure 5.

4 Related Work

Stock market prediction has attracted a great deal of attention across the fields of finance, computer science and other research communities in the past. The literature of stock market prediction was initiated by economists (Keynes, 1937). Subsequently, the influential theory of Efficient Market Hypothesis (EMH) (Fama, 1965) was established, which states that the price of a security reflects all of the information available and that everyone has a certain degree of access to the information. EMH had a significant impact on security investment, and can serve as the theoretical basis of event-based stock price movement prediction.

Various studies have found that financial news can dramatically affect the share price of a security (Chan, 2003; Tetlock et al., 2008). Culter et al. (1998) was one of the first to investigate the relationship between news coverage and stock prices, since which empirical text analysis technology has been widely used across numerous disciplines (Lavrenko et al., 2000; Kogan et al., 2009; Luss and d'Aspremont, 2012). These studies primarily use bags-of-words to represent financial news documents. However, as Schumaker and Chen (2009) and Xie et al. (2013) point out, bag-of-words features are not the best choice for predicting stock prices. Schumaker and Chen (2009) extract noun phrases and named entities to augment bags-of-words. Xie et al. (2013) explore a rich feature space that relies on frame semantic parsing. Wang et al. (2014) use the same features as Xie et al. (2013), but they perform non-parametric kernel density estimation to smooth out the distribution of features. These can be regarded as extensions to the bag-of-word method. The drawback of these approaches, as discussed in the introduction, is that they do not directly model events, which have structured information.

There has been efforts to model events more directly (Fung et al., 2002; Hayo and Kutan, 2005; Feldman et al., 2011). Fung, Yu, and Lam (2002) use a normalized word vector-space to model event. Feldman et al. (2011) extract 9 predefined categories of events based on heuristic rules. There are two main problems with these efforts. First, they cannot extract structured event (e.g. the actor of the event and the object of the event). Second, Feldman et al. (2011) can obtain only limited categories of events, and hence the scalability of their work is not strong. In contrast, we extract structured events by leveraging Open Information Extraction technology (Open IE; Yates et al. (2007); Etzioni et al. (2011); Faber et al. (2011)) without predefined event types, which can effectively solve the two problems above.

Apart from events, sentiment analysis is another perspective to the problem of stock prediction (Das and Chen, 2007; Tetlock, 2007; Tetlock et al., 2008; Bollen et al., 2011; Si et al., 2013). Tetlock (2007) examines how qualitative information (i.e. the fraction of negative words in a particular news column) is incorporated in aggregate market valuations. Tetlock, Saar-Tsechansky, and Macskassy (2008) extend that analysis to address the impact of negative words in all Wall Street Joural (WSJ) and Dow Jones News Services (DJNS) stories about individual S&P500 firms from 1980 to 2004. Bollen and Zeng (2011) study whether the large-scale collective emotion on Twitter is correlated with the volatility of Dow Jones Industrial Average (DJIA). From the experimental results, they find that changes of the public mood match shifts in the DJIA values that occur 3 to 4 days later. Sentiment-analysis-based stock market prediction focuses on investigating the influence of subjective emotion. However, this paper puts emphasis on the relationship between objective events and the stock price movement, and is orthogonal to the study of subjectivity. As a result, our model can be combined with the sentiment-analysis-based method.

5 Conclusion

In this paper, we have presented a framework for event-based stock price movement prediction. We extracted structured events from large-scale news based on Open IE technology and employed both linear and nonlinear models to empirically investigate the complex relationships between events and the stock market. Experimental results showed that events-based document representations are

better than bags-of-words-based methods, and deep neural networks can model the hidden and indirected relationship between events and the stock market. For further comparisons, we freely release our data at http://ir.hit.edu.cn/∼xding/data.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, and gratefully acknowledge the support of the National Basic Research Program (973 Program) of China via Grant 2014CB340503, the National Natural Science Foundation of China (NSFC) via Grant 61133012 and 61202277, the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301 and SRG ISTD 2012 038 from Singapore University of Technology and Design. We are very grateful to Ji Ma for providing an implementation of the neural network algorithm.

References

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction for the web. In *IJCAI*, volume 7, pages 2670–2676.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

Werner FM Bondt and Richard Thaler. 1985. Does the stock market overreact? *The Journal of finance*, 40(3):793–805.

Wesley S Chan. 2003. Stock price reaction to news and no-news: Drift and reversal after headlines. *Journal of Financial Economics*, 70(2):223–260.

David M Cutler, James M Poterba, and Lawrence H Summers. 1998. What moves stock prices? *Bernstein, Peter L. and Frank L. Fabozzi*, pages 56–63.

George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Sanjiv R Das and Mike Y Chen. 2007. Yahoo! for amazon: Sentiment extraction from small talk on the web. *Management Science*, 53(9):1375–1388.

Xiao Ding, Bing Qin, and Ting Liu. 2013. Building chinese event type paradigm based on trigger clustering. In *Proc. of IJCNLP*, pages 311–319, October.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, pages 3–10. AAAI Press.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Eugene F Fama. 1965. The behavior of stock-market prices. *The journal of Business*, 38(1):34–105.

Ronen Feldman, Benjamin Rosenfeld, Roy Bar-Haim, and Moshe Fresko. 2011. The stock sonarsentiment analysis of stocks based on a hybrid approach. In *Twenty-Third IAAI Conference*.

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Wai Lam. 2002. News sensitive stock trend prediction. In *Advances in Knowledge Discovery and Data Mining*, pages 481–493. Springer.

Bernd Hayo and Ali M Kutan. 2005. The impact of news, oil prices, and global market developments on russian financial markets1. *Economics of Transition*, 13(2):373–393.

Narasimhan Jegadeesh and Sheridan Titman. 1993. Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance*, 48(1):65–91.

Narasimhan Jegadeesh. 1990. Evidence of predictable behavior of security returns. *The Journal of Finance*, 45(3):881–898.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*, pages 254–262.

John Maynard Keynes. 1937. The general theory of employment. *The Quarterly Journal of Economics*, 51(2):209–223.

Jaegwon Kim. 1993. *Supervenience and mind: Selected philosophical essays*. Cambridge University Press.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2006. Extending verbnet with novel verb classes. In *Proceedings of LREC*, volume 2006, page 1.

Shimon Kogan, Dimitry Levin, Bryan R. Routledge, Jacob S. Sagi, and Noah A. Smith. 2009. Predicting risk from financial reports with regression. In *Proc. NAACL*, pages 272–280, Boulder, Colorado, June. Association for Computational Linguistics.

Victor Lavrenko, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. 2000. Mining of concurrent text and time series. In *KDD-2000 Workshop on Text Mining*, pages 37–44.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proc. of ACL (Volume 1: Long Papers)*, pages 73–82, August.

Andrew W Lo and Archie Craig MacKinlay. 1990. When are contrarian profits due to stock market overreaction? *Review of Financial studies*, 3(2):175–205.

Ronny Luss and Alexandre d'Aspremont. 2012. Predicting abnormal returns from news using text classification. *Quantitative Finance*, pp.1–14, doi:10.1080/14697688.2012.672762.

Burton G. Malkiel. 1973. *A Random Walk Down Wall Street*. W. W. Norton, New York.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Kira Radinsky and Eric Horvitz. 2013. Mining the web to predict future events. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 255–264. ACM.

Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. 2012. Learning causality for news events prediction. In *Proc. of WWW*, pages 909–918. ACM.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical report, DTIC Document.

Robert P Schumaker and Hsinchun Chen. 2009. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):12.

Ramesh Sharda and Dursun Delen. 2006. Predicting box-office success of motion pictures with neural networks. *Expert Systems with Applications*, 30(2):243–254.

Jianfeng Si, Arjun Mukherjee, Bing Liu, Qing Li, Huayi Li, and Xiaotie Deng. 2013. Exploiting topic based twitter sentiment for stock prediction. In *Proc. of ACL (Volume 2: Short Papers)*, pages 24–29, Sofia, Bulgaria, August. Association for Computational Linguistics.

Paul C Tetlock, Maytal Saar-Tsechansky, and Sofus Macskassy. 2008. More than words: Quantifying language to measure firms' fundamentals. *The Journal of Finance*, 63(3):1437–1467.

Paul C Tetlock. 2007. Giving content to investor sentiment: The role of media in the stock market. *The Journal of Finance*, 62(3):1139–1168.

William Yang Wang and Zhenhao Hua. 2014. A semiparametric gaussian copula regression model for predicting financial risks from earnings calls. In *Proc. of ACL*, June.

Boyi Xie, Rebecca J. Passonneau, Leon Wu, and Germán G. Creamer. 2013. Semantic frames to predict stock price movement. In *Proc. of ACL (Volume 1: Long Papers)*, pages 873–883, August.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. Textrunner: open information extraction on the web. In *Proc. of NAACL: Demonstrations*, pages 25–26. Association for Computational Linguistics.

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

Extracting Clusters of Specialist Terms from Unstructured Text

Aaron Gerow
Computation Institute
University of Chicago
Chicago, IL, USA
gerow@uchicago.edu

Abstract

Automatically identifying related specialist terms is a difficult and important task required to understand the lexical structure of language. This paper develops a corpus-based method of extracting coherent clusters of satellite terminology — terms on the edge of the lexicon — using co-occurrence networks of unstructured text. Term clusters are identified by extracting communities in the co-occurrence graph, after which the largest is discarded and the remaining words are ranked by centrality within a community. The method is tractable on large corpora, requires no document structure and minimal normalization. The results suggest that the model is able to extract coherent groups of satellite terms in corpora with varying size, content and structure. The findings also confirm that language consists of a densely connected core (observed in dictionaries) and systematic, semantically coherent groups of terms at the edges of the lexicon.

1 Introduction

Natural language consists of a number of relational structures, many of which can be obscured by lexical idiosyncrasies, regional variation and domain-specific conventions. Despite this, patterns of word use exhibit loose semantic structure, namely that proximate words tend to be related. This distributional hypothesis has been operationalized in a variety of ways, providing insights and solutions into practical and theoretical questions about meaning, intention and the use of language. Distributional analyses rely primarily on observing natural language to build statistical representations of words, phrases and documents (Turney and Pantel, 2010). By studying dictionaries and thesauri, lexicographic and terminological research has proposed that a core lexicon is used to define the remaining portions of vocabulary (Itô and Mester, 1995; Massé et al., 2008). Though many words that comprise general language use reside in this core lexicon, even the most general language contains specialist or so-called "satellite" words. This paper introduces a method of extracting this peripheral structure, with co-occurrence networks of unstructured text.

The core-periphery structure has been observed in dictionaries where definitions tend to use a restricted vocabulary, repetitively employing a core set of words to define others (Sinclair, 1996; Picard et al., 2013). In the farther regions of the lexicon, it is more difficult to find systematic semantic definition with corpus-based techniques due to the overwhelming number of infrequent words. Unfortunately, the fringe of the lexicon can be more important than the core because this is where domain-specific terminology resides — features that may be more important than frequent.

Examining dictionaries, (Picard et al., 2013) propose that the lexicon consists of four main parts: a *core* set of ubiquitous words used to define other words, a *kernel* that makes up most of the lexicon, a *minimal grounding set* that includes most of the core and some of the kernel, leaving a set of *satellites* in the periphery. This topography, reproduced in Figure 1, has been found in the way dictionary entries use words to define one another. In networks of dictionary definitions, the core component tends to form a strongly connected component (SCC) leaving satellites in smaller SCCs with relatively weak links to the core. This paper explores whether these these satellites form systematic, cohesive groups and whether they are observable in natural language.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1426–1434,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

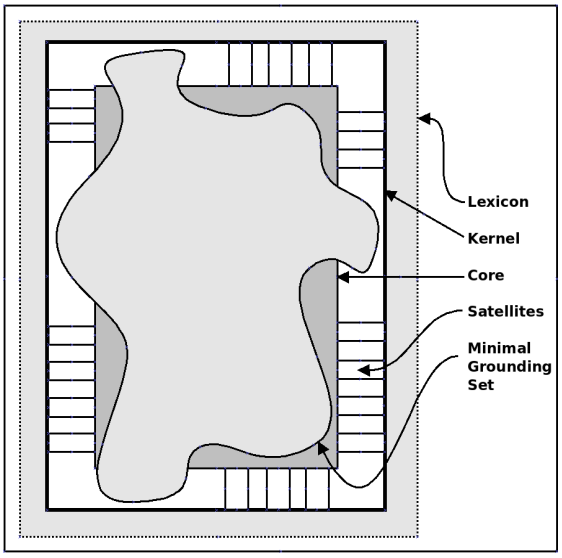


Figure 1: Dictionary studies have proposed that the lexicon consists of a strongly connected core, around which there is a kernel, an asymmetric grounding set and satellites. Adapted from (Picard et al., 2013).

Words with relatively specific definitions within subjects, referred to as *terms* in lexicographic research, are apparent in nearly all domains of discourse. Here, the goal is to explore structure among these peripheral terms without a dictionary. To do this, a method based on community detection in textual co-occurrence networks is developed. Such graph-based methods have become increasingly popular in a range of language-related tasks such as word clustering, document clustering, semantic memory, anaphora resolution and dependency parsing (see Mihalcea and Radev, 2011 for a review).

This paper seeks to address two important questions about the observed landscape of the lexicon in natural language: to investigate whether satellite clusters found in dictionaries can be observed in text, and more importantly, to explore whether statistical information in co-occurrence networks can elucidate this peripheral structure in the lexicon. We frame these questions as the task of extracting clusters of related terms. If satellites are systematically organized, then we can expect to find cohesive clusters in this region. Moreover, if the networked structure of dictionary entries supports the landscape in Figure 1, a similar structure may be present in co-occurrence patterns in natural text.

2 Method

Word clustering, as a means to explore underlying lexical structure, should accommodate fuzzy and potentially contradictory notions of similarity. For example, *red* and *green* are at once similar, being colors, but as colors, they are very different. Alternatively, the words *car*, *fast*, *wheel*, *export* and *motorists* share a thematic similarity in their relation to automobiles. One conception of word clustering is to construct a thesaurus of synonyms (Calvo et al., 2005), but clustering could allow other lexical semantic relationships. One such database, WordNet, defines specific semantic relationships and has been used to group words according to explicit measures of relatedness and similarity (Miller, 1995; Pedersen et al., 2004). Distributional, corpus-based techniques that define words as feature vectors (eg. word-document co-occurrences), can address many limitations of manually created lexicons (see Turney et al., 2010 for a review). Clustering nouns by argument structure can uncover naturally related objects (Hindle, 1990) and spectral methods can relate distinct classes of nouns with certain kinds of verbs to induce selectional preferences (Resnik, 1997; Sun and Korhonen, 2009; Wilks, 1975) and assist metaphor processing (Shutova et al., 2013).

A pervasive weakness of many existing approaches to word-clustering, is an underlying prioritization of frequent words. To help address this sparsity, many models collapse words into stems, preclude uncommon words, or underestimate the relevance of infrequent words (Dagan et al., 1999). Probabilistic topic models have emerged as a uniquely flexible kind of word-clustering used in content analysis (Steyvers and Griffiths, 2007), text classification (Wei and Croft, 2006) and provide an extensible framework to address other tasks (Blei, 2012). Because the structure of satellite terms is not likely to rely on specific (much less consistent) lexical semantic relationships, we adopt a measure of *semantic coherence*, commonly used to qualify the results of topic models, as an indirect measure of what people tend to view as a cohesive set of words. This measure, which is defined in the next section, is particularly attractive because it is corpus-based, does not assume any specific semantic relationship and correlates with expert evaluations (Mimno et al., 2011; Newman et al., 2010). Using semantic coherence provides a way of measuring the qual-

ity of word-associations without appeal to a dictionary or assuming rigid relationships among the clustered words.

The first step is to construct a co-occurrence graph from which communities are extracted. Then the centrality of each word is computed within a community to generate cluster-specific rankings. The goal is not to categorize words into classes, nor to provide partitions that separate associated words across a corpus. Instead, the method is designed to extract qualifiable sets of specialist terms found in arbitrary text. Crucially, the method is designed to require no document structure and minimal pre-processing: stop-words and non-words are not removed and no phrasal, sentence or document structure is required. Although stemming or lemmatization could provide more stream-lined interpretations, the minimal pre-processing allows the method to operate efficiently on large amounts of unstructured text of any language.

Co-occurrence networks have been used in a variety of NLP applications, the basic idea being to construct a graph where proximate words are connected. Typically, words are connected if they are observed in an n-word window. We set this window to a symmetric 7 words on either side of the target and did not use any weighting[1]. In the resulting network, edge frequencies are set to the number of times the given co-occurrence is observed. The resulting networks are typically quite dense and exhibit small-world structure where most word-pairs are only a few edges apart (Baronchelli et al., 2013; Ferror i Cancho and Solé, 2001). To explore the effect of this density, different minimum node- and edge-frequencies were tested (analogous to the word- and co-occurrence frequencies in text). It was found that not setting any thresholds provided the best results (see Figure 2), supporting our minimal pre-processing approach.

To extract clusters from the co-occurrence matrix, the Infomap community detection algorithm was used. Infomap is an information-theoretic method that optimizes a compression dictionary using it to describe flow through connected nodes (Rosvall and Bergstrom, 2008). By minimizing a description of this flow, the algorithm can also extract nested communities (Rosvall and Bergstrom, 2011). In our experiments, we used the co-occurrence frequencies as edge-weights and ran 50 trials for each run of the algorithm. Co-occurrence networks tended to form one monolithic community, corresponding to the lexicon's core SCC, surrounded by a number of smaller communities. The monolithic community is discarded out-right, as it represents the core of the lexicon where few specialist terms reside. As we will see, the community detection algorithm naturally identifies this SCC, distinguishing satellite clusters of terminology. Though we do not explore its effect, the sensitivity of Infomap can be tuned to vary the relative size of the core SCC compared to the satellites, effectively allowing less modular communities to be considered satellites.

[1]7 was found to be the optimal window-size in terms of coherence. These preliminary results are available at knowledgelab.org/docs/coherent_clusters-data.xls.

Corpus	Docs	Tokens	Nodes	Edges
TASA	38,972	10.7M	58,357	1,319,534
NIPS	3,742	5.2M	28,936	1,612,659
enTenTen	92,327	72.2M	69,745	7,721,413

Table 1: Co-occurrence networks of each corpus.

To compare and interpret the resulting clusters, various measures of centrality were tested for ranking words within their communities. The goal of this ranking is to find words that typify or define their community without assuming its underlying semantics. The results in the next section show that a number of common centrality measures work comparably well for this task. The final output of the system is a set of communities, in which words are ranked by their centrality.

3 Results & Analysis

Three corpora were used for evaluation: the TASA, NIPS and enTenTen collections. TASA consists of paragraph-length excerpts from high-school level, American English texts (Landauer and Dumais, 1997). The NIPS collection contains 17 volumes of annual proceedings from the conference of the same name. The enTenTen corpus is a web-based collection of text-heavy, English web-sites. Table 1 summarizes the collections and their co-occurrence networks.

The extracted communities, which consist of word-centrality pairs, are similarly structured to the output of topic models. Because appeals to human judgement are expensive and can introduce issues of consistency (Chang et al., 2009; Hu et al., 2011), a corpus-based measure of semantic coherence has been proposed (Mimno et al., 2011). Co-

herence is used as a proxy for human judgments. A general form of semantic coherence can be defined as the mean pair-wise similarity over the top n words in a topic or cluster t

$$C(t) = \frac{1}{n} \sum_{\substack{(w_i,w_j)\in t \\ i<j}}^{n} S(w_i, w_j)$$

where S is a symmetric measure of similarity. Newman, et al. (2010) surveyed a number of similarity metrics and found that mean point-wise mutual information (PMI) correlated best to human judgements. PMI is a commonly used measure of how much more information co-occurring words convey together compared to their independent contributions (Church and Hanks, 1990; Bouma, 2009). Using PMI as S, we can define a version of coherence, known as *UCI Coherence*:

$$C_{UCI}(t) = \frac{1}{n} \sum_{\substack{(w_i,w_j)\in t \\ i<j}}^{n} \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)}$$

where $p(w)$ is estimated as relative frequency in a corpus: $\frac{f(w)}{\sum_i f(w_i)}$. Using coherence to optimize topic models, Mimno et al. (2011) found that a simplified measure, termed *UMass Coherence*, is more strongly correlated to human judgments than C_{UCI}. For topic t, C_{UMass} is defined as follows:

$$C_{UMass}(t) = \frac{1}{n} \sum_{\substack{(w_i,w_j)\in t \\ i<j}}^{n} \log \frac{D(w_i, w_j) + 1}{D(w_j)}$$

where $D(w)$ is the number of documents containing w, and $D(w, w')$ is the number of documents containing both w and w'. Note that D relies crucially on document segmentation in the reference corpus, which is not encoded in the co-occurrence networks derived by the method described above. Thus, though the networks being analyzed and the coherence scores are both based on co-occurrence information, they are distinct from one another. Following convention, we compute coherence for the top 10 words in a given community. C_{UMass} was used as the measure of semantic coherence. and D was computed over the TASA corpus, which means the resulting scores are not directly comparable to (Mimno et al., 2011), though comparisons to other published results are provided below.

3.1 Ranking Functions & Frequency Thresholds

After communities are extracted from the co-occurrence graph, words are ranked by their centrality in a community. Six centrality measures were tested as ranking functions: degree centrality, closeness centrality, eigenvector centrality, Pagerank, hub-score and authority-score (Friedl et al., 2010). Degree centrality uses a node's degree as its centrality under the assumption that highly connected nodes are central. Closeness centrality measures the average distance between a node and all other nodes, promoting nodes that are "close" to the rest of the network. Eigenvector centrality favors well-connected nodes that are themselves connected to well-connected nodes. Pagerank is similar to eigenvector centrality, but also promotes nodes that mediate connections between strongly connected nodes. Hub and authority scores measure interconnectedness (hubs) and connectedness to interconnected nodes (authorities). Figure 2 shows the average coherence, across all communities extracted from the TASA corpus, for each centrality measure. The average coherence scores are highest using hub-score, though not significantly better than auth-score, eigenvector centrality or closeness centrality. In the results that follow, hub-scores were used to rank nodes within communities.

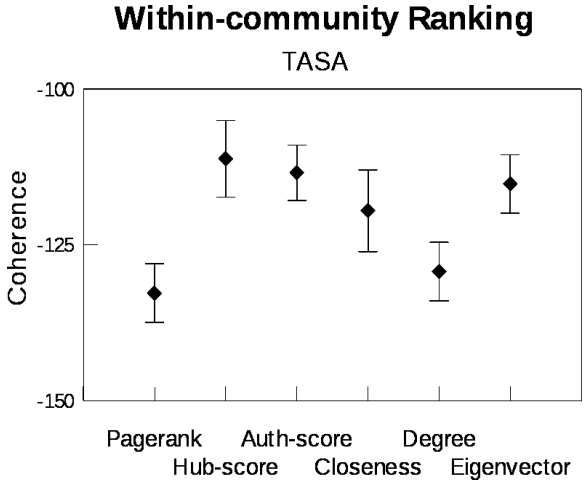


Figure 2: Mean coherence for six centrality measures. Error-bars are ±2 SE of the mean.

Imposing minimum node and edge frequencies in the co-occurrence graph was also tested. However, applying no thresholds provided the highest average coherence. Figure 3 shows the average coherence for eight threshold configurations. Though we used the TASA corpus for these tests, we have no reason to believe the results would differ significantly for the other corpora.

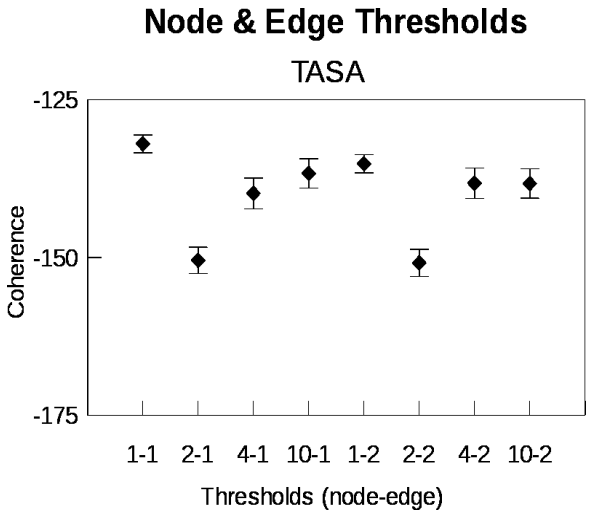


Figure 3: Mean coherence for different minimum node and edge frequencies, corresponding to thresholds for word and co-occurrence counts. Error-bars are ± 2 SE of the mean.

3.2 Community Coherence

Table 2 shows three communities of specialist terms extracted from each text collection, with their normalized hub-scores. Normalizing the scores preserves their rank-ordering and provides an indication of relative centrality within the community itself. For example, compare the first and last words from the top TASA and NIPS clusters: the difference between *thou* and *craven* (TASA) is considerably more than *model* and *network* (NIPS). In general, higher ranked words appear to typify their communities, with words like *model*, *university* and *nuclear* in the NIPS examples. These clusters are typical of those produced by the method, though in some cases, the communities contain less than 10 terms and were not included in the coherence analysis. Note that these clusters are not systematic in any lexical semantic sense, though in almost every case there are discernible thematic relations (middle-English words, Latin America and seafood in TASA).

TASA		**NIPS**		**enTenTen**	
thou	1.00	model	1.00	cortex	1.00
shalt	0.72	learning	0.99	prefrontal	0.88
hast	0.49	data	0.96	anterior	0.41
thyself	0.26	neural	0.94	cingulate	0.33
dost	0.24	using	0.85	medulla	0.28
wilt	0.24	network	0.85	parietal	0.13
canst	0.12	training	0.73	insula	0.13
knowest	0.10	algorithm	0.66	cruciate	0.11
mayest	0.10	function	0.63	striatum	0.11
craven	0.01	networks	0.62	ventral	0.10
peru	1.00	university	1.00	pradesh	1.00
ecuador	0.84	science	0.85	andhra	0.67
bolivia	0.80	computer	0.83	madhya	0.56
argentina	0.67	department	0.74	uttar	0.50
paraguay	0.54	engineering	0.30	bihar	0.21
chile	0.52	report	0.30	rajasthan	0.19
venezuela	0.48	technical	0.29	maharashtra	0.16
uruguay	0.28	institute	0.26	haryana	0.12
lima	0.17	abstract	0.25	himachal	0.10
parana	0.11	california	0.23	arunachal	0.04
clams	1.00	nuclear	1.00	cilia	1.00
crabs	0.87	weapons	0.66	peristomal	0.73
oysters	0.87	race	0.57	stalk	0.62
crab	0.67	countries	0.40	trochal	0.51
lobsters	0.66	rights	0.37	vorticella	0.35
shrimp	0.62	india	0.27	campanella	0.32
hermit	0.50	russia	0.26	hairlike	0.17
mussels	0.27	philippines	0.26	swimmers	0.15
lice	0.23	brazil	0.25	epistylis	0.12
scallops	0.20	waste	0.22	telotroch	0.11

Table 2: Sample clusters from the TASA, NIPS and enTenTen collections. Shown are the clusters' top ten words, ranked by their normalized hub-score within the community. Note the differences in hub-score distributions between clusters.

Figure 4 shows the average coherence for our method, compared to that of a 20-topic latent Dirichlet allocation (LDA) model fit to the same corpora. Results from an LDA model fit to our corpora, as well as from a sample of published topics, are provided as a baseline to calibrate readers' intuitions about coherence[2]. Although topics from LDA do not necessarily consist of specialist terms those in the current model, the expectation of coherence remains: probable or central words should comprise a cohesive group. In every case, coherence is calculated over the top 10 words ranked using within-community hub-scores, for every community of 10 or more words. The results show that LDA provides relatively consistent coherence across collections, though with generally more variance than the communities of specialist terms. The term clusters are more coherent for the enTenTen collection than the others, which may

[2]Coherence was computed for the published results with C_{UMass} using TASA as the reference corpus.

be due to its larger size. This up-tick on the largest corpus may have to do with the proportional size of the monolithic community for the less structured documents in enTenTen. Figure 5 depicts how the proportional size of the core would effect the number and size of satellite clusters. It was found that the largest community (the core SCC) comprised 95% of TASA, 90% of NIPS and 97% of enTenTen. It may be that specialized language will have a proportionally smaller core and more satellite communities, whereas more general language will have a larger core and fewer satellites.

A critical question remains as to whether the method is actually observing the core-periphery structure of the lexicon or if it is an artifact. To test this, the frequencies of words in satellite communities were compared to those in the monolithic cases. If the monolithic community does indeed correspond to the core proposed in Figure 1, words in the satellites should have significantly lower frequencies. Indeed, the monolithic community in every corpus contained words that were significantly more frequent than those in the communities (Wilcoxon rank-sum test; Table 3). Taken with the coherence scores, these results show that there is coherent structure in the periphery of the lexicon, that can be extracted from unstructured text.

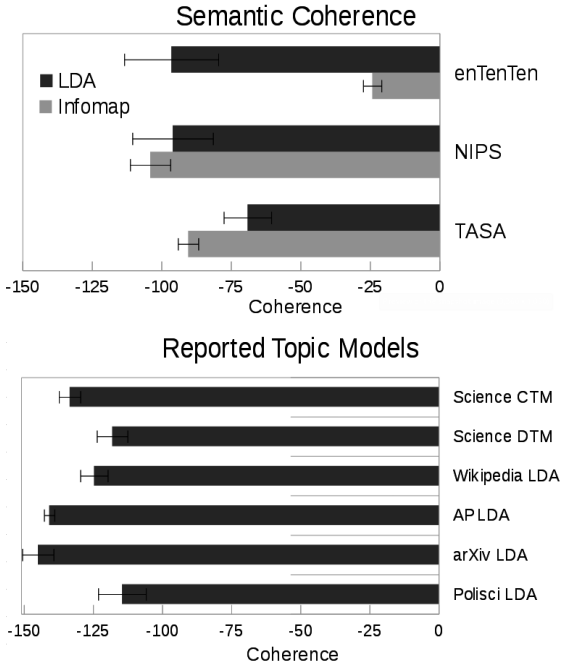


Figure 4: Mean coherence (C_{UMass}) for satellite clusters and topics from LDA on the TASA, NIPS and enTenTen collections (top). Also shown are the mean coherence of topics found in published models (LDA, a dynamic topic model, *DTM* and a correlated topic model, *CTM*; bottom). Error-bars are ± 2 SE of the mean.

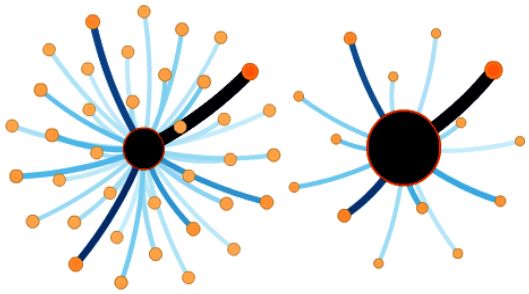


Figure 5: A proportionally larger core SCC (right) would force satellite communities to be smaller, less numerous and more isolated. Alternatively, with a small core (left), satellite communities would be more numerous and prominent.

Corpus	mean f_c	mean f_s	W	df
TASA	112.3	7.3	39985454	40895
NIPS	211.5	10.7	28342663	25077
enTenTen	365.1	15.9	246095083	72695

Table 3: Comparison of frequency for core words, f_c, found in the monolithic community and specialist terms, f_s, found in the satellite communities (Wilcoxon rank-sum test). All differences were significant at $p < 0.001$.

4 Discussion

The results of our method show that outlying structure in the lexicon can be extracted directly from large collections of unstructured text. The lexicon's topography, previously explored in dictionary studies, contains modular groups of satellite terms that are observable without appeal to external resources or document structure and with minimal normalization. The contribution of this method is two-fold: it confirms the structure of the *observed* lexicon is similar to that apparent in the organization of dictionaries (Picard et al., 2013). Second, it offers a tractable, reliable means of extracting and summarizing structure in the fringes of the lexicon.

The output of the model developed here is similar to topic models, but with some important differences. Topic models produce a probability distribution over words to define a topic, which can be summarized by the top 10 to 20 most likely words. Instead of probabilities, the within-

community hub-scores were used to rank words in each cluster. This means that the actual structure of the community (to which topics have no analogue) is responsible for producing the scores that rate words' internal relevance. Another crucial difference is that topic size from a single sampling iteration tends to correlate with coherence (Mimno et al., 2011), but in the current method, there is no correlation between cluster size and coherence ($p = 0.98$). The other important difference is that whereas topic models produce a topic-document mixture that can be used for posterior inference, to perform such inference with our method, the output would have to be used indirectly.

One understated strength of the community detection method is the minimal required preprocessing. Whereas many solutions in NLP (including topic models) require document segmentation, lexical normalization and statistical normalizations on the co-occurrence matrix itself, the only variable in our method is the co-occurrence window size. However, lemmatization (or stemming) could help collapse morpho-syntactic variation among terms in the results, but stop-word removal, sentence segmentation and TF-IDF weighting appear unnecessary. What might be most surprising given the examples in Table 2 is that word-document occurrence information is not used at all. This makes the the method particularly useful for large collections with little to no structure.

One question overlooked in our analysis concerns the effect the core has on the satellites. It could be that the proportional size of a collection's core is indicative of the degree of specialist terminology contained in the collection. Also, the raw number of satellite communities might indicate the level of diversity in a corpus. Addressing these questions could yield measures of previously vague and latent variables like specialty or topical diversity, without employing a direct semantic analysis. By measuring a collection's core size, relative to its satellites, one could also use measure changes in specialization. The Infomap algorithm could accommodate such an experiment: by varying the threshold of density that constitutes a community, the core could be made smaller, yielding more satellites, the coherence of which could be compared to those reported here. One could examine the position of individual words in the satellite(s) to explore what features signal important, emerging and dying terms or to track diachronic movement of terms like *computer* or *gene* from the specialized periphery to core of the lexicon.

At the level of inter-related term clusters, there are likely important or central groups that influence other satellites. There is no agreed upon measure of "community centrality" in a network sense (Eaton and Mansbach, 2012). One way to measure the importance of a community would be to use significance testing on the internal link mass compared to the external (Csardi and Nepusz, 2006). However, this approach discards some factors for which one might want to account, such as centrality in the network of communities and their composition. Future work could seek to combine graph-theoretic notions of centrality and intuitions about the defining features of term clusters. Another avenue for future research would be to use mixed membership community detection (Gopalan and Blei, 2013). Allowing terms to be represented in more than one community would accommodate words like *nuclear*, that might be found relating to weaponry, energy production and physics research at the same time. Using co-occurrence networks to extract clusters of specialist terms, though an important task, is perhaps only a starting point for exploring the observed lexicon. Network-based analysis of language offers a general and powerful potential to address a range of questions about the lexicon, other NLP tasks and language more generally.

Acknowledgments

Thanks to E. Duede and three reviewers for comments on earlier versions of this manuscript. This work was supported by a grant from the Templeton Foundation to the Metaknowledge Research Network and grant #1158803 from the National Science Foundation.

References

Andrea Baronchelli, Ramon Ferrer i Cancho, Romualdo Pastor-Satorras, Nick Chater, and Morten H. Christiansen. 2013. Networks in cognitive science. *Trends in cognitive sciences*, 17(7):348–360.

David M. Blei. 2012. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of the German Society for Computational Linguistics & Language Technology*, pages 31–40.

Hiram Calvo, Alexander Gelbukh, and Adam Kilgarriff. 2005. Distributional thesaurus versus wordnet: A comparison of backoff techniques for unsupervised pp attachment. In *Computational Linguistics and Intelligent Text Processing*, pages 177–188. Springer.

Jonathan Chang, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Neural Information Processing Systems*, volume 22, pages 288–296.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Gabor Csardi and Tamas Nepusz. 2006. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.

Ido Dagan, Lillian Lee, and Fernando C.N. Pereira. 1999. Similarity-based models of word cooccurrence probabilities. *Machine Learning*, 34(1-3):43–69.

Eric Eaton and Rachael Mansbach. 2012. A spin-glass model for semi-supervised community detection. In *Association for the Advancement of Artificial Intelligence*.

Ramon Ferror i Cancho and Richard V. Solé. 2001. The small world of human language. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268(1482):2261–2265.

Dipl-Math Bettina Friedl, Julia Heidemann, et al. 2010. A critical review of centrality measures in social networks. *Business & Information Systems Engineering*, 2(6):371–385.

Prem K. Gopalan and David M. Blei. 2013. Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences*, 110(36):14534–14539.

Donald Hindle. 1990. Noun classification from predicate-argument structures. In *Proceedings of the 28th annual meeting on Association for Computational Linguistics*, pages 268–275.

Yuening Hu, Jordan Boyd-Graber, and Brianna Satinoff. 2011. Interactive topic modeling. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 248–257.

Junko Itô and Armin Mester. 1995. The core-periphery structure of the lexicon and constraints on reranking. *University of Massachusetts occasional papers in linguistics*, 18:181–209.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

A. Blondin Massé, Guillaume Chicoisne, Yassine Gargouri, Stevan Harnad, Olivier Picard, and Odile Marcotte. 2008. How is meaning grounded in dictionary definitions? In *Proceedings of the 3rd Textgraphs Workshop on Graph-Based Algorithms for Natural Language Processing*, pages 17–24.

Rada Mihalcea and Dragomir Radev. 2011. *Graph-based natural language processing and information retrieval*. Cambridge University Press.

George A. Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

David Mimno, Hanna M. Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 262–272.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 100–108.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity: measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*, pages 38–41.

Olivier Picard, Mélanie Lord, Alexandre Blondin-Massé, Odile Marcotte, Marcos Lopes, and Stevan Harnad. 2013. Hidden structure and function in the lexicon. *NLPCS 2013: 10th International Workshop on Natural Language Processing and Cognitive Science, Marseille, France.*

Philip Resnik. 1997. Selectional preference and sense disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How*, pages 52–57.

Martin Rosvall and Carl T. Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123.

Martin Rosvall and Carl T. Bergstrom. 2011. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PloS one*, 6(4):e18209.

Ekaterina Shutova, Simone Teufel, and Anna Korhonen. 2013. Statistical metaphor processing. *Computational Linguistics*, 39(2):301–353.

John Sinclair. 1996. The empty lexicon. *International Journal of Corpus Linguistics*, 1(1):99–119.

Mark Steyvers and Tom Griffiths. 2007. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440.

Lin Sun and Anna Korhonen. 2009. Improving verb clustering with automatically acquired selectional preferences. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 638–647.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

Xing Wei and Bruce Croft. 2006. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185.

Yorick Wilks. 1975. A preferential, pattern-seeking, semantics for natural language inference. *Artificial Intelligence*, 6(1):53–74.

Citation-Enhanced Keyphrase Extraction from Research Papers: A Supervised Approach

Cornelia Caragea[1], **Florin Bulgarov**[1], **Andreea Godea**[1], **Sujatha Das Gollapalli**[2]
[1]Computer Science and Engineering, University of North Texas, TX, USA
[2]Institute for Infocomm Research, A*STAR, Singapore
ccaragea@unt.edu, FlorinBulgarov@my.unt.edu,
AndreeaGodea@my.unt.edu, gsdas@cse.psu.edu

Abstract

Given the large amounts of online textual documents available these days, e.g., news articles, weblogs, and scientific papers, *effective* methods for extracting keyphrases, which provide a high-level topic description of a document, are greatly needed. In this paper, we propose a supervised model for keyphrase extraction from research papers, which are embedded in citation networks. To this end, we design novel features based on citation network information and use them in conjunction with traditional features for keyphrase extraction to obtain remarkable improvements in performance over strong baselines.

1 Introduction

Keyphrase extraction is the problem of automatically extracting important phrases or concepts (i.e., the *essence*) of a document. Keyphrases provide a high-level topic description of a document and are shown to be rich sources of information for many applications such as document classification, clustering, recommendation, indexing, searching, and summarization (Jones and Staveley, 1999; Zha, 2002; Hammouda et al., 2005; Pudota et al., 2010; Turney, 2003). Despite the fact that keyphrase extraction has been widely researched in the natural language processing community, its performance is still far from being satisfactory (Hasan and Ng, 2014).

Many previous approaches to keyphrase extraction generally used only the textual content of a target document to extract keyphrases (Hulth, 2003; Mihalcea and Tarau, 2004; Liu et al., 2010). Recently, Wan and Xiao (2008) proposed a model that incorporates a local neighborhood of a document. However, their neighborhood is limited to textually-similar documents, where the cosine similarity between the *tf-idf* vectors of documents is used to compute their similarity. We posit that, in addition to a document's textual content and textually-similar neighbors, other informative neighborhoods exist that have the potential to improve keyphrase extraction. For example, in a scholarly domain, research papers are not isolated. Rather, they are highly inter-connected in giant *citation networks*, in which papers *cite* or *are cited* by other papers. In a citation network, information flows from one paper to another via the citation relation (Shi et al., 2010). This information flow and the influence of one paper on another are specifically captured by means of *citation contexts*, i.e., short text segments surrounding a citation's mention. These contexts are not arbitrary, but they serve as brief summaries of a cited paper. Figure 1 illustrates this idea using a small citation network of a paper by Rendle et al. (2010) that cites (Zimdars et al., 2001), (Hu et al., 2008), (Pan and Scholz, 2009) and (Shani et al., 2005) and is cited by (Cheng et al., 2013). The citation mentions and citation contexts are shown with a dashed line. Note the high overlap between the words in contexts and those in the title and abstract (shown in bold) and the author-annotated keywords.

One question that can be raised is the following: *Can we effectively exploit information available in large inter-linked document networks in order to improve the performance of keyphrase extraction?* The research that we describe in this paper addresses specifically this question using *citation networks of research papers* as a case study. Extracting keyphrases that can accurately "represent" research papers is crucial to dealing with the large numbers of research papers published during these "big data" times. The importance of keyphrase extraction from research papers is also emphasized by the recent SemEval 2010 Shared Task on this topic (Kim et al., 2010; Kim et al., 2013).

Our contributions. We present a supervised

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1435–1446,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

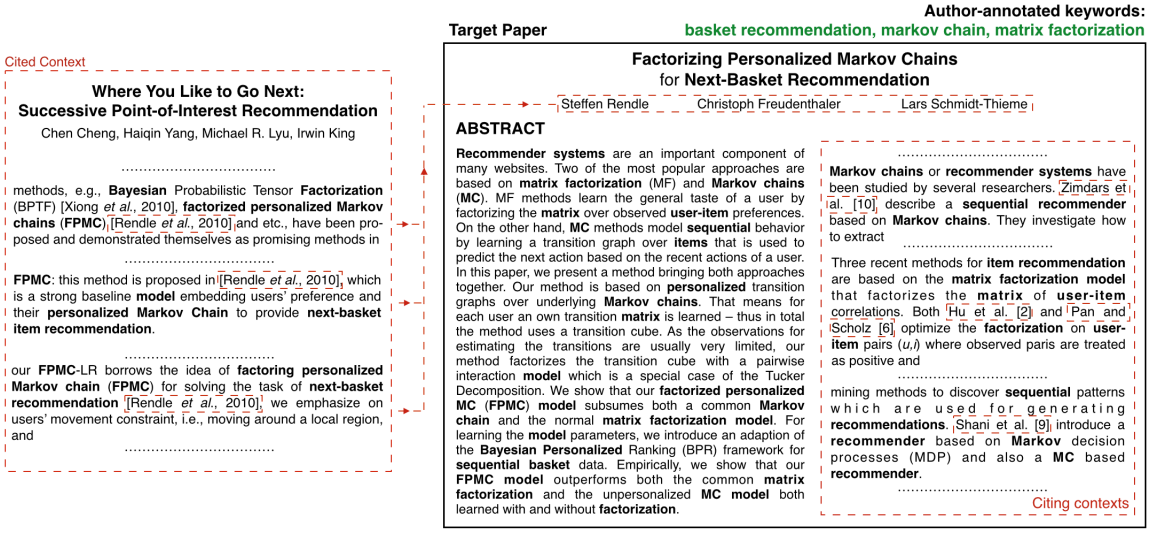


Figure 1: A small citation network corresponding to a paper by Rendle et al. (2010).

approach to keyphrase extraction from research papers that, in addition to the information contained in a paper itself, effectively incorporates, in the learned models, information from the paper's local neighborhood available in citation networks. To this end, we design novel features for keyphrase extraction based on citation context information and use them in conjunction with traditional features in a supervised probabilistic framework. We show empirically that the proposed models substantially outperform strong baselines on two datasets of research papers compiled from two machine learning conferences: the World Wide Web and Knowledge Discovery from Data.

The rest of the paper is organized as follows: We summarize closely related work in Section 2. The supervised classification for keyphrase extraction is discussed in Section 3. Experiments and results are presented in Section 4, followed by conclusions and future directions of our work.

2 Related Work

Many approaches to keyphrase extraction have been proposed in the literature along two lines of research: supervised and unsupervised, using different types of documents including scientific abstracts, newswire documents, meeting transcripts, and webpages (Frank et al., 1999; Hulth, 2003; Nguyen and Kan, 2007; Liu et al., 2009; Marujo et al., 2013; Mihalcea and Tarau, 2004).

In the supervised line of research, keyphrase extraction is formulated as a binary classification problem, where candidate phrases are classified as either positive (i.e., keyphrases) or negative (i.e., non-keyphrases) (Frank et al., 1999; Turney, 2000; Hulth, 2003). Different feature sets and classification algorithms gave rise to different models. For example, Hulth (2003) used four different features in conjunction with a *bagging* technique. These features are: term frequency, collection frequency, the relative position of the first occurrence and the part-of-speech tag of a term. Frank et al. (1999) developed a system called KEA that used only two features: *tf-idf* (term frequency-inverse document frequency) of a phrase and the *distance* of a phrase from the beginning of a document (i.e., its relative position) and used them as input to Naïve Bayes. Nguyen and Kan (2007) extended KEA to include features such as the distribution of keyphrases among different sections of a research paper, and the acronym status of a term. In contrast to these works, we propose novel features extracted from the local neighborhoods of documents available in interlinked document networks. Medelyan et al. (2009) extended KEA as well to integrate information from Wikipedia. In contrast, we used only information intrinsic to our data. Enhancing our models with Wikipedia information would be an interesting future direction to pursue.

In the unsupervised line of research, keyphrase extraction is formulated as a ranking problem, where keyphrases are ranked using their *tf* (Barker and Cornacchia, 2000), *tf-idf* (Zhang et al., 2007; Lee and Kim, 2008; Liu et al., 2009; Tonella et al., 2003), and *term informativeness* (Wu and Giles, 2013; Rennie and Jaakkola, 2005; Kireyev, 2009) (among others). The ranking based on *tf-idf* has

been shown to work well in practice (Liu et al., 2009; Hasan and Ng, 2010) despite its simplicity. Frantzi et al. (1998) combined linguistics and statistical information to extract technical terms from documents in digital libraries. Graph-based algorithms and centrality measures are also widely used in unsupervised models. A word graph is built for each document such that nodes correspond to words and edges correspond to word association patterns. Nodes are then ranked using graph centrality measures such as PageRank and its variants (Mihalcea and Tarau, 2004; Wan and Xiao, 2008; Liu et al., 2010; Zhao et al., 2011), HITS scores (Litvak and Last, 2008), as well as node degree and betweenness (Boudin, 2013; Xie, 2005). Wan and Xiao (2008) were the first to consider modeling a local neighborhood of a target document in addition to the document itself, and applied this approach to news articles on the Web. Their local neighborhood consists of textually similar documents, and did not capture information contained in document networks.

Using terms from citation contexts of scientific papers is not a new idea. It was used before in various applications. For example, Ritchie et al. (2006) used a combination of terms from citation contexts and existing index terms of a paper to improve indexing of cited papers. Citation contexts were also used to improve the performance of citation recommendation systems (Kataria et al., 2010; He et al., 2010) and to study author influence (Kataria et al., 2011). This idea of using terms from citation contexts resembles the analysis of hyperlinks and the graph structure of the Web, which are instrumental in Web search (Manning et al., 2008). Many current Web search engines build on the intuition that the anchor text pointing to a page is a good descriptor of its content, and thus use anchor text terms as additional index terms for a target webpage. The use of links and anchor text was thoroughly researched for IR tasks (Koolen and Kamps, 2010), broadening a user's search (Chakrabarti et al., 1998), query refinement (Kraft and Zien, 2004), and enriching document representations (Metzler et al., 2009).

Moreover, citation contexts were used for scientific paper summarization (Abu-Jbara and Radev, 2011; Qazvinian et al., 2010; Qazvinian and Radev, 2008; Mei and Zhai, 2008; Lehnert et al., 1990; Nakov et al., 2004). Among these, probably the most similar to our work is the work by Qazvinian et al. (2010), where a set of important keyphrases is extracted first from the citation contexts in which the paper to be summarized is cited by other papers and then the "best" subset of sentences that contain such keyphrases is returned as the summary. However, keyphrases in (Qazvinian et al., 2010) are extracted using frequent n-grams in a language model framework, whereas in our work, we propose a supervised approach to a different task: keyphrase extraction. Mei and Zhai (2008) used information from citation contexts to determine what sentences of a paper are of high impact (as measured by the influence of a target paper on further studies of similar or related topics). These sentences constitute the impact-based summary of the paper.

Despite the use of citation contexts and anchor text in many IR and NLP tasks, to our knowledge, we are the first to propose the incorporation of information available in citation networks for keyphrase extraction. In our recent work (Gollapalli and Caragea, 2014), we designed a fully unsupervised graph-based algorithm that incorporates evidence from multiple sources (citation contexts as well as document content) in a flexible manner to score keywords. In the current work, we present a supervised approach to keyphrase extraction from research papers that are embedded in large citation networks, and propose novel features that show improvement over strong supervised and unsupervised baselines. To our knowledge, features extracted from citation contexts have not been used before for keyphrase extraction in a supervised learning framework.

3 Problem Characterization

In citation networks, in addition to the information contained in a paper itself, *citing* and *cited* papers capture different aspects (e.g., topicality, domain of study, algorithms used) about the target paper (Teufel et al., 2006), with *citation contexts* playing an instrumental role. A citation context is defined as a window of n words surrounding a citation mention. We conjecture that citation contexts, which act as brief summaries about a cited paper, provide additional clues in extracting keyphrases for a target paper. These clues give rise to the unique design of our model, called citation-enhanced keyphrase extraction (CeKE).

3.1 Citation-enhanced Keyphrase Extraction

Our proposed citation-enhanced keyphrase extraction (CeKE) model is a supervised binary classifi-

Feature Name	Description
Existing features for keyphrase extraction	
tf-idf	term frequency * inverse document frequency, computed from a target paper; used in KEA
relativePos	the position of the first occurrence of a phrase divided by the total number of tokens; used in KEA and Hulth's methods
POS	the part-of-speech tag of the phrase; used in Hulth's methods
Novel features - Citation Network Based	
inCited	if the phrase occurs in cited contexts
inCiting	if the phrase occurs in citing contexts
citation tf-idf	the *tf-idf* value of the phrase, computed from the aggregated citation contexts
Novel features - Extensions of Existing Features	
first position	the distance of the first occurrence of a phrase from the beginning of a paper
tf-idf-Over	*tf-idf* larger than a threshold θ
firstPosUnder	the distance of the first occurrence of a phrase from the beginning of a paper is below some value β

Table 1: The list of features used in our model.

cation model, built on a combination of *novel* features that capture information from citation contexts and existing features from previous works. The features are described in §3.1.1. CeKE classifies candidate phrases as keyphrases (i.e., positive) or non-keyphrases (i.e., negative) using Naïve Bayes classifiers. Positive examples for training correspond to manually annotated keyphrases from the training research papers, whereas negative examples correspond to the remaining candidate phrases from these papers. The generation of candidate phrases is explained in §3.2.

Note that Naïve Bayes classifies a phrase as a keyphrase if the probability of the phrase belonging to the positive class is greater than 0.5. However, the default threshold of 0.5 can be varied to allow only high-confidence (e.g., 0.9 confidence) phrases to be classified as keyphrases.

3.1.1 Features

We consider the following features in our model, which are shown in Table 1. They are divided into three categories: (1) *Existing features for keyphrase extraction* include: *tf-idf*, i.e., the term frequency - inverse document frequency of a candidate phrase, computed for each target paper; This feature was used in KEA (Frank et al., 1999); *relative position*, i.e., the position of the first occurrence of a phrase normalized by the length (in the number of tokens) of the target paper; *POS*, i.e., a phrase's part-of-speech tag. If a phrase is composed by more than one term, then the POS will contain the tags of all terms. The relative position was used in both KEA and Hulth (2003), and POS was used in Hulth; (2) *Novel features - Citation Network Based* include: *inCited* and *inCiting*, i.e., boolean features that are true if the candidate phrase occurs in cited and citing contexts, respectively. We differentiate between cited and citing contexts for a paper: let d be a target paper and $\mathcal{C}$ be a citation network such that $d \in \mathcal{C}$. A cited context for d is a context in which d is cited by some paper d_i in $\mathcal{C}$. A citing context for d is a context in which d is citing some paper d_j in $\mathcal{C}$. If a paper is cited in multiple contexts by another paper, the contexts are aggregated into a single one; *citation tf-idf*, i.e., the tf-idf score of each phrase computed from the citation contexts; (3) *Novel features - Extend Other Existing Features* include: *first position* of a candidate phrase, i.e., the distance of the first occurrence of a phrase from the beginning of a paper; this is similar to relative position except that it does not consider the length of a paper; *tf-idf-Over*, i.e., a boolean feature, which is true if the *tf-idf* of a candidate phrase is greater than a threshold θ, and *firstPosUnder*, also a boolean feature, which is true if the distance of the first occurrence of a phrase from the beginning of a target paper is below some value β. This feature is similar to the feature *is-in-title*, used previously in the literature (Litvak and Last, 2008; Jiang et al., 2009). Both *tf-idf* and *citation tf-idf* features showed better results when each *tf* was divided by the maximum *tf* values from the target paper or citation contexts.

The *tf-idf* features have high values for phrases that are frequent in a paper or citation contexts, but are less frequent in collection and have low values for phrases with high collection frequency. We computed the *idf* component from each collection used in experiments. Phrases that occur in cited and citing contexts as well as early in a paper are likely to be keyphrases since: (1) they capture some aspect about the target paper and (2) authors start to describe their problem upfront.

3.2 Generating Candidate Phrases

We generate candidate phrases from the textual content of a target paper by applying parts-of-

Dataset	Num. (#) Papers	Average Cited Ctx.	Average Citing Ctx.	Average Keyphrases	#uni-grams	#bi-grams	#tri-grams
WWW	425	15.45	18.78	4.87	680	1036	247
KDD	365	12.69	19.74	4.03	363	853	189

Table 2: A summary of our datasets.

speech filters. Consistent with previous works (Hulth, 2003; Mihalcea and Tarau, 2004; Liu et al., 2010; Wan and Xiao, 2008), only nouns and adjectives are retained to form candidate phrases. The generation process consists of two steps. First, using the NLP Stanford part of speech tagger, we preprocess each document and keep only the nouns and adjectives corresponding to $\{NN, NNS, NNP, NNPS, JJ\}$. We apply the Porter stemmer on every word. The position of each word is kept consistent with the initial state of the document before any word removal is made.

Second, words extracted in the first step that have contiguous positions in a document are concatenated into n-grams. We used unigrams, bigrams, and trigrams ($n = 1, 2, 3$) as candidate phrases for classification. Similar to Wan and Xiao (2008), we eliminated phrases that end with an adjective and the unigrams that are adjectives.

4 Experiments and Results

In this section, we first describe our datasets and then present experimental design and results.

4.1 Datasets

In order to test the performance of our proposed approach, we built our own datasets since *citation-enhanced* evaluation benchmarks are not available for keyphrase extraction tasks. In particular, we compiled two datasets consisting of research papers from two top-tier machine learning conferences: World Wide Web (WWW) and Knowledge Discovery and Data Mining (KDD). Our choice for WWW and KDD was motivated by the availability of *author-input* keywords for each paper, which we used as gold-standard for evaluation.

Using the CiteSeer^x digital library[1], we retrieved the papers published in WWW and KDD (available in CiteSeer^x), and their citation network information, i.e., their cited and citing contexts. Since our goal is to study the impact of citation network information on extracting keyphrases, a paper was considered for analysis if it had at least one cited and one citing context. For each paper, we used: the title and abstract (referred to as the target paper) and its citation contexts. The reason for not considering the entire text of a paper is that scientific papers contain details, e.g., discussion of results, experimental design, notation, that do not provide additional benefits for extracting keyphrases. Hence, similar to (Hulth, 2003; Mihalcea and Tarau, 2004; Liu et al., 2009), we did not use the entire text of a paper. However, extracting keyphrases from sections such as "introduction" or "conclusion" needs further attention.

From the pdf of each paper, we extracted the author-input keyphrases. An analysis of these keyphrases revealed that generally authors describe their work using, almost half of the time, bigrams, followed by unigrams and only rarely using trigrams (or higher n-grams). A summary of our datasets that contains the number of papers, the average number of cited and citing contexts per paper, the average number of keyphrases per paper, and the number of unigrams, bigrams and trigrams, in each collection, is shown in Table 2.

Consistent with previous works (Frank et al., 1999; Hulth, 2003), the positive and negative examples in our datasets correspond to candidate phrases that consist of up to three tokens. The positive examples are candidate phrases that have a match in the author-input keyphrases, whereas negative examples correspond to the remaining candidate phrases.

Context lengths. In CiteSeer^x, citation contexts have about 50 words on each side of a citation mention. A previous study by Ritchie et al. (2008) shows that a fixed window length of about 100 words around a citation mention is generally effective for information retrieval tasks. For this reason, we used the contexts provided by CiteSeer^x directly. However, in future, it would be interesting to incorporate in our models more sophisticated approaches to identifying the text that is relevant to a target citation (Abu-Jbara and Radev, 2012; Teufel, 1999) and study the influence of context lengths on the quality of extracted keyphrase.

[1] http://citeseerx.ist.psu.edu/

	WWW			KDD		
Method	Precision	Recall	F1-score	Precision	Recall	F1-score
Citation - Enhanced (CeKE)	**0.227**	**0.386**	**0.284**	**0.213**	**0.413**	**0.280**
Hulth - n-gram with tags	0.165	0.107	0.129	0.206	0.151	0.172
KEA	0.210	0.146	0.168	0.178	0.124	0.145

Table 3: The comparison of CeKE with supervised approaches on WWW and KDD collections.

4.2 Experimental Design

Our experiments are designed around the following research questions:

1. *How does the performance of citation-enhanced keyphrase extraction (CeKE) compare with the performance of existing supervised models that use only information intrinsic to the data and what are the most informative features for classification?* We compared CeKE's performance with that of classifiers trained on KEA features only and Hulth's features only and present a ranking of features based on information gain.
2. *How do supervised models that integrate citation network information compare with recent unsupervised models?* Since recent unsupervised approaches are becoming competitive with supervised approaches (Hasan and Ng, 2014), we also compared CeKE with unsupervised ranking of candidate phrases by TF-IDF, TextRank (Mihalcea and Tarau, 2004) and ExpandRank (Wan and Xiao, 2008). For unsupervised, we considered top 5 and top 10 ranked phrases when computing "@5" and "@10" measures.
3. *How well does our proposed model perform in the absence of either cited or citing contexts?* Since newly published scientific papers are not cited by many other papers, e.g., due to their recency, no cited contexts are available. We studied the quality of predicted keyphrases when either cited or citing contexts are missing. For this, we compared the performance of models trained using both cited and citing contexts with that of models that use either cited or citing contexts.

Evaluation metrics. To evaluate the performance of CeKE, we used the following metrics: precision, recall and F1-score for the positive class since correct identification of keyphrases is of most interest. These metrics were widely used in previous works (Hulth, 2003; Mihalcea and Tarau, 2004; Wan and Xiao, 2008; Hasan and Ng, 2010). The reported values are averaged in 10-fold cross-validation experiments, where folds were created at document level and candidate phrases were extracted from the documents in each fold to form the training and test sets. In all experiments, we used Naïve Bayes and their Weka implementation[2]. However, any probabilistic classifier that returns a posterior probability of the class given an example, can be used with our features.

The θ parameter was set to the (title and abstract) *tf-idf averaged* over the entire collection, while β was set to 20. These values were estimated on a validation set sampled from training.

4.3 Results and Discussion

The impact of citation network information on the keyphrase extraction task. Table 3 shows the results of the comparison of CeKE with two supervised approaches, KEA and Hulth's approach. The features used in KEA are the *tf-idf* and the *relative position* of a candidate phrase, whereas those used in Hulth's approach are *tf*, *cf* (i.e., collection frequency), *relative position* and *POS tags*. CeKE is trained using all features from Table 1. Among the three methods for candidate phrase formation proposed in Hulth (2003), i.e., n-grams, NP-chunks, and POS Tag Patterns, our Hulth's implementation is based on n-grams since this gives the best results among all methods (see (Hulth, 2003) for more details). In addition, the n-grams method is the most similar to our candidate phrase generation and that used in Frank et al. (1999).

As can be seen from Table 3, CeKE outperforms KEA and Hulth's approach in terms of all performance measures on both WWW and KDD, with a substantial improvement in recall over both approaches. For example, on WWW, CeKE achieves a recall of 0.386 compared to 0.146 and 0.107 recall achieved by KEA and Hulth's, respectively.

[2]http://www.cs.waikato.ac.nz/ml/weka/

	WWW			KDD		
Method	Precision	Recall	F1-score	Precision	Recall	F1-score
Citation - Enhanced (CeKE)	**0.227**	**0.386**	**0.284**	**0.213**	**0.413**	**0.280**
TF-IDF - Top 5	0.089	0.100	0.094	0.083	0.102	0.092
TF-IDF - Top 10	0.075	0.169	0.104	0.080	0.203	0.115
TextRank - Top 5	0.058	0.071	0.062	0.051	0.065	0.056
TextRank - Top 10	0.062	0.133	0.081	0.053	0.127	0.072
ExpandRank - 1 neigh. - Top 5	0.088	0.109	0.095	0.077	0.103	0.086
ExpandRank - 1 neigh. - Top 10	0.078	0.165	0.101	0.071	0.177	0.098
ExpandRank - 5 neigh. - Top 5	0.093	0.113	0.100	0.080	0.108	0.090
ExpandRank - 5 neigh. - Top 10	0.080	0.172	0.104	0.068	0.172	0.095
ExpandRank - 10 neigh. - Top 5	0.094	0.113	0.100	0.077	0.103	0.086
ExpandRank - 10 neigh. - Top 10	0.076	0.162	0.099	0.065	0.164	0.091

Table 5: The comparison of CeKE with unsupervised approaches on WWW and KDD collections.

Rank	*Feature*	IG Score
1	*abstract tf-idf*	0.0234
2	*first position*	0.0188
3	*citation tf-idf*	0.0177
4	*relativePos*	0.0154
5	*firstPosUnder*	0.0148
6	*inCiting*	0.0129
7	*inCited*	0.0098
8	*POS*	0.0085
9	*tf-idf-Over*	0.0078

Table 4: Feature ranking by Info Gain on WWW.

Although there are only small variations from KEA to Hulth's approach, KEA performs better on WWW, but worse on KDD compared with Hulth's approach. In contrast, CeKE shows consistent improvement over the two approaches on both datasets, hence, effectively making use of the information available in the citation network.

In order to understand the importance of our features, we ranked them based on Information Gain (IG), which determines how informative a feature is with respect to the class variable. Table 4 shows the features ranked in decreasing order of their IG scores for WWW. As can be seen from the table, *tf-idf* and *citation tf-idf* are both highly ranked, first and third, respectively, illustrating that they contain significant information in predicting keyphrases. The *first position* of a phrase is also of great impact. This is consistent with the fact that almost half of the identified keywords and about 20% of the annotated keyphrases appear in title. Similar ranking is obtained on KDD.

The comparison of CeKE with unsupervised state-of-the-art models. Table 5 shows the results of the comparison of CeKE with three unsupervised ranking approaches: TF-IDF (Tonella et al., 2003), TextRank (Mihalcea and Tarau, 2004), and ExpandRank (Wan and Xiao, 2008). TF-IDF and TextRank use information only from the target paper, whereas ExpandRank uses a small textual neighborhood in addition to the target paper. Note that, for all unsupervised methods, we used Porter stemmer and the same candidate phrase generation as in CeKE, as explained in §3.2.

For TF-IDF, we first tokenized the target paper and computed the score for each word, and then formed phrases and summed up the score of every word within a phrase. For TextRank, we built an undirected graph for each paper, where the nodes correspond to words in the target paper and edges are drawn between two words that occur next to each other in the text, i.e., the window size is 2. For ExpandRank, we built an undirected graph for each paper and its local textual neighborhood. Again, nodes correspond to words in the target paper and its textually similar papers and edges are drawn between two words that occur within a window of 10 words from each other in the text, i.e., the window size is 10. We performed experiments with 1, 5, and 10 textually-similar neighbors. For TextRank and ExpandRank, we summed up the scores of words within a phrase as in TF-IDF.

	WWW			KDD		
Method	Precision	Recall	F1-score	Precision	Recall	F1-score
CeKE - Both contexts	**0.227**	**0.386**	**0.284**	**0.213**	**0.413**	**0.280**
CeKE - Only cited contexts	0.222	0.286	0.247	0.192	0.300	0.233
CeKE - Only citing contexts	0.203	0.342	0.253	0.195	0.351	0.250

Table 6: Results of CeKE using both contexts and using with only cited or citing contexts.

For each unsupervised method, we computed results for top 5 and top 10 ranked phrases. As can be seen from Table 5, CeKE substantially outperforms all the other methods for our domain of study, i.e., papers from WWW and KDD, illustrating again that the citation network of a paper contains important information that can show remarkable benefits for keyphrase extraction. Among all unsupervised methods, ExpandRank with fewer textual similar neighbor (one or five) performs the best. This is generally consistent with the results shown in (Wan and Xiao, 2008) for news articles.

The effect of cited and citing contexts information on models' performance. Table 6 shows the precision, recall and F-score values for some variations of our method when: (1) all the citation contexts for a paper are used, (2) only cited contexts are used, (3) only citing contexts are used. The motivation behind this experiment was to determine how well the proposed model would perform on newly published research papers that have not accumulated citations yet. As shown in the table, there is no substantial difference in terms of precision between CeKE models that use only cited or only citing contexts, although the recall is substantially higher for the case when only citing contexts are used, for both WWW and KDD. The CeKE that uses both citing and cited contexts achieves a substantially higher recall and only a slightly higher precision compared with the cases when only one context type is available. The fact that the citing context information provides a slight improvement in performance over cited contexts is consistent with the intuition that when citing a paper y, an author generally summarizes the main ideas from y using important words from a target paper x, making the citing contexts to have higher overlap with words from x. In turn, a paper z that cites x may use paraphrasing to summarize ideas from x with words more similar to those from z.

Note that the results of all above experiments are statistically significant at p-values ≥ 0.05, using a paired t-test on F1-scores.

4.4 Anecdotal Evidence

In order to check the transferability of our proposed approach to other research fields, e.g., natural language processing, it would be interesting to use our trained classifiers on WWW and KDD collections and evaluate them on new collections such as NLP related collections. Since NLP collections annotated with keyphrases are not available, we show anecdotal evidence for only one paper. We selected for this task an award winning paper published in the EMNLP conference. The paper's title is "Unsupervised semantic parsing" and has won the Best Paper Award in the year 2009 (Poon and Domingos, 2009). In order for our algorithm to work, we gathered from the Web (using Google Scholar) all the cited and citing contexts that were available (49 cited contexts and 30 citing contexts). We manually annotated the target paper with keyphrases. The title, abstract and all the contexts were POS tagged using the NLP Stanford tool. We then trained a classifier on the features shown in Table 1, on both WWW and KDD datasets combined. The trained classifier was used to make predictions, which were compared against the manually annotated keyphrases. The results are shown in Figure 2, which displays the title and abstract of the paper and the predicted keyphrases. Candidate phrases that are predicted as keyphrases are marked in red bold, those predicted as non-keyphrases are shown in black, while the filtered out words are shown in light gray.

We tuned our classifier trained on WWW and KDD to return as keyphrases only those that had an extremely high probability to be keyphrases. Specifically, we used a threshold of 0.985. The probability of each returned keyphrase (which is above 0.985) is shown in the upper right corner of a keyphrase. Human annotated keyphrases are marked in italic, under the figure. There is a clear match between the predictions and the human annotations. It is also possible to extract more or less keyphrases simply by adjusting the threshold

Unsupervised Semantic Parsing$^{0.997}$

We present the first unsupervised approach to the problem of learning a **semantic parser**$^{1.000}$, using **Markov logic**$^{0.991}$. Our **USP system**$^{0.985}$ transforms dependency trees into quasi-logical forms, recursively induces lambda forms from these, and clusters them to abstract away syntactic variations of the same meaning. The MAP **semantic parse**$^{1.000}$ of a sentence is obtained by recursively assigning its parts to lambda-form clusters and composing them. We evaluate our approach by using it to extract a knowledge base from biomedical abstracts and answer questions. **USP**$^{1.000}$ substantially outperforms TextRunner, DIRT and an informed baseline on both precision and recall on this task.

Human annotated labels: *unsupervised semantic parsing, Markov logic, USP system*

Figure 2: The title and abstract of an EMNLP paper by Poon and Domingos (2009) and human annotated keyphrases for the paper. Black words represent candidate phrases. Red bold words represent predicted keyphrases. The numbers above predicted keyphrases are probabilities for the positive class assignment.

on the probability output by Naïve Bayes. For example, if we decrease the threshold to 0.920 the following phrases would be added to the returned set of keyphrases: *dependency trees, quasi-logical forms* and *unsupervised approach*.

Another interesting aspect is the frequency of occurrence of the predicted keyphrases in the cited and citing contexts. Table 7 shows the term-frequency of every predicted keyphrase within the citation network. For example, the phrase *semantic parser* appears in 29 cited contexts and 26 citing contexts. The reason for the higher cited context frequency is not necessarily due to importance, but could be due to the larger number of cited vs. citing contexts for this paper (49 vs. 30). The high rate of keyphrases within the citation network validates our assumption of the importance of citation networks for keyphrase extraction.

Finally, we performed the same experiment with Hulth's and KEA methods. While the classifier trained on Hulth's features did not identify any keyphrases, KEA managed to identify several good ones (e.g., *USP*, *semantic parser*), but left out some important ones (e.g., *Markov logic, unsupervised*). Moreover, the keyphrases predicted by KEA have a lower confidence. For this reason, lowering the probability threshold would result in selecting other bad keyphrases.

4.5 Error analysis

We performed an error analysis and found that candidate phrases are predicted as keyphrases (FPs), although they do not appear in gold standard, i.e., the set of author-input keyphrases, in cases when: 1) a more general terms is used to describe an important concept of a document, e.g., *co-authorship prediction* represented as *link prediction* or *Twitter platform* represented as *social media*; 2) an important concept is omitted (either intentionally or forgetfully) from the set of author-input keyphrases.

Keyphrase	#cited c.	#citing c.
semantic parser	29	26
USP	31	10
Markov logic	15	10
unsupervised semantic parsing	12	1
USP system	3	2

Table 7: Frequency of the predicted keyphrases in cited / citing contexts.

Hence, while we believe that authors are the best keyphrase annotators for their own work, there are cases when important keyphrases are overlooked or expressed in different ways, possibly due to the human subjective nature in choosing important keyphrases that describe a document. To this end, a limitation of our model is the use of a single gold standard keyphrase annotation. In future, we plan to acquire several human keyphrase annotation sets for our datasets and test the performance of the proposed approach on these annotation sets, independently and in combination.

Keyphrases that appear in gold standard are predicted as non-keyphrases (FNs) when: 1) a keyphrase is infrequent in abstract; 2) its distance from the beginning of a document is large; 3) does not occur or occurs only rarely in a document's citation contexts, either citing or cited contexts. Examples of FNs are model/algorithm/approach names, e.g., *random walks*, that appear in sentences such as: "In this paper, we model the problem [· · ·] by using *random walks*." Although such

a sentence may appear further away from the beginning of an abstract, it contains significant information from the point of view of keyphrase extraction. The design of patters such as $< by\ using\ \$model >$ or $< uses\ \$model >$ could lead to improved classification performance.

Further investigation of FPs and FNs will be considered in future work. We believe that a better understanding of errors has the potential to advance *state-of-the-art* for keyphrase extraction.

5 Conclusion and Future Directions

In this paper, we presented a supervised classification model for keyphrase extraction from scientific research papers that are embedded in citation networks. More precisely, we designed novel features that take into account citation network information for building supervised models for the classification of candidate phrases as keyphrases or non-keyphrases. The results of our experiments show that the proposed supervised model trained on a combination of citation-based features and existing features for keyphrase extraction performs substantially better compared with state-of-the-art supervised and unsupervised models.

Although we illustrated the benefits of leveraging inter-linked document networks for keyphrase extraction from scientific documents, the proposed model can be extended to other types of documents such as webpages, emails, and weblogs. For example, the anchor text on hyperlinks in weblogs can be seen as the "citation context".

Another aspect of future work would be the use of external sources to better identify candidate phrases. For example, the use of Wikipedia was studied before to check if the concept behind a phrase has its own Wikipedia page (Medelyan et al., 2009). Furthermore, since citations occur in all sciences, extensions of the proposed model to other domains, e.g., Biology and Chemistry, and other applications, e.g., document summarization, similar to Mihalcea and Tarau (2004) and Qazvinian et al. (2010), are of particular interest.

Acknowledgments

We are grateful to Dr. C. Lee Giles for the CiteSeerX data, which allowed the generation of citation graphs. We also thank Kishore Neppalli and Juan Fernández-Ramízer for their help with various dataset construction tasks. We very much appreciate the constructive feedback from our anonymous reviewers. This research was supported in part by NSF awards #1353418 and #1423337 to Cornelia Caragea. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of NSF.

References

Amjad Abu-Jbara and Dragomir Radev. 2011. Coherent citation-based summarization of scientific papers. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, HLT '11, pages 500–509.

Amjad Abu-Jbara and Dragomir Radev. 2012. Reference scope identification in citing sentences. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 80–90.

Ken Barker and Nadia Cornacchia. 2000. Using Noun Phrase Heads to Extract Document Keyphrases. In *Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, AI '00, pages 40–52, London, UK, UK. Springer-Verlag.

Florian Boudin. 2013. A comparison of centrality measures for graph-based keyphrase extraction. In *Proc. of IJCNLP*, pages 834–838, Nagoya, Japan.

Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, and Jon Kleinberg. 1998. Automatic resource compilation by analyzing hyperlink structure and associated text. *Comput. Netw. ISDN Syst.*, 30(1-7):65–74, April.

Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where you like to go next: Successive point-of-interest recommendation. In *Proc. of IJCAI'13*, pages 2605–2611, Beijing, China.

Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 668–673, Stockholm, Sweden.

Katerina T. Frantzi, Sophia Ananiadou, and Jun-ichi Tsujii. 1998. The c-value/nc-value method of automatic recognition for multi-word terms. In *Proc. of ECDL '98*, pages 585–604.

Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14)*, Québec City, Québec, Canada.

Khaled M. Hammouda, Diego N. Matute, and Mohamed S. Kamel. 2005. Corephrase: Keyphrase extraction for document clustering. In *Proc. of the 4th*

International Conference on Machine Learning and Data Mining in Pattern Recognition, MLDM'05, pages 265–274, Leipzig, Germany.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in Unsupervised Keyphrase Extraction: Making Sense of the State-of-the-Art. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 365–373.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Qi He, Jian Pei, Daniel Kifer, Prasenjit Mitra, and Lee Giles. 2010. Context-aware citation recommendation. In *Proc. of WWW '10*, pages 421–430, Raleigh, North Carolina, USA.

Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proc. of the 8th IEEE Intl. Conference on Data Mining*, ICDM '08, pages 263–272.

Anette Hulth. 2003. Improved Automatic Keyword Extraction Given More Linguistic Knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, EMNLP '03, pages 216–223.

Xin Jiang, Yunhua Hu, and Hang Li. 2009. A Ranking Approach to Keyphrase Extraction. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 756–757. ACM.

Steve Jones and Mark S. Staveley. 1999. Phrasier: A system for interactive document retrieval using keyphrases. In *Proceedings of SIGIR '99*, pages 160–167, Berkeley, California, USA.

Saurabh Kataria, Prasenjit Mitra, and Sumit Bhatia. 2010. Utilizing context in generative bayesian models for linked corpus. In *In Proc. of AAAI '10*, pages 1340–1345, Atlanta, Georgia, USA.

Saurabh Kataria, Prasenjit Mitra, Cornelia Caragea, and C. Lee Giles. 2011. Context sensitive topic models for author influence in document networks. In *Proceedings of IJCAI'11*, pages 2274–2280, Barcelona, Catalonia, Spain.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. SemEval-2010 Task 5: Automatic Keyphrase Extraction from Scientific Articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 21–26, Los Angeles, California.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2013. Automatic keyphrase extraction from scientific articles. *Language Resources and Evaluation, Springer*, 47(3):723–742.

Kirill Kireyev. 2009. Semantic-based estimation of term informativeness. In *Proc. of NAACL '09*, pages 530–538, Boulder, Colorado.

Marijn Koolen and Jaap Kamps. 2010. The importance of anchor text for ad hoc search revisited. In *Proceedings of SIGIR '10*, pages 122–129, Geneva, Switzerland.

Reiner Kraft and Jason Zien. 2004. Mining anchor text for query refinement. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 666–674, New York, NY, USA. ACM.

Sungjick Lee and Han-joon Kim. 2008. News Keyword Extraction for Topic Tracking. In *Proceedings of the 2008 Fourth International Conference on Networked Computing and Advanced Information Management - Volume 02*, NCM '08, pages 554–559, Washington, DC, USA. IEEE Computer Society.

Wendy Lehnert, Claire Cardie, and Ellen Rilofl. 1990. Analyzing research papers using citation sentences. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, pages 511–518.

Marina Litvak and Mark Last. 2008. Graph-Based Keyword Extraction for Single-Document Summarization. In *Proceedings of the Workshop on Multi-source Multilingual Information Extraction and Summarization*, MMIES '08, pages 17–24, Manchester, United Kingdom.

Feifan Liu, Deana Pennell, Fei Liu, and Yang Liu. 2009. Unsupervised Approaches for Automatic Keyword Extraction Using Meeting Transcripts. In *Proceedings of NAACL '09*, pages 620–628, Boulder, Colorado.

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic Keyphrase Extraction via Topic Decomposition. In *Proceedings of EMNLP '10*, pages 366–376, Cambridge, Massachusetts.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Luís Marujo, Ricardo Ribeiro, David Martins de Matos, João Paulo Neto, Anatole Gershman, and Jaime G. Carbonell. 2013. Key phrase extraction of lightly filtered broadcast news. *CoRR*.

Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1318–1327, Singapore.

Qiaozhu Mei and ChengXiang Zhai. 2008. Generating impact-based summaries for scientific literature. In *Proceedings of ACL-08: HLT*, pages 816–824, Columbus, Ohio.

Donald Metzler, Jasmine Novak, Hang Cui, and Srihari Reddy. 2009. Building enriched document representations using aggregated anchor text. In *Proc. of SIGIR '09*, pages 219–226, Boston, MA, USA.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of EMNLP 2004*, pages 404–411, Barcelona, Spain.

Preslav I. Nakov, Ariel S. Schwartz, and Marti A. Hearst. 2004. Citances: Citation sentences for semantic analysis of bioscience text. In *SIGIR Workshop on Search and Discovery in Bioinformatics*.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase Extraction in Scientific Publications. In *Proc. of the Intl. Conf. on Asian digital libraries*, ICADL'07, pages 317–326, Hanoi, Vietnam.

Rong Pan and Martin Scholz. 2009. Mind the gaps: Weighting the unknown in large-scale one-class collaborative filtering. In *Proceedings of KDD '09*, pages 667–676, Paris, France.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proc. of the 2009 Conference on Empirical Methods in Natural Language Processing*, EMNLP '09, pages 1–10, Singapore.

Nirmala Pudota, Antonina Dattolo, Andrea Baruzzo, Felice Ferrara, and Carlo Tasso. 2010. Automatic keyphrase extraction and ontology mining for content-based tag recommendation. *International Journal of Intelligent Systems*.

Vahed Qazvinian and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *Proc. of the 22nd Intl. Conference on Computational Linguistics*, COLING '08, pages 689–696, Manchester, United Kingdom.

Vahed Qazvinian, Dragomir R. Radev, and Arzucan Özgür. 2010. Citation summarization through keyphrase extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 895–903.

Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW '10*, pages 811–820, Raleigh, North Carolina.

Jason D. M. Rennie and Tommi Jaakkola. 2005. Using Term Informativeness for Named Entity Detection. In *Proc. of SIGIR '05*, pages 353–360.

Anna Ritchie, Simone Teufel, and Stephen Robertson. 2006. How to find better index terms through citations. In *Proc. of the Workshop on How Can Computational Linguistics Improve Information Retrieval?*, CLIIR '06, pages 25–32, Sydney, Australia.

Anna Ritchie, Stephen Robertson, and Simone Teufel. 2008. Comparing citation contexts for information retrieval. In *Proc. of CIKM '08*, pages 213–222, Napa Valley, California, USA.

Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An mdp-based recommender system. *J. Mach. Learn. Res.*, 6:1265–1295, December.

Xiaolin Shi, Jure Leskovec, and Daniel A. McFarland. 2010. Citing for high impact. In *Proceedings of the 10th Annual Joint Conference on Digital Libraries*, JCDL '10, pages 49–58, Gold Coast, Queensland, Australia.

S. Teufel, A. Siddharthan, and D. Tidhar. 2006. Automatic classification of citation function. In *Proceedings of EMNLP-06*.

S. Teufel. 1999. *Argumentative Zoning: Information Extraction from Scientific Text*. Ph.D. thesis, University of Edinburgh.

Paolo Tonella, Filippo Ricca, Emanuele Pianta, and Christian Girardi. 2003. Using Keyword Extraction for Web Site Clustering. In *Web Site Evolution, 2003. Theme: Architecture. Proceedings. Fifth IEEE International Workshop on*, pages 41–48.

Peter D. Turney. 2000. Learning algorithms for keyphrase extraction. *Inf. Retr.*, 2.

Peter D. Turney. 2003. Coherent Keyphrase Extraction via Web Mining. In *Proceedings of the 18th international joint conference on Artificial intelligence*, IJCAI'03, pages 434–439, Acapulco, Mexico.

Xiaojun Wan and Jianguo Xiao. 2008. Single Document Keyphrase Extraction Using Neighborhood Knowledge. In *Proceedings of AAAI '08*, pages 855–860, Chicago, Illinois.

Zhaohui Wu and Lee C. Giles. 2013. Measuring term informativeness in context. In *Proceedings of NAACL '13*, pages 259–269, Atlanta, Georgia.

Zhuli Xie. 2005. Centrality Measures in Text Mining: Prediction of Noun Phrases that Appear in Abstracts. In *Proceedings of the ACL Student Research Workshop*, pages 103–108, Ann Arbor, Michigan.

Hongyuan Zha. 2002. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *SIGIR*.

Yongzheng Zhang, Evangelos Milios, and Nur Zincir-Heywood. 2007. A Comparative Study on Key Phrase Extraction Methods in Automatic Web Site Summarization. *Journal of Digital Information Management*, 5(5):323.

Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming Li. 2011. Topical Keyphrase Extraction from Twitter. In *Proceedings of HLT '11*, pages 379–388, Portland, Oregon.

Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. 2001. Using temporal data for making recommendations. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, pages 580–588.

Using Mined Coreference Chains as a Resource for a Semantic Task

Heike Adel and Hinrich Schütze
Center for Information and Language Processing
University of Munich
Germany
heike.adel@cis.lmu.de

Abstract

We propose to use coreference chains extracted from a large corpus as a resource for semantic tasks. We extract three million coreference chains and train word embeddings on them. Then, we compare these embeddings to word vectors derived from raw text data and show that coreference-based word embeddings improve F_1 on the task of antonym classification by up to .09.

1 Introduction

After more than a decade of work on coreference resolution, coreference resolution systems have reached a certain level of maturity (e.g., Recasens et al. (2010)). While accuracy is far from perfect and many phenomena such as bridging still pose difficult research problems, the quality of the output of these systems is high enough to be useful for many applications.

In this paper, we propose to run coreference resolution systems on large corpora, to collect the coreference chains found and to use them as a resource for solving semantic tasks. This amounts to using mined coreference chains as an automatically compiled resource similar to the way cooccurrence statistics, dependency pairs and aligned parallel corpora are used in many applications in NLP. Coreference chains have interesting complementary properties compared to these other resources. For example, it is difficult to distinguish true semantic similarity (e.g., "cows" – "cattle") from mere associational relatedness (e.g., "cows" – "milk") based on cooccurrence statistics. In contrast, coreference chains should be able to make that distinction since only "cows" and "cattle" can occur in the same coreference chain, not "cows" and "milk".

As a proof of concept we compile a resource of mined coreference chains from the Gigaword corpus and apply it to the task of identifying antonyms. We induce distributed representations for words based on (i) cooccurrence statistics and (ii) mined coreference chains and show that a combination of both outperforms cooccurrence statistics on antonym identification.

In summary, we make two contributions. First, we propose to use coreference chains mined from large corpora as a resource in NLP and publish the first such resource. Second, in a proof of concept study, we show that they can be used to solve a semantic task – antonym identification – better than is possible with existing resources.

We focus on the task of finding antonyms in this paper since antonyms usually are distributionally similar but semantically dissimilar words. Hence, it is often not possible to distinguish them from synonyms with distributional models only. In contrast, we expect that the coreference-based representations can provide useful complementary information to this task. In general, coreference-based similarity can however be used as an additional feature for any task that distributional similarity is useful for. Thus, our coreference resource can be applied to a variety of NLP tasks, e.g. finding alternative names for entities (in a way similar to Wikipedia anchors) for tasks in the context of knowledge base population.

The remainder of the paper is organized as follows. In Section 2, we describe how we create word embeddings and how our antonym classifier works. The word embeddings are then evaluated qualitatively, quantitatively and for the task of antonym detection (Section 3). Section 4 discusses related work and Section 5 concludes.

2 System description

2.1 Coreference-based embeddings

Standard word embeddings derived from text data may not be able to distinguish between semantic

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1447–1452,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

	text-based	coref.-based
his	my, their, her, your, our	he, him, himself, zechariah, ancestor
woman	man, girl, believer, pharisee, guy	girl, prostitute, lupita, betsy, lehia

Table 1: Nearest neighbors of "his" / "woman" for text-based & coreference-based embeddings

association and true synonymy. As a result, synonyms and antonyms may be mapped to similar word vectors (Yih et al., 2012). For many NLP tasks, however, information about true synonymy or antonymy may be important.

In this paper, we develop two different word embeddings: embeddings calculated on raw text data and embeddings derived from automatically extracted coreference chains. For the calculation of the vector representations, the word2vec toolkit[1] by Mikolov et al. (2013) is applied. We use the skip-gram model for our experiments because its results for semantic similarity are better according to Mikolov et al. (2013). We train a first model on a subset of English Gigaword data.[2] In the following sections, we call the resulting embeddings *text-based*. To improve the semantic similarities of the vectors, we prepare another training text consisting of coreference chains. We use CoreNLP (Lee et al., 2011) to extract coreference chains from the Gigaword corpus. Then we build a skip-gram model on these coreference chains. The extracted coreference chains are provided as an additional resource to this paper[3]. Although they have been developed using only a publicly available toolkit, we expect this resource to be helpful for other researchers since the process to extract the coreference chains of such a large text corpus takes several weeks on multi-core machines. In total, we extracted 3.1M coreference chains. 2.7M of them consist of at least two different markables. The median (mean) length of the chains is 3 (4.0) and the median (mean) length of a markable is 1 (2.7). To train word embeddings, the markables of each coreference chain are concatenated to one text line. These lines are used as input sentences for word2vec. We refer to the resulting embeddings as *coreference-based*.

2.2 Antonym detection

In the following experiments, we use word embeddings to discriminate antonyms from non-antonyms. We formalize this as a supervised classification task and apply SVMs (Chang and Lin, 2011).

The following features are used to represent a pair of two words w and v:

1. cosine similarity of the text-based embeddings of w and v;
2. inverse rank of v in the nearest text-based neighbors of w;
3. cosine similarity of the coreference-based embeddings of w and v;
4. inverse rank of v in the nearest coreference-based neighbors of w;
5. difference of (1) and (3);
6. difference of (2) and (4).

We experiment with three different subsets of these features: text-based (1 and 2), coreference-based (3 and 4) and all features.

3 Experiments and results

3.1 Qualitative analysis of word vectors

Table 1 lists the five nearest neighbors based on cosine similarity of text-based and coreference-based word vectors for "his" and "woman".

We see that the two types of embeddings capture different notions of similarity. Unlike the text-based neighbors, the coreference-based neighbors have the same gender. The text-based neighbors are mutually substitutable words, but substitution seems to change the meaning more than for the coreference-based neighbors.

In Figure 1, we illustrate the vectors for some antonyms (connected by lines).

For reducing the dimensionality of the vector space to 2D, we applied the t-SNE toolkit[4]. It uses stochastic neighbor embedding with a Student's t-distribution to map high dimensional vectors into a lower dimensional space (Van der Maaten and Hinton, 2008). The Figure shows that the coreference-based word embeddings are able to

[1] https://code.google.com/p/word2vec

[2] LDC2012T21, Agence France-Presse 2010

[3] https://code.google.com/p/cistern

[4] http://homepage.tudelft.nl/19j49/t-SNE.html

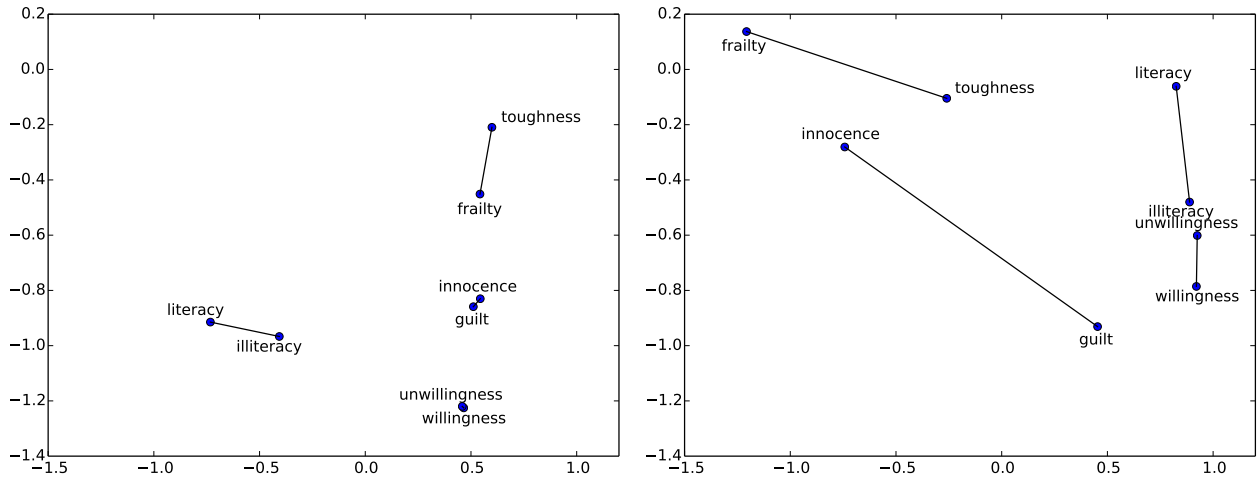


Figure 1: 2D-positions of words in the text-based (top) and coreference-based embeddings (bottom)

enlarge the distance between antonyms (especially for *guilt* vs. *innocence* and *toughness* vs. *frailty*) compared to text-based word vectors.

3.2 Quantitative analysis of word vectors

To verify that coreference-based embeddings better represent semantic components relevant to coreference, we split our coreference resource into two parts (about 85% and 15% of the data), trained embeddings on the first part and computed the cosine similarity – both text-based and coreference-based – for each pair of words occurring in the same coreference chain in the second part. The statistics in Table 2 confirm that coreference-based vectors have higher similarity within chains than text-based vectors.

3.3 Experimental setup

We formalize antonym detection as a binary classification task. Given a target word w and one of its nearest neighbors v, the classifier decides whether v is an antonym of w. Our data set is a set of pairs, each consisting of a target word w and a candidate v. For all word types of our vocabulary, we search for antonyms using the online dictionary Merriam Webster.[5] The resulting list is provided as an additional resource[6]. It contains 6225 words with antonyms. Positive training examples are collected by checking if the 500 nearest text-based neighbors of w contain one of the antonyms listed by Webster. Negative training examples are created by replacing the antonym with a random word from the 500 nearest neighbors that is not listed as an antonym. By selecting both the positive and the negative examples from the nearest neighbors of the word vectors, we intend to develop a task which is hard to solve: The classifier has to find the small portion of semantically dissimilar words (i.e., antonyms) among distributionally very similar words. The total number of positive and negative examples is 2337 each. The data are split into training (80%), development (10%) and test (10%) sets.

In initial experiments, we found only a small difference in antonym classification performance between text-based and coreference-based features. When analyzing the errors, we realized that our rationale for using coreference-based embeddings only applies to nouns, not to other parts of speech. This will be discussed in detail below. We therefore run our experiments in two modes: *all word classification* (all pairs are considered) and *noun classification* (only pairs are considered for which the target word is a noun). We use the Stanford part-of-speech tagger (Toutanova et al., 2003) to determine whether a word is a noun or not.

Our classifier is a radial basis function (rbf) support vector machine (SVM). The rbf kernel performed better than a linear kernel in initial experiments. The SVM parameters C and γ are optimized on the development set. The representation of target-candidate pairs consists of the features described in Section 2.

3.4 Experimental results and discussion

We perform the experiments with the three different feature sets described in Section 2: text-based, coreference-based and all features. Table 3 shows

[5] http://www.merriam-webster.com

[6] https://code.google.com/p/cistern

	all word classification						noun classification					
	development set			test set			development set			test set		
feature set	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
text-based	**.83**	**.66**	**.74**	.74	.55	.63	**.91**	.61	.73	.74	.51	.60
coreference-based	.67	.42	.51	.65	.43	.52	.86	.47	.61	.77	.45	.57
text+coref	.79	.65	.72	**.75**	**.58**	**.66**	.88	**.70**	**.78**	**.79**	**.61**	**.69**

Table 3: Results for different feature sets. Best result in each column in bold.

	minimum	maximum	median
text-based vectors	-0.350	0.998	0.156
coref.-based vectors	-0.318	0.999	0.161

Table 2: Cosine similarity of words in the same coreference chain

results for development and test sets.

For all word classification, coreference-based features do not improve performance on the development set (e.g., F_1 is .74 for text-based vs .72 for text+coref). On the test set, however, the combination of all features (text+coref) has better performance than text-based alone: .66 vs .63.

For noun classification, using coreference-based features in addition to text-based features improves results on development set (F_1 is .78 vs .73) and test set (.69 vs .60).

These results show that mined coreference chains are a useful resource and provide information that is complementary to other methods. Even though adding coreference-based embeddings improves performance on antonym classification, the experiments also show that using only coreference-based embeddings is almost always worse than using only text-based embeddings. This is not surprising given that the amount of training data for the word embeddings is different in the two cases. Coreference chains provide only a small subset of the word-word relations that are given to the word2vec skip-gram model when applied to raw text. If the sizes of the training data sets were similar in the two cases, we would expect performance to be comparable.

In the beginning, our hypothesis was that coreference information should be helpful for antonym classification in general. When we performed an error analysis for our initial results, we realized that this hypothesis only holds for nouns. Other types of words cooccurring in coreference chains are not more likely to be synonyms than words cooccurring in text windows. Two contexts that illustrate this point are "*bright sides*, but also *difficult and dark ones*" and "a series of *black and white shots*" (elements of coreference chains in italics). Thus, adjectives with opposite meanings can cooccur in coreference chains just as they can cooccur in window-based contexts. For nouns, it is much less likely that the same coreference chain will contain both a noun and its antonym since – by definition – markables in a coreference chain refer to the same identical entity.

4 Related work

Traditionally, words have been represented by vectors of the size of the vocabulary with a one at the word index and zeros otherwise (one-hot vectors). However, this approach cannot handle unknown words (Turian et al., 2010) and similarities among words cannot be represented (Mikolov et al., 2013). Therefore, distributed word representations (embeddings) become more and more popular. They are low-dimensional, real-valued vectors. Mikolov et al. (2013) have published word2vec, a toolkit that provides different possibilities to estimate word embeddings (cbow model and skip-gram model). They show that the resulting word vectors capture semantic and syntactic relationships of words. Baroni et al. (2014) show that word embeddings are able to outperform count based word vectors on a variety of NLP tasks. Recently, Levy and Goldberg (2014) have generalized the skip-gram model to include not only linear but arbitrary contexts like contexts derived from dependency parse trees. Andreas and Klein (2014) investigate the amount of additional information continuous word embeddings could add to a constituency parser and find that most of their information is redundant to what can be learned from labeled parse trees. In (Yih et al., 2012), the vector space representation of words is modified so that high positive similarities are assigned to synonyms and high negative similarities to antonyms. For this, latent semantic analysis is applied to a matrix of thesaurus entries. The val-

ues representing antonyms are negated.

There has been a great deal of work on applying the vector space model and cosine similarity to find synonyms or antonyms. Hagiwara et al. (2006) represent each word as a vector with cooccurrence frequencies of words and contexts as elements, normalized by the inverse document frequency. The authors investigate three types of contextual information (dependency, sentence cooccurrence and proximity) and find that a combination of them leads to the most stable results. Schulte im Walde and Köper (2013) build a vector space model on lexico-syntactic patterns and apply a Rocchio classifier to distinguish synonyms from antonyms, among other tasks. Van der Plas and Tiedemann (2006) use automatically aligned translations of the same text in different languages to build context vectors. Based on these vectors, they detect synonyms.

In contrast, there are also studies using linguistic knowledge from external resources: Senellart and Blondel (2008) propose a method for synonym detection based on graph similarity in a graph generated using the definitions of a monolingual dictionary. Harabagiu et al. (2006) recognize antonymy by generating antonymy chains based on WordNet relations. Mohammad et al. (2008) look for the word with the highest degree of antonymy to a given target word among five candidates. For this task, they use thesaurus information and the similarity of the contexts of two contrasting words. Lin et al. (2003) use Hearst patterns to distiguish synonyms from antonyms. Work by Turney (2008) is similar except that the patterns are learned.

Except for the publicly available coreference resolution system, our approach does not need external resources such as dictionaries or bilingual corpora and no human labor is required. Thus, it can be easily applied to any corpus in any language as long as there exists a coreference resolution system in this language. The pattern-based approach (Lin et al., 2003; Turney, 2008) discussed above also needs few resources. In contrast to our work, it relies on patterns and might therefore restrict the number of recognizable synonyms and antonyms to those appearing in the context of the pre-defined patterns. On the other hand, patterns could explicitely distinguish contexts typical for synonyms from contexts for antonyms. Hence, we plan to combine our coreference-based method with pattern-based methods in the future.

5 Conclusion

In this paper, we showed that mined coreference chains can be used for creating word embeddings that capture a type of semantic similarity that is different from the one captured by standard text-based embeddings. We showed that coreference-based embeddings improve performance of antonym classification by .09 F_1 compared to using only text-based embeddings. We achieved precision values of up to .79, recall values of up to .61 and F_1 scores of up to .69.

Acknowledgments

This work was supported by DFG (grant SCHU 2246/4-2).

References

Jacob Andreas and Dan Klein. 2014. How much do word embeddings encode about syntax? In *ACL*, pages 822–827.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, pages 238–247.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Masato Hagiwara, Yasuhiro Ogawa, and Katsuhiko Toyama. 2006. Selection of effective contextual information for automatic synonym acquisition. In *COLING/ACL*, pages 353–360.

Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2006. Negation, contrast and contradiction in text processing. In *AAAI*, volume 6, pages 755–762.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *CoNLL: Shared Task*, pages 28–34.

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

Dekang Lin, Shaojun Zhao, Lijuan Qin, and Ming Zhou. 2003. Identifying synonyms among distributionally similar words. In *IJCAI*, pages 1492–1493.

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

Saif Mohammad, Bonnie Dorr, and Graeme Hirst. 2008. Computing word-pair antonymy. In *EMNLP*, pages 982–991.

Marta Recasens, Lluís Màrquez, Emili Sapena, M Antònia Martí, Mariona Taulé, Véronique Hoste, Massimo Poesio, and Yannick Versley. 2010. Semeval-2010 task 1: Coreference resolution in multiple languages. In *5th International Workshop on Semantic Evaluation*, pages 1–8.

Sabine Schulte im Walde and Maximilian Köper. 2013. Pattern-based distinction of paradigmatic relations for German nouns, verbs, adjectives. In *Language Processing and Knowledge in the Web*, pages 184–198. Springer.

Pierre Senellart and Vincent D Blondel. 2008. Automatic discovery of similar words. In *Survey of Text Mining II*, pages 25–44. Springer.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL-HLT*, pages 252–259.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *ACL*, pages 384–394.

Peter D. Turney. 2008. A uniform approach to analogies, synonyms, antonyms, and associations. In *COLING*, pages 905–912.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11):2579–2605.

Lonneke Van der Plas and Jörg Tiedemann. 2006. Finding synonyms using automatic word alignment and measures of distributional similarity. In *COLING/ACL*, pages 866–873.

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

Financial Keyword Expansion via Continuous Word Vector Representations

Ming-Feng Tsai
Department of Computer Science &
Program in Digital Content and Technology
National Chengchi University
Taipei 116, Taiwan
mftsai@nccu.edu.tw

Chuan-Ju Wang
Department of Computer Science
University of Taipei
Taipei 100, Taiwan
cjwang@utaipei.edu.tw

Abstract

This paper proposes to apply the continuous vector representations of words for discovering keywords from a financial sentiment lexicon. In order to capture more keywords, we also incorporate syntactic information into the Continuous Bag-of-Words (CBOW) model. Experimental results on a task of financial risk prediction using the discovered keywords demonstrate that the proposed approach is good at predicting financial risk.

1 Introduction

In the present environment with a great deal of information, how to discover useful insights for decision making is becoming increasingly important. In finance, there are typically two kinds of information (Petersen, 2004): *soft information* usually refers to text, including opinions, ideas, and market commentary, whereas *hard information* is recorded as numbers, such as financial measures and historical prices. Most financial studies related to risk analysis are based on hard information, especially on time series modeling (Christoffersen and Diebold, 2000; Lee and Tong, 2011; Wu et al., 2014; Yümlü et al., 2005). Despite of using only hard information, some literature incorporates soft textual information to predict financial risk (Kogan et al., 2009; Leidner and Schilder, 2010; Tsai and Wang, 2013). Moreover, sentiment analysis, a technique to make an assessment of the sentiments expressed in various information, has also been applied to analyze the soft textual information in financial news, reports, and social media data (Devitt and Ahmad, 2007; Loughran and McDonald, 2011; Wang et al., 2013).

Continuous vector space models (Bengio et al., 2003; Schwenk, 2007; Mikolov et al., 2010) are neural network language models, in which words are represented as high dimensional real valued vectors. These representations have recently demonstrated promising results across variety of tasks (Schwenk, 2007; Collobert and Weston, 2008; Glorot et al., 2011; Socher et al., 2011; Weston et al., 2011), because of their superiority of capturing syntactic and semantic regularities in language. In this paper, we apply the Continuous Bag-of-Words (CBOW) model (Mikolov et al., 2013) on the soft textual information in financial reports for discovering keywords via financial sentiments. In specific, we use the continuous vector representations of words to find out similar terms based on their contexts. Additionally, we propose a straightforward approach to incorporate syntactic information into the CBOW model for better locating similarly meaningful or highly correlated words. To the best of our knowledge, this is the first work to incorporate more syntactic information by adding Part-Of-Speech (POS) tags to the words before training the CBOW model.

In our experiments, the corpora are the annual SEC[1]-mandated financial reports, and there are 3,911 financial sentiment keywords for expansion. In order to verify the effectiveness of the expanded keywords, we then conduct two prediction tasks, including regression and ranking. Observed from our experimental results, for the regression and ranking tasks, the models trained on the expanded keywords are consistently better than those trained the original sentiment keywords only. In addition, for comparison, we conduct experiments with random keyword expansion as baselines. According to the experimental results, the expansion either with or without syntactic information outperforms the baselines. The results suggest that the CBOW model is effective at expanding keywords for financial risk prediction.

[1] Securities and Exchange Commission

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1453–1458,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

2 Keyword Expansion via Financial Sentiment Lexicon

2.1 Financial Sentiment Lexicon

A sentiment lexicon is the most important resource for sentiment analysis. Loughran and McDonald (2011) states that a general purpose sentiment lexicon (e.g., the Harvard Psychosociological Dictionary) might misclassify common words in financial texts. Therefore, in this paper, we use a finance-specific lexicon that consists of the 6 word lists provided by (Loughran and McDonald, 2011) as seeds to expand keywords. The six lists are negative (Fin-Neg), positive (Fin-Pos), uncertainty (Fin-Unc), litigious (Fin-Lit), strong modal words (MW-Strong), and weak modal words (MW-Weak).[2]

2.2 Simple Keyword Expansion

With the financial sentiment lexicon, we first use a collection of financial reports as the training texts to learn continuous vector representations of words. Then, each word in the sentiment lexicon is used as a seed to obtain the words with the highest n cosine distances (called the top-n words for the word) via the learned word vector representations. Finally, we construct an expanded keyword list from the top-n words for each word.

2.3 Keyword Expansion with Syntactic Information

For the expansion considering syntactic information, we attach the POS tag to each word in the training texts first. Then, the words in the sentiment lexicon with 4 major POS tags (i.e., `JJ`, `NN`, `VB`, `RB`) are used as seeds to expand. The rest of steps is similar to that in Section 2.2.

The reason of considering POS tags for expansion is that, in general, a word with different POS tags may result in different lists of top-n words. Table 1 shows the top-5 words for the word "default" with different POS tags (noun and adjective). Note that none of the words in the two lists overlaps.

default (`NN`)		default (`JJ`)	
Word	Cosine Distance	Word	Cosine Distance
default (v.)	0.63665	nonconform (v.)	0.63462
unwaiv (v.)	0.63466	subprim (v.)	0.62404
uncur (v.)	0.62285	chattel (n.)	0.61510
trigger (n.)	0.60080	foreclos (adj.)	0.61397
unmatur (v.)	0.58208	unguarante (v.)	0.60559

Table 1: Top-5 Words for the word "default."

3 Financial Risk Prediction

3.1 The Risk Measure: Volatility

Volatility is a measure for variation of prices of a stock over a period of time. Let S_t be the price of a stock at time t. Holding the stock from time $t-1$ to time t would lead to a simple return: $R_t = S_t/S_{t-1} - 1$ (Tsay, 2005). The volatility of returns for a stock from time $t-n$ to t can thus be defined as follows:

$$v_{[t-n,t]} = \sqrt{\frac{\sum_{i=t-n}^{t} (R_i - \bar{R})^2}{n}}, \qquad (1)$$

where $\bar{R} = \sum_{i=t-n}^{t} R_i/(n+1)$.

3.2 Regression Task

Given a collection of financial reports $D = \{\mathbf{d_1}, \mathbf{d_2}, \ldots, \mathbf{d_n}\}$, in which each $\mathbf{d_i} \in \mathbb{R}^p$ and is associated with a company c_i, we aim to predict the future risk of each company c_i (which is characterized by its volatility v_i). This prediction problem can be defined as follows:

$$\hat{v}_i = f(\mathbf{d_i}; \mathbf{w}). \qquad (2)$$

The goal is to learn a p-dimensional vector $\mathbf{w}$ from the training data $T = \{(\mathbf{d}_i, v_i) | \mathbf{d}_i \in \mathbb{R}^p, v_i \in \mathbb{R}\}$. In this paper, we adopt the Support Vector Regression (SVR) (Drucker et al., 1997) for training such a regression model. More details about SVR can be found in (Schölkopf and Smola, 2001).

3.3 Ranking Task

Instead of predicting the volatility of each company in the regression task, the ranking task aims to rank companies according to their risk via the textual information in their financial reports. We first split the volatilities of company stock returns within a year into different risk levels by the mechanism provided in (Tsai and Wang, 2013). The risk levels can be considered as the relative difference of risk among the companies.

After obtaining the relative risk levels of the companies, the ranking task can be defined as follows: Given a collection of financial reports D, we aim to rank the companies via a ranking model $f : \mathbb{R}^p \rightarrow \mathbb{R}$ such that the rank order of the set of companies is specified by the real value that the

[2] `http://www.nd.edu/~mcdonald/Word_Lists.html`.

model f takes. Specifically, $f(\mathbf{d_i}) > f(\mathbf{d_j})$ means that the model asserts that $c_i \succ c_j$, where $c_i \succ c_j$ means that c_i is ranked higher than c_j; that is, the company c_i is more risky than c_j. For this task, this paper adopts Ranking SVM (Joachims, 2006).

4 Experiments

4.1 Dataset and Preprocessings

In the experiments, we use the 10-K corpus (Kogan et al., 2009) to conduct our financial risk prediction tasks. All documents and the 6 financial sentiment word lists are stemmed by the Porter stemmer (Porter, 1980), and some stop words are also removed.

For financial risk prediction, the ground truth, the twelve months after the report volatility for each company, $v^{+(12)}$, (which measures the future risk for each company) can be calculated by Equation (1), where the stock prices can be obtained from the Center for Research in Security Prices (CRSP) US Stocks Database. In addition, to obtain the relative risks among companies used in the ranking task, we follow (Tsai and Wang, 2013) to split the companies of each year into 5 risk levels.

4.2 Keyword Expansion

In our experiments, Section 7 (Management Discussion and Analysis) in 10-K corpus is used as training texts for the tool (word2vec[3]) to learn the continuous vector representations of words.

For the simple expansion (denoted as EXP−SIM hereafter), we use the total 1,667 stemmed sentiment words as seeds to obtain the expanded keywords via the learned word vector representations. For the expansion considering syntactic information (denoted as EXP−SYN), NLTK[4] is applied to attach the POS tag[5] to each word in the training texts; we attach the POS tag to a word with an underscore notation (e.g., default_VB). For simplicity, we combine some POS tags to one tag via the tag replacement; for example, the tags JJR (adjective, comparative) and JJS (adjective, superlative) are replaced to JJ (adjective). The detailed replacement rules are tabulated in Table 2. Words from the sentiment lexicon with the four types of POS tags (i.e., JJ, NN, VB, RB) are consider as the seeds to expand the keywords. For both EXP−SIM and EXP−SYN, we use the top-20 expanded words for each word (e.g., Table 3) to construct expanded keyword lists. In total, for EXP−SIM, the expanded list contains 9,282 unique words and for EXP−SYN, the list has 13,534 unique words.

[3] https://code.google.com/p/word2vec/

[4] http://www.nltk.org/

[5] The most common POS tag scheme, the Penn Treebank POS Tags, is adopt in the paper.

After Replacement	Before Replacement
JJ	JJ, JJR, JJS
NN	NN, NNS, NNP, NNPS
PRP	PRP, PRP$
RB	RB, RBR, RBS
VB	VB, VBD, VBG, VBN, VBP, VBZ
WP	WP, WP$

Table 2: Tag Replacement Rules.

Word	Cosine Distance	Word	Cosine Distance
uncur	0.569498	event	0.466834
indentur	0.565450	lender	0.459995
waiv	0.563656	forbear	0.456556
trigger	0.559936	represent	0.450631
cure	0.539999	breach	0.446851
nonpay	0.538445	noncompli	0.431490
unmatur	0.525251	gecc	0.430712
unwaiv	0.510359	customari	0.424447
insolv	0.488534	waiver	0.419338
occurr	0.471123	prepay	0.418969

Table 3: Top-20 (Stemmed) Words for the Word "default."

4.3 Word Features

In the experiments, the bag-of-words model is adopted and three word features are used to represent the 10-K reports in the experiments. Given a document $\mathbf{d}$, three word features (i.e., TF, TFIDF and LOG1P) are calculated as follows:

- $\texttt{TF}(t, \mathbf{d}) = \texttt{TC}(t, \mathbf{d})/|\mathbf{d}|$,
- $\texttt{TFIDF}(t, \mathbf{d}) = \texttt{TF}(t, \mathbf{d}) \times \texttt{IDF}(t, \mathbf{d}) = \texttt{TC}(t, \mathbf{d})/|\mathbf{d}| \times \log(|D|/|\mathbf{d} \in D : t \in \mathbf{d}|)$,
- $\texttt{LOG1P} = \log(1 + \texttt{TC}(t, \mathbf{d}))$,

where $\texttt{TC}(t, \mathbf{d})$ denotes the term count of t in $\mathbf{d}$, $|\mathbf{d}|$ is the length of document $\mathbf{d}$, and D denotes the set of all documents in each year.

4.4 Experimental Results

Tables 4 and 5 tabulate the experimental results of regression and ranking, respectively, in which the training data is composed of the financial reports in a five-year period, and the following year is the test data. For example, the reports from year 1996 to 2000 constitute a training data, and the learned model is tested on the reports of year 2001.

[TFIDF] Year	SEN	(Baseline) EXP−RAN	EXP−SIM	EXP−SYN	SEN	(Baseline) EXP−RAN	EXP−SIM	EXP−SYN
	Kendall's Tau (Kendall, 1938).				Spearman's Rho (Myers et al., 2003)			
2001	0.4384	0.4574	0.4952	**0.5049**	0.4701	0.4889	0.5266	**0.5375**
2002	0.4421	0.4706	0.4881	**0.4944**	0.4719	0.5007	0.5187	**0.5256**
2003	0.4414	0.4706	**0.5105**	0.5006	0.4716	0.5015	**0.5418**	0.5318
2004	0.4051	0.4551	0.4750	**0.4961**	0.4335	0.4842	0.5043	**0.5255**
2005	0.3856	0.4482	0.5126	**0.5294**	0.4117	0.4757	0.5418	**0.5579**
2006	0.3784	0.4385	0.4588	**0.4867**	0.4029	0.4641	0.4847	**0.5129**

Table 5: Performance of Ranking.

[LOGP] Year	SEN	(Baseline) EXP−RAN	EXP−SIM	EXP−SYN
	Mean Squared Error			
2001	0.2526	0.2360	0.2195	**0.2148**
2002	0.2858	0.2649	0.2433	**0.2381**
2003	0.2667	0.2512	0.2320	**0.2350**
2004	0.2345	0.2140	0.1902	**0.1872**
2005	0.2241	0.2014	0.1754	**0.1682**
2006	0.2256	0.2072	0.1889	**0.1825**

Table 4: Performance of Regression

In the tables, SEN denotes the experiments trained on the words from the original financial sentiment lexicon. Despite of the experiments trained on EXP−SIM and EXP−SYN, we also conduct experiments with random keyword expansion (called EXP−RAN); for the comparison purpose, we keep the number of words in the randomly expanded word list the same as that in EXP−SYN. Note that the randomly expanded list contains all sentiment words and the rest of words are randomly chosen from the vocabulary of the dataset. The columns with label EXP−RAN denote the results averaged from 20 randomly expanded word lists. The bold face numbers denote the best performance among the four word lists.

As shown in Tables 4 and 5, for both regression and ranking tasks, the models trained on expanded keywords (i.e., EXP−$\star$) are consistently better than those trained on the original sentiment keywords only.[6] Additionally, we treat the experiments with randomly expanded word list (EXP−RAN) as the baselines.[7] From the two tables, we observe that the expansion either with or without syntactic information outperforms the baselines. Note that, for the EXP−SIM, the number of words used for training the regression and ranking models is even less than that of EXP−RAN. The results suggest that the CBOW model is effective at expanding keywords for financial risk prediction. Furthermore, incorporating syntactic information into the CBOW model can even enhance the performance for the tasks of financial risk prediction.

[6]Due to the page limits, only the results trained on features LOGP for regression and TFIDF for ranking are reported, but the performance for models trained on features TF, TFIFG, and LOGP is very consistent.

[7]The results for EXP−SYN are all significant better than the baseline with $p < 0.05$.

4.5 Discussions

Below we provide the original texts from 10-K reports that contain the top 1 expanded word, "uncur" (stemmed), for "default" in Table 3. Two pieces of sentences are listed as follows (the company Investment Technology Group, 1997):

> $\cdots$ terminate the agreement upon certain events of bankruptcy or insolvency or upon an **uncured** breach by the Company of certain covenants $\cdots$

> $\cdots$ any termination of the license agreement resulting from an **uncured** default would have a material adverse effect on the Company's results of operations.

From the above examples, the expanded word "uncur" has similar meaning to "default," which confirms the capability of our method of capturing similarly meaningful or highly correlated words.

5 Conclusions

This paper applies the continuous bag-of-words model on the textual information in financial reports for expanding keywords from a financial sentiment lexicon. Additionally, we propose a simple but novel approach to incorporate syntactic information into the continuous bag-of-words model for capturing more similarly meaningful or highly correlated keywords. The experimental results for the risk prediction problem show that the expansion either with or without syntactic information outperforms the baselines. As a direction for further

research, it is interesting and important to provide more analysis on the expanded words via the continuous vector representations of words.

Acknowledgments

This research was partially supported by the National Science Council of Taiwan under the grants NSC 102-2420-H-004-052-MY2, 102-2221-E-004-006, and 102-2221-E-845-002-MY3.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Peter F Christoffersen and Francis X Diebold. 2000. How relevant is volatility forecasting for financial risk management? *Review of Economics and Statistics*, 82(1):12–22.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, IMCL '08, pages 160–167.

Ann Devitt and Khurshid Ahmad. 2007. Sentiment polarity identification in financial news: A cohesion-based approach. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, ACL '07, pages 984–991.

Harris Drucker, Chris JC Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. 1997. Support vector regression machines. *Advances in Neural Information Processing Systems*, 9:155–161.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning*, ICML '11, pages 513–520.

Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 217–226.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30:81–93.

Shimon Kogan, Dimitry Levin, Bryan R Routledge, Jacob S Sagi, and Noah A Smith. 2009. Predicting risk from financial reports with regression. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 272–280.

Yi-Shian Lee and Lee-Ing Tong. 2011. Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming. *Knowledge-Based Systems*, 24(1):66–72.

Jochen L. Leidner and Frank Schilder. 2010. Hunting for the black swan: risk mining from text. In *Proceedings of the ACL 2010 System Demonstrations*, ACLDemos '10, pages 54–59.

Tim Loughran and Bill McDonald. 2011. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance*, 66(1):35–65.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of Interspeech*, pages 1045–1048.

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

Jerome L Myers, Arnold D Well, and Robert F Lorch Jr. 2003. *Research design and statistical analysis*. Routledge.

Mitchell A Petersen. 2004. Information: Hard and soft. Technical report, Northwestern University.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137.

Bernhard Schölkopf and Alexander J Smola. 2001. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning*, ICML '11, pages 129–136.

Ming-Feng Tsai and Chuan-Ju Wang. 2013. Risk ranking from financial reports. In *Advances in Information Retrieval*, pages 804–807. Springer.

Ruey S Tsay. 2005. *Analysis of financial time series*. Wiley.

Chuan-Ju Wang, Ming-Feng Tsai, Tse Liu, and Chin-Ting Chang. 2013. Financial sentiment analysis for risk prediction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, IJCNLP '13, pages 802–808.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of the Twenty-Second international joint conference on Artificial*

Intelligence-Volume Volume Three, pages 2764–2770.

Desheng Dash Wu, Shu-Heng Chen, and David L Olson. 2014. Business intelligence in risk management: Some recent progresses. *Information Sciences*, 256:1–7.

Serdar Yümlü, Fikret S Gürgen, and Nesrin Okay. 2005. A comparison of global, recurrent and smoothed-piecewise neural models for istanbul stock exchange (ise) prediction. *Pattern Recognition Letters*, 26(13):2093–2103.

Intrinsic Plagiarism Detection using N-gram Classes

Imene Bensalem
MISC Lab
Constantine 2 University, Algeria
bens.imene@gmail.com

Paolo Rosso
NLE Lab
PRHLT Research Center
Universitat Politècnica de València, Spain
prosso@dsic.upv.es

Salim Chikhi
MISC Lab
Constantine 2 University, Algeria
slchikhi@yahoo.com

Abstract

When it is not possible to compare the suspicious document to the source document(s) plagiarism has been committed from, the evidence of plagiarism has to be looked for intrinsically in the document itself. In this paper, we introduce a novel language-independent intrinsic plagiarism detection method which is based on a new text representation that we called n-gram classes. The proposed method was evaluated on three publicly available standard corpora. The obtained results are comparable to the ones obtained by the best state-of-the-art methods.

1 Introduction and Related Works

Intrinsic plagiarism detection is an essential alternative in situations where the plagiarism source does not have a digital version, e.g. an old book, or the plagiarized text was directly written by another author without copying from any source, e.g. the case of a student who asked someone else to write for him parts of his essay or thesis. Hence, the task of detecting plagiarism intrinsically is to identify, in the given suspicious document, the fragments that are not consistent with the rest of the text in terms of writing style.

The automatic analysis of the writing style is an important component of many NLP applications. For some of them, when analyzing the style, a document is considered as a whole, which is the case of the authorship identification (Stamatatos, 2009a) and the authorship verification (Koppel and Seidman, 2013). For other applications, a document is perceived as a set of fragments, for each of them the writing style needs to be analyzed individually. Examples of such applications include: paragraph authorship clustering (Brooke and Hirst, 2012), authorial segmentation of multi-author documents (Akiva and Koppel, 2013), detection of stylistic inconsistencies between consecutive paragraphs (Graham et al., 2005) and plagiarism direction identification (Grozea and Popescu, 2010).

For intrinsic plagiarism detection, it is crucial to analyze the writing style at fragments level. However, the majority of methods tend to analyze the whole document writing style as well. Indeed, intrinsic plagiarism detection puts together, in one research problem, many difficulties that are not present, or present separately, in the aforementioned related problems. Its main difficulties are listed below.

In contrast to multi-author documents related problems, the number of authors in the suspicious documents is unknown, i.e., it might be one author if the document is plagiarism-free or many unknown authors if it contains plagiarism.

Unlike the authorship attribution and verification, where the examined text and the potential author text are separate (and hence their writing styles could be readily characterized and compared), these two parts are both merged in the same document with unknown boundaries. Furthermore, the plagiarized fragments in a suspicious document might stem from different authors, which renders the computational characterization of plagiarism difficult.

As opposed to the problem of authorship clustering, where the task is merely to attribute already defined fragments of a given document to different authors, the segmentation is a crucial and inevitable task in a real scenario of intrinsic plagiarism detection. Indeed, a granular segmentation may lead to an undependable style analysis, and a coarse segmentation may prevent the identification of the short plagiarized texts.

Due to the aforementioned difficulties, intrinsic plagiarism detection is still a challenging

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1459–1464,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

problem. This is evidenced by the still low performance scores of the majority of methods[1]. To the best of our knowledge, just two methods, namely Stamatatos (2009b) and Oberreuter et al. (2011), reached an f-measure greater than 0.30 on a standardized corpus. Other methods, for instance (Stein et al., 2011) and (Tschuggnall and Specht, 2013), obtained better performance scores. Nonetheless, they have been evaluated on only selected documents from the whole standardized evaluation corpus which makes their results not comparable to the others.

Although the writing style analysis is an old research area and has been applied successfully to solve many problems, notably authorship attribution, it is obvious that its application to identify the plagiarized fragments still needs to be investigated further. In this paper, we address this research problem by proposing a novel way of quantifying the writing style that we called n-gram classes. We show that our method, which is supervised classification-based, is able to discriminate between the plagiarized and the original text fragments with a performance comparable to the best state-of-the-art methods despite it uses a small number of features when building the classification model.

The remainder of the paper is organized as follows. Section 2 presents our motivation. Sections 3 and 4 present the new features and the proposed method. Section 5 provides the evaluation results. Finally, Section 6 draws our conclusions.

2 Motivation

The idea of our method is inspired by the work of Grozea and Popescu (2010), in the context of plagiarism direction identification. They reported that the character 8-grams of a plagiarized text fragment are more frequent in the source document (because the author is the same) than in the plagiarized document. Thus, we believe that, it is possible to distinguish the plagiarized fragments from the original ones on the basis of the frequency of their character n-grams in the suspicious document. That is, if many of the character n-grams of a fragment are infrequent in the document, it would be probably a plagiarized fragment. However, if many of them are frequent, then the fragment is likely to be original.

On the other hand, according to the authorship attribution researches, character n-grams are a powerful tool for characterizing the writing style (Stamatatos, 2009a). Moreover, they have been used in one of the best intrinsic plagiarism detection methods (Stamatatos, 2009b).

Generally, in n-gram based methods the text is represented by a vector of n-grams with their frequencies. The shortcoming of this text representation is the increase of its size with the increase of the text or the n-gram length.

Our method proposes a novel way of using character n-grams[2] for text representation. The idea is to represent the fragments of the suspicious document in a reduced vector where each feature value is the frequency of a class of n-grams instead of a particular n-gram. Therefore, the dimension of any fragment vector is always equal to the number of classes rather than the number of n-grams. The class of an n-gram is determined according to its frequency level in the given document as we will show in the next section.

3 N-gram Classes

Formally, we define an n-gram class as a number from 0 to $m-1$ such that the class labeled 0 involves the *least frequent* n-grams and the class labeled $m-1$ contains the *most frequent* n-grams in a document. If $m > 2$, classes between 0 and $m-1$ will contain n-grams with *intermediate frequency levels*.

Concretely, to assign the n-grams of a given document to m classes, first, the document is represented by a $2 \times l$ matrix (l is the total number of n-grams), where the first row contains the n-grams ng_i (i =1..l) and the second one contains their number of occurrences $freq_i$ (raw frequency).

Let *max_freq* denotes the maximum frequency, so:

$$max\_freq = \text{argmax} \quad freq_i ; \quad i=1..l \qquad (1)$$

Then, the class of a n-gram ng_i is computed as follows:

$$\text{Class } ng_i = \text{Log}_{\text{base}} (freq_i); \qquad (2)$$

Given that:

$$\text{base} = \sqrt[m-1]{max\_freq} \quad . \qquad (3)$$

By computing the base of the logarithm as shown in the equation (3), the most frequent n-grams (i.e. the n-grams with the maximum number of occurrences) will be in the class $m-1$, and

[1] See for instance PAN workshop (http://pan.webis.de) series, from 2007 to 2012, where several papers on intrinsic plagiarism detection have been published.

[2] In the rest of the paper, when not said differently, the term n-gram is always used to denote character n-gram.

the least frequent n-grams (e.g. the ones that appear only once) will be in the class 0, and the n-grams with intermediate levels of frequency will be in the classes between 0 and $m-1$. Figure 1 illustrates an example of computing the n-gram classes of a document. The chosen number of classes m in this example is 3.

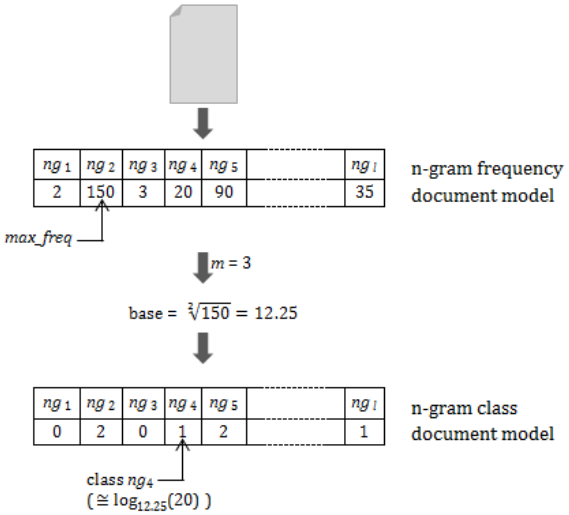


Figure 1. Steps for computing the n-gram classes of a document. The number of classes in this example is 3 (class labels are from 0 to 2).

Note that, what we explained above is solely how to compute the class of each n-gram of a document. However, our purpose is to represent the document fragments using these classes. To this end, for each fragment, first, its n-grams are extracted. Then, each n-gram is replaced by its class obtained from the document model built previously. Finally, the proportion of each class in the fragment is computed. So, the fragment can be represented by a vector of m values, where the first value is the proportion of the class 0, the second value is the proportion of the class 1 and so on. Figure 2 illustrates these steps. For the sake of simplicity, we suppose that the fragment contains only 5 n-grams.

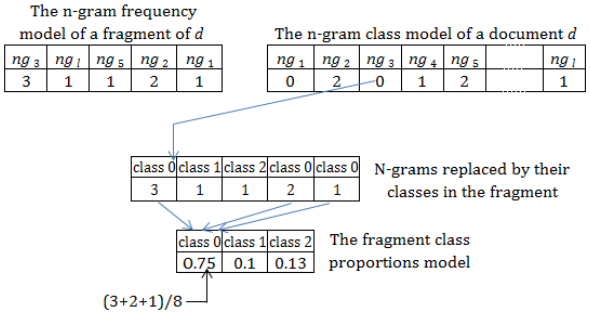


Figure 2. Steps for representing a document fragment by the proportion of 3 n-gram classes.

4 The Proposed Method

Once the suspicious document has been segmented to fragments and these latter have been represented by a set of features, an important phase in the process of the intrinsic plagiarism detection is to decide whether a fragment is plagiarized or original. This phase has been implemented in the literature methods using different techniques, notably clustering (Akiva, 2011), supervised classification (Meyer zu Eissen et al., 2007), distance functions with thresholds (Stamatatos, 2009b; Oberreuter et al., 2011) and density-based methods (Stein et al., 2011).

In our supervised method, the classification model is trained with a small number of features which are the proportions of the n-gram classes described in the previous section.

In detail, our method is composed of the following steps:

1. Segment each document d into fragments s_i by using the sliding window technique. Let S denotes the set of these fragments.
2. Build the n-gram class document model (see Figure 1) without considering numerals. We choose to consider the frequency of a n-gram ng_i as the number of its occurrence in d such that it is counted once per fragment. Therefore, the minimum value that could take a frequency is 1 if ng_i appears only in one fragment, and its maximum value is $|S|$ (the number of fragments in d) if ng_i occurs in each fragment $s_i \in S$.
3. Represent each fragment s_i by a vector of m features f_j , $j \in \{0,\ldots, m-1\}$. So that, each f_j is the proportion of the n-grams that belong to the class labeled j to the total number of n-grams in s_i.
4. Combine into one dataset the fragment vectors obtained from all the training corpus documents. Then, label each vector with its authenticity state, i.e. plagiarized, if the fragment plagiarism percentage exceeds 50% and original otherwise.
5. Build a classifier using the training set produced in the previous step. For this purpose, we trained and tested several classification algorithms implemented on WEKA software (Hall et al., 2009). The best results were obtained with the Naïve Bayes algorithm[3].

The aforementioned steps represent the training phase of our method, which aims to construct the classifier. In practice, in order to detect the plagiarism in a given document, this classifier is

[3] Consult the arff file from the archive file associated to this paper which contains the fragments class proportion model and the plagiarism prediction for each fragment.

directly applied to the document fragments after the step 3.

5 Evaluation

5.1 Datasets

We evaluated our method on 3 corpora: PAN-PC-09[4] and PAN-PC-11[5] which are the corpora used in the international competition of plagiarism detection in 2009 and 2011 respectively[6], as well as InAra corpus[7], which is a publicly available collection of artificial suspicious documents in Arabic (Bensalem et al., 2013). The three document collections include XML annotations indicating the plagiarized segments positions.

For the evaluation on English and Spanish documents, the classifier has been trained on PAN-PC-11 test corpus and evaluated on this same corpus using 10-fold cross validation as well as PAN-PC-09 test corpus. For the evaluation on Arabic documents, the classifier has been trained and tested on InAra corpus using 10-fold cross validation.

5.2 Results

As evaluation measures we used macro-averaged precision, recall, f-measure, granularity and plagdet as they were defined in (Potthast et al., 2010).

In order to choose the parameters of our method, we trained the classifier using various training sets generated by using the different combinations of the n-gram length *n* (from 1 to 10) and the number of classes *m* (from 2 to 10). We adopted the parameters that yielded the higher f-measure, namely $n = 6$ and $m = 4$.

With regard the sliding window parameters, we used three different options for the window size, which are 100, 200 and 400 words, with a step equal to the quarter of the window size. Only one option is applied to a given document depending on its length.

We deliberately use similar sliding window parameters as the method of Oberreuter et al. (2011)[8] in order to compare the two methods without being much affected by the segmentation strategy.

Table 1 compares the results of our method to the one of Oberreuter et al. (2011) being the winner in PAN 2011 competition and considered one of the best intrinsic plagiarism detection methods.

		Our method	Oberreuter et al.[9]
PAN-PC-09	Precision	0.31	**0.39**
	Recall	**0.49**	0.31
	F-measure	**0.38**	0.35
	Granularity	1.21	**1.00**
PAN-PC-11	Precision	0.22	**0.34**
	Recall	**0.50**	0.31
	F-measure	0.30	**0.33**
	Granularity	1.13	**1.00**
InAra	Precision	0.24	**0.29**
	Recall	**0.69**	0.25
	F-measure	**0.35**	0.27
	Granularity	**1.27**	1.44

Table 1. Performance of the n-gram frequency class method on 3 corpora.

From Table 1 it can be appreciated that our method in terms of recall noticeably outperforms Oberreuter et al. (2011), although precision and granularity still needs to be further improved. Nonetheless, in comparison with other methods such as the one of Stamatatos (2009b), that obtained the best results in PAN 2009 competition on plagiarism detection, precision is still very much competitive: 0.31 vs. 0.23 (PAN-PC-09) and 0.22 vs. 0.14 (PAN-PC-11). In terms of f-measure, Oberreuter et al. (2011) method is significantly higher than our method on PAN-PC-11 corpus, but both methods have statistically similar results on InAra[10].

Considering plagdet, which is a score that represents the overall performance of a plagiar-

[4] http://www.uni-weimar.de/en/media/chairs/webis/research/corpora/corpus-pan-pc-09/

[5] http://www.uni-weimar.de/en/media/chairs/webis/research/corpora/corpus-pan-pc-11/

[6] We used only the corpora parts that are dedicated to the evaluation of the intrinsic approach.

[7] http://sourceforge.net/projects/inaracorpus/

[8] Oberreuter et al. (2011) used mainly 400 words as the window size that may change according to the document length.

[9] The results of Oberreuter et al. method (2011) on PAN-PC-09 and PAN-PC-11 are taken from his paper. However, we re-implemented this method in order to evaluate it on InAra. Note that our re-implementation maybe not perfectly similar to the original one since the authors did not provide details on the parameters tuning.

[10] The Kolomogorov Smirnov test with a significance level of 5% has been used to compare the two methods f-measures on PAN-PC-11 and InAra. Unfortunately, on the PAN-PC-09 corpora we were unable to carry out this test since we do not have the results of Oberreuter et al. per each document.

ism detection method, our method could be ranked the 2nd, after Oberreuter et al. (2011) and before Stamatatos (2009b) as it is shown in Table 2.

	Oberreuter et al.	Our method	Stamatatos
PAN-PC-09	0.35	0.33	0.25
PAN-PC-11	0.33	0.28	0.19

Table 2. Plagdet of our method in comparison with the two best methods on PAN competition corpora.

6 Conclusion

In this paper we have shown that representing the text fragments of a given suspicious document by the proportion of character n-gram classes (the most frequent, the least frequent and intermediate levels) is a promising way for detecting plagiarism intrinsically.

The experiments described in this paper were performed on three corpora comprising documents in English, Spanish and for the first time Arabic. We obtained comparable results to the best performing systems.

Our method best configuration is 6 as the n-grams length and only 4 as the number of classes (i.e. 4 features). As future work, it would be interesting to combine the most precise classes of different n-gram lengths in order to improve the precision. It would be important as well to try other segmentation strategies and post-processing techniques in order to improve the granularity. Another interesting experiment we plan to carry out in the future is to use the n-gram classes along with the traditional stylistic features such as the vocabulary richness, average sentence length, etc.

Acknowledgments

The first author would like to thank Parth Gupta for his helpful feedback and Gabriel Oberreuter for providing some implementation details of his method.

The work of the second author was carried out in the framework of DIANA APPLICATIONS-Finding Hidden Knowledge in Texts: Applications (TIN2012-38603-C02-01) and WIQ-EI IRSES (Grant No. 269180 within the EC FP 7 Marie Curie People) research projects, and the VLC/CAMPUS Microcluster on Multimodal Interaction in Intelligent Systems.

References

Navot Akiva. 2011. Using Clustering to Identify Outlier Chunks of Text - Notebook for PAN at CLEF 2011. In *Notebook Papers of CLEF 2011 LABs and Workshops, September 19-22, Amsterdam, The Netherlands*, pages 5–7.

Navot Akiva and Moshe Koppel. 2013. A Generic Unsupervised Method for Decomposing Multi-Author Documents. *Journal of the American Society for Information Science and Technology*, 64(11):2256–2264.

Imene Bensalem, Paolo Rosso, and Salim Chikhi. 2013. A New Corpus for the Evaluation of Arabic Intrinsic Plagiarism Detection. In Pamela Forner, Henning Müller, Roberto Paredes, Paolo Rosso, and Benno Stein, editors, *CLEF 2013, LNCS, vol. 8138*, pages 53–58, Heidelberg. Springer.

Julian Brooke and Graeme Hirst. 2012. Paragraph Clustering for Intrinsic Plagiarism Detection using a Stylistic Vector-Space Model with Extrinsic Features - Notebook for PAN at CLEF 2012. In *CLEF 2012 Evaluation Labs and Workshop – Working Notes Papers, 17-20 September, Rome, Italy*.

Neil Graham, Graeme Hirst, and Bhaskara Marthi. 2005. Segmenting Documents by Stylistic Character. *Natural Language Engineering*, 11(04):397–415.

Cristian Grozea and Marius Popescu. 2010. Who ' s the Thief ? Automatic Detection of the Direction of Plagiarism. In *CICLing 2010, Iaşi, Romania, March 21-27, LNCS, vol. 6008*, pages 700–710. Springer.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1):10–18.

Moshe Koppel and Shachar Seidman. 2013. Automatically Identifying Pseudepigraphic Texts. In *EMNLP 2013*, pages 1449–1454, Seattle,Washington, USA. Association for Computational Linguistics.

Sven Meyer zu Eissen, Benno Stein, and Marion Kulig. 2007. Plagiarism Detection without Reference Collections. In Reinhold Decker and Hans -J. Lenz, editors, *Advances in Data Analysis, Selected Papers from the 30th Annual Conference of the German Classification Society (GfKl), Berlin*, pages 359–366, Heidelberg. Springer.

Gabriel Oberreuter, Gaston L'Huillier, Sebastián A. Ríos, and Juan D. Velásquez. 2011. Approaches for Intrinsic and External Plagiarism Detection - Notebook for PAN at CLEF 2011. In *CLEF 2011 Evaluation Labs and Workshop – Working Notes*

Papers, September 19-22, Amsterdam, The Netherlands, pages 1–10.

Martin Potthast, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso. 2010. An Evaluation Framework for Plagiarism Detection. In Chu-Ren Huang and Daniel Jurafsky, editors, *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 997–1005, Stroudsburg, USA. Association for Computational Linguistics.

Efstathios Stamatatos. 2009a. A Survey of Modern Authorship Attribution Methods. *Journal of the American Society for Information Science*, 60(3):538–556.

Efstathios Stamatatos. 2009b. Intrinsic Plagiarism Detection Using Character n-gram Profiles. In Benno Stein, Paolo Rosso, Efstathios Stamatatos, Moshe Koppel, and Eneko Agirre, editors, *Proceedings of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN 09)*, pages 38–46. CEUR-WS.org.

Benno Stein, Nedim Lipka, and Peter Prettenhofer. 2011. Intrinsic Plagiarism Analysis. *Language Resources and Evaluation*, 45(1):63–82.

Michael Tschuggnall and Günther Specht. 2013. Using Grammar-Profiles to Intrinsically Expose Plagiarism in Text Documents. In *NLDB 2013, LNCS, vol. 7934*, pages 297–302. Springer.

Verifiably Effective Arabic Dialect Identification

Kareem Darwish, Hassan Sajjad, Hamdy Mubarak
Qatar Computing Research Institute
Qatar Foundation
{kdarwish,hsajjad,hmubarak}@qf.org.qa

Abstract

Several recent papers on Arabic dialect identification have hinted that using a word unigram model is sufficient and effective for the task. However, most previous work was done on a standard fairly homogeneous dataset of dialectal user comments. In this paper, we show that training on the standard dataset does not generalize, because a unigram model may be tuned to topics in the comments and does not capture the distinguishing features of dialects. We show that effective dialect identification requires that we account for the distinguishing lexical, morphological, and phonological phenomena of dialects. We show that accounting for such can improve dialect detection accuracy by nearly 10% absolute.

1 Introduction

Modern Standard Arabic (MSA) is the lingua franca of the so-called Arab world, which includes northern Africa, the Arabian Peninsula, and Mesopotamia. However, Arabic speakers generally use dramatically different languages (or dialects) in daily interactions and in social media. These dialects may differ in vocabulary, morphology, and spelling from MSA and most do not have standard spellings. There is often large lexical overlap between dialects and MSA. Performing proper Arabic dialect identification may positively impact many Natural Language Processing (NLP) application. For example, transcribing dialectal speech or automatically translating into a particular dialect would be aided by the use of targeted language models that are trained on texts in that dialect.

This has led to recent interest in automatic identification of different Arabic dialects (Elfardy et al., 2013; Cotterell et al., 2014; Zaidan et al., 2014). Though previous work (Cotterell et al., 2014) have reported high accuracies for dialect identification using word unigram model, which implies that this is a solved problem, we argue that the problem is far from being solved. The reason for this assertion stems from the fact that the available dialectal data is drawn from singular sources, namely online news sites, for each dialect. This is problematic because comments on singular news site are likely to have some homogeneity in topics and jargon. Such homogeneity has caused fairly simple classification techniques that use word unigrams and character n-grams to yield very high identification accuracies. Perhaps, this can be attributed to topical similarity and not just differences between dialects. To showcase this, we trained a classifier using the best reported methods, and we tested the classifier on a new test set of 700 tweets, with dialectal Egyptian (ARZ) and MSA tweets, which led to a low accuracy of 83.3%. We also sorted words in the ARZ part from our training dataset by how much they discriminate between ARZ and MSA (using mutual information) and indeed many of the top words were in fact MSA words.

There seems to be a necessity to identify lexical and linguistic features that discriminate between MSA and different dialects. In this paper, we highlight some such features that help in separating between MSA and ARZ. We identify common ARZ words that do not overlap with MSA and identify specific linguistic phenomena that exist in ARZ, and not MSA, such as morphological patterns, word concatenations, and verb negation constructs (Section 3). We also devise methods for capturing the linguistic phenomena, and we use the appearance of such phenomena as features (Section 4). Further, we show the positive impact of using the new features in identifying ARZ (Section 5).

2 Previous Work

Previous work on Arabic dialect identification uses n-gram based features at both word-level and character-level to identify dialectal sentences (Elfardy et al., 2013; Cotterell et al., 2014; Zaidan et al., 2011; Zaidan et al., 2014). Zaidan et al. (2011) created a dataset of dialectal Arabic. They performed cross-validation experiments for dialect identification using word n-gram based features. Elfardy et al. (2013) built a system to distinguish between ARZ and MSA. They used word n-gram features combined with core (token-based and perplexity-based features) and meta features for training. Their system showed a 5% improvement over the system of Zaidan et al. (2011). Later, Zaidan et al. (2014) used several word n-gram based and character n-gram based features for dialect identification. The system trained on word unigram-based feature performed the best with character five-gram-based feature being second best. A similar result is shown by Cotterell et al. (2014) where word unigram model performs

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1465–1468,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

the best.

All of the previous work except Cotterell et al. (2014)[1] evaluate their systems using cross-validation. These models heavily rely on the coverage of training data to achieve better identification. This limits the robustness of identification to genres inline with the training data.

Language identification is a related area to dialect identification. It has raised some of the issues which we discussed in this paper in the context of dialect identification. Lui et al. (2011) showed that in-domain language identification performs better than cross domain language identification. Tiedemann et al. (2012) argued that the linguistic understanding of the differences between languages can lead to a better language identification system. kilgarriff (2001) discussed the differences between datasets as a poor representation of differences between dialects of English.

In this paper, we exploit the linguistic phenomena that are specific to Arabic dialects to show that they produce significant improvements in accuracy. We show that this also helps in achieving high quality cross-domain dialect identification system.

3 Dialectal Egyptian Phenomena

There are several phenomena in ARZ that set it apart from MSA. Some of them are as follows:

Dialectal words: ARZ uses unique words that do not overlap with MSA and may not overlap with other dialects. Some of the common ARZ words are: "zy" (like), "kdh" (like this), and "Azyk" (how are you) [2]. These dialectal terms stem from the following:

• Using proper Arabic words that are rarely used in MSA such as "$nTp" (bag) and "n$wf" (we see).

• Fusing multiple words together by concatenating and dropping letters such as the word "mEl$" (no worry), which is a fusion of "mA Elyh $y' ".

• Using non-standard spelling of words such as "SAbE" (finger) instead of "<sbE" in MSA. Consequently, broken plurals may also be non-standard.

• using non-Arabic words such as "<y$Arb" (scarf), which is transliterated from the French écharpe.

• altering the forms of some pronouns such as the feminine second person pronoun from "k" to "ky", the second person plural pronoun "tm" to "tw", and the object pronoun "km" to "kw".

Morphological differences: ARZ makes use of particular morphological patterns that do not exist in MSA and often alters some morphological constructs. Some examples include:

• Adding the letter "b" in front of verb in present tense. Ex. MSA: "ylEb" (he plays) → EG: "bylEb".

• Using the letters "H" or "h", instead of "s", to indicate future tense. Ex. MSA: "sylEb" (he will play) → EG: "hylEb" or "HylEb".

• Adding the letters "At" to passive past tense verbs. Ex. MSA: "luEiba" (was played) → "AtlaEab".

• Adding the letters "m" or "mA" before the verb and "$" or "$y" after the verb to express negation. Ex. MSA: "lm ylEb" (he did not play) → "mlEb$".

• the merging of verbs and prepositional phrases of the form (to-pronoun) that follow it. Ex. MSA: "ylEb lh" (he plays for/to him) → "bylEblh".

• Replacing a short vowel with a long vowel in imperative verbs that are derived from hollow roots. Ex. MSA: "qul" (say) → "qwl".

Letter substitution: in ARZ the following letter substitutions are common:

• "v" → "t". Ex. MSA: "kvyr" (a lot) → EG: "ktyr".

• "}" → "y". Ex. MSA: "b}r" (well) → "byr".

• Trailing "y" → "Y". Ex. MSA: "Hqy" (my right) → "HqY".

• "*" → "d". Ex. MSA: "xu*" (take) → "xud".

• middle or trailing ">" → "A". Ex. MSA: "f>r" (mouse) → "fAr".

• "D" → "Z". Ex. MSA: "DAbT" (officer) → "ZAbT".

• "Z" → "D". Ex. MSA: "Zhr" (back) → "Dhr".

• Middle "|" → "yA". Ex. MSA: "ml|n" (full) → "mlyAn".

• Removal of trailing " ' ". Ex. MSA: "AlsmA' " (the sky) → "AlsmA".

Syntactic differences: some of the following phenomena are generally observed:

• Common use of masculine plural or singular noun forms instead dual and feminine plural. Ex. MSA "jnyhyn" (two pounds) → EG: "Atnyn jnyh".

• Dropping some articles and preposition in some syntactic constructs. For example, the preposition "<lY" (to) in ">nA rAyH <lY Al$gl" (I am going to work) is typically dropped. Also, the particle ">n" (to) is dropped in the sentence ">nA mHtAj >n >nAm" (I need to sleep).

• Using only one form of noun and verb suffixes such as "yn" instead of "wn" and "wA" instead of "wn" respectively. Also, so-called "five nouns", are used in only one form (ex. ">bw" (father of) instead of ">bA" or ">by").

4 Detecting Dialectal Peculiarities

ARZ is different from MSA lexically, morphologically, phonetically, and syntactically. Here, we present methods to handle such peculiarities. We chose not to handle syntactic differences, because they may be captured using word n-gram models.

To capture lexical variations, we extracted and sorted by frequency all the unigrams from the Egyptian side of the LDC2012T09 corpus (Zbib et al., 2012), which has ≈ 38k Egyptian-English parallel sentences. A linguist was tasked with manually reviewing the words from the top until 1,300 dialectal words were found. Some of the words on the list included dialectal words, commonly used foreign words, words that exhibit morphological variations, and others with letter substitution.

[1] Zaidan et al. (2014) applied their classifier to a different genre but did not evaluate it's performance.

[2] Buckwalter encoding is used throughout the paper.

For morphological phenomenon, we employed three methods, namely:

• **Unsupervised Morphology Induction:** We employed the unsupervised morpheme segmentation tool, Morfessor (Virpioja et al., 2013). It is a data driven tool that automatically learns morphemes from data in an unsupervised fashion. We used the trained model to segment the training and test sets.

• **Morphological Rules:** In contrast to Morfessor, we developed only 15 morphological rules (based on the analysis proposed in Section 3) to segment ARZ text. These rules would separate prefixes and suffixes like a light stemmer. Example rules would segment a leading "b" and segment a combination of a leading "m" and trailing "$".

• **Morphological Generator:** For morphological generation, we enumerated a list of ≈ 200 morphological patterns that derive dialectal verbs from Arabic roots. One such pattern is ytCCC that would generate the dialectal verb-form ytktb (to be written) from the root "ktb". We used the root list that is distributed with Sebawai (Darwish, 2002). We also expanded the list by attaching negation affixes and pronouns. We retained generated word forms that: a) exist in a large corpus of 63 million Arabic tweets from 2012 with more than 1 billion tokens; and b) don't appear in a large MSA corpus of 10 years worth of Aljazeera articles containing 114 million tokens [3]. The resulting list included 94k verb surface forms such as "mbyEmlhA$" (he does not do it).

For phonological differences, we used a morphological generator that makes use of the aforementioned root list and an inventory of ≈ 605 morphological patterns (with diacritization) to generate possible Arabic stems. The generated stems with their diacritics were checked against a large diacritized Arabic corpus containing more than 200 million diacritized words [4]. If generated words contained the letters "v", "}", "*", and "D", we used the aforementioned letter substitutions. We retained words that exist in the large tweet corpus but not in the Aljazeera corpus. The list contained 8k surface forms.

5 Evaluation Setup

Dataset: We performed dialect identification experiment for ARZ and MSA. For ARZ, we used the Egyptian side of the LDC2012T09 corpus (Zbib et al., 2012) [5]. For MSA, we used the Arabic side of the English/Arabic parallel corpus from the International Workshop on Arabic Language Translation[6] which consists of ≈ 150k sentences. For testing, we constructed an evaluation set that is markedly different from the training set. We crawled Arabic tweets from Twitter during March 2014 and selected those where user location was set to Egypt or a geographic location within Egypt, leading to 880k tweets. We randomly selected 2k tweets, and we manually annotated them as ARZ, MSA, or neither until we obtained 350 ARZ and 350 MSA tweets. We used these tweets for testing. We plan to release the tweet ID's and our annotations. We preprocessed the training and test sets using the method described by Darwish et al. (2012), which includes performing letter and word normalizations, and segmented all data using an open-source MSA word segmentor (Darwish et al., 2012). We also removed punctuations, hashtags, and name mentions from the test set. We used a Random Forest (RF) ensemble classifier that generates many decision trees, each of which is trained on a subset of the features.[7] We used the RF implementation in Weka (Breiman, 2001).

5.1 Classification Runs

Baseline *BL*: In our baseline experiments, we used word unigram, bigram, and trigram models and character unigram to 5-gram models as features. We first performed a cross-validation experiment using ARZ and MSA training sets. The classifier achieved fairly high results (+95%) which are much higher than the results mentioned in the literature. This could be due in part to the fact that we were doing ARZ-MSA classification instead of multi-dialect classification and MSA data is fairly different in genre from ARZ data. We did not further discuss these results. This experiment was a sanity check to see how does in-domain dialect identification perform.

Later, we trained the RF classifier on the complete training set using word n-gram features (WRD), character n-gram features (CHAR), or both (BOTH) and tested it on the tweets test set. We referred to this system as *BL* later on.

Dialectal Egyptian Lexicon S_{lex}: As mentioned earlier, we constructed three word lists containing 1,300 manually reviewed ARZ words (MAN), 94k dialectal verbs (VERB), and 8k words with letter substitutions (SUBT). Using the lists, we counted the number of words in a tweet that exist in the word lists and used it as a standalone feature for classifications. LEX refers to concatenation of all three lists.

Morphological Features: For S_{mrph}, we trained Morfessor separately on the MSA and Egyptian training data and applied to the same training data for segmentation. For S_{rule}, we segmented Egyptian part of the training data using the morphological rules mentioned in Section 4. For both, word and character n-gram features were calculated from the segmented data and the

[3] http://aljazeera.net

[4] http://www.sh.rewayat2.com

[5] We did not use the Arabic Online Commentary data (Zaidan et al., 2011) as annotations were often not reliable.

[6] https://wit3.fbk.eu/mt.php?release=2013-01

[7] We tried also the multi-class Bayesian classifier and SVM classifier. SVM classifier had comparable results with Random Forest classifier. However, it was very slow. So, we decided to go with Random Forest classifier for the rest of the experiments.

SYS	WRD	CHR	BOTH	BEST+LEX
BL	53.0	74.0	83.3	84.7
S_{mrph}	72.0	88.0	62.1	89.3
S_{rule}	53.9	85.9	85.9	90.1

Table 1: Dialect identification accuracy using various classification settings: only word-based (WRD), character-based (CHAR), and both features. BEST+LEX is built on the best feature of that system plus a feature built on the concatenation of all lists

SYS	MAN	+VERB	+SUBT
S_{lex}	93.6	94.6	94.4

Table 2: Accuracy of the dialect identification system with the addition of various types of lexicon

classifier was trained on them and tested on the tweet test set.

5.2 Results

Table 1 summarizes the results. Unlike results in the literature, character-based n-gram features outperformed word-based n-gram features, as they seemed to better generalize to the new test set, where lexical overlap between the training and test sets was low. Except for S_{mrph}, adding both character and word n-gram features led to improved results. We observed that Morfessor over-segmented the text, which in turns created small character segments and enabled the character-based language model to learn the phenomenon inherit in a word. The baseline system achieved an accuracy of 84.7% when combined with the S_{lex} feature. Combining S_{mrph} and S_{rule} features with the S_{lex} feature led to further improvement. However, as shown in Table 2, using the S_{lex} feature alone with the MAN and VERB lists led to the best results (94.6%), outperforming using all other features either alone or in combination. This suggests that having a clean list of dialectal words that cover common dialectal phenomena is more effective than using word and character n-grams. It also highlights the shortcomings of using a homogeneous training set where word unigrams could be capturing topical cues along with dialectal ones.

6 Conclusion

In this paper, we identified lexical, morphological, phonological, and syntactic features that help distinguish between dialectal Egyptian and MSA. Given the substantial lexical overlap between dialectal Egyptian and MSA, targeting words that exhibit distinguishing traits is essential to proper dialect identification. We used some of these features for dialect detection leading to nearly 10% (absolute) improvement in classification accuracy. We plan to extend our work to other dialects.

References

Leo Breiman. 2001. Random Forests. Machine Learning. 45(1):5-32.

Ryan Cotterell, Chris Callison-Burch. 2014. A Multi-Dialect, Multi-Genre Corpus of Informal Written Arabic. LREC-2014, pages 241–245.

Kareem Darwish. 2002. Building a shallow morphological analyzer in one day. In Proceedings of the ACL-2002 Workshop on Computational Approaches to Semitic Languages.

Kareem Darwish, Walid Magdy, Ahmed Mourad. 2012. Language Processing for Arabic Microblog Retrieval. CIKM-2012, pages 2427–2430.

Kareem Darwish, Ahmed Abdelali, Hamdy Mubarak. 2014. Using Stem-Templates to improve Arabic POS and Gender/Number Tagging. LREC-2014.

Heba Elfardy, Mona Diab. 2013. Sentence Level Dialect Identification in Arabic. ACL-2013, pages 456–461.

Sami Virpioja, Peter Smit, Stig-Arne Grnroos, and Mikko Kurimo. 2013. Morfessor 2.0: Python Implementation and Extensions for Morfessor Baseline. Aalto University publication series SCIENCE + TECHNOLOGY, 25/2013. Aalto University, Helsinki, 2013.

Omar F. Zaidan, Chris Callison-Burch. 2011. The Arabic Online Commentary Dataset: An Annotated Dataset of Informal Arabic with High Dialectal Content. ACL-11, pages 37–41.

Omar F. Zaidan, Chris Callison-Burch. 2014. Arabic Dialect Identification. CL-11, 52(1).

Rabih Zbib, Erika Malchiodi, Jacob Devlin, David Stallard, Spyros Matsoukas, Richard Schwartz, John Makhoul, Omar F. Zaidan, Chris Callison-Burch. 2012. Machine translation of Arabic dialects. NAACL-2012, pages 49–59.

Marco Lui and Timothy Baldwin. 2011. Cross-domain feature selection for language identification. IJCNLP-2011, page 553–561.

Jörg Tiedemann and Nikola Ljubesic. 2012. Efficient discrimination between closely related languages. COLING-2012, 2619–2634.

Adam Kilgarriff. 2001. Comparing corpora. CL-01, 6(1).

Keystroke Patterns as Prosody in Digital Writings: A Case Study with Deceptive Reviews and Essays

Ritwik Banerjee Song Feng Jun S. Kang
Computer Science
Stony Brook University
{rbanerjee, songfeng, junkang}
@cs.stonybrook.edu

Yejin Choi
Computer Science & Engineering
University of Washington
yejin@cs.washington.edu

Abstract

In this paper, we explore the use of keyboard strokes as a means to access the real-time writing process of online authors, analogously to prosody in speech analysis, in the context of deception detection. We show that differences in keystroke patterns like editing maneuvers and duration of pauses can help distinguish between truthful and deceptive writing. Empirical results show that incorporating keystroke-based features lead to improved performance in deception detection in two different domains: online reviews and essays.

1 Introduction

Due to the practical importance of detecting deceit, interest in it is ancient, appearing in papyrus dated back to 900 B.C. (Trovillo, 1939). In more recent years, several studies have shown that the deceiver often exhibits behavior that belies the content of communication, thus providing cues of deception to an observer. These include linguistic (*e.g.,* Newman et al. (2003), Hancock et al. (2004)) as well as paralinguistic (*e.g.,* Ekman et al. (1991), DePaulo et al. (2003)) cues. Recognizing deception, however, remains a hard task for humans, who perform only marginally better than chance (Bond and DePaulo, 2006; Ott et al., 2011).

Recent studies suggest that computers can be surprisingly effective in this task, albeit in limited domains such as product reviews. Prior research has employed lexico-syntactic patterns (Ott et al., 2011; Feng et al., 2012) as well as online user behavior (Fei et al., 2013; Mukherjee et al., 2013). In this paper, we study the effect of keystroke patterns for deception detection in digital communications, which might be helpful in understanding the psychology of deception and help toward trustful online communities. This allows us to investigate differences in the writing and revisional processes of truthful and fake writers. Our work thus shares intuition with HCI research linking keystroke analysis to cognitive processes (Vizer et al., 2009; Epp et al., 2011) and psychology research connecting cognitive differences to deception (Ekman, 2003; Vrij et al., 2006).

Recent research has shown that lying generally imposes a cognitive burden (*e.g.,* McCornack (1997), Vrij et al. (2006)) which increases in real-time scenarios (Ekman, 2003). Cognitive burden has been known to produce differences in keytroke features (Vizer et al., 2009; Epp et al., 2011). Previous research has not, however, directly investigated any quantitative connection between keystroke patterns and deceptive writing.

In this paper, we posit that cognitive burdens in deception may lead to measurable characteristics in keystroke patterns. Our contributions are as follows: (1) introducing keystroke logs as an extended linguistic signal capturing the real-time writing process (analogous to prosody in speech analysis) by measuring the *writing rate*, *pauses* and *revision rate*. (2) showing their empirical value in deception detection, (3) providing novel domain-specific insights into deceptive writing, and (4) releasing a new corpus of deception writings in new domains.[1]

2 Related Work

Prior research has focused mainly on using keystroke traits as a behvioral biometric. Forsen et al. (1977) first demonstrated that users can be distinguished by the way they type their names. Subsequent work showed that typing patterns are *unique to individuals* (Leggett and Williams, 1988), and can be used for authentication (Cho et al., 2000; Bergadano et al., 2002) and intrusion detection (Killourhy and Maxion, 2009).

Keystroke pauses have been linked to linguistic patterns in discourse (e.g. Matsuhashi (1981), van Hell et al. (2008)) and regarded as indications of cognitive burden (*e.g.,* Johansson (2009), Zulkifli (2013)). In this paper, we present the *first* empirical study that quantitatively measures the deception cues in real-time writing process as manifested in keystroke logs.

3 Data Collection

As discussed by Gokhman et al. (2012), the crowdsourcing approach to soliciting deceptive content simulates the real world of online deceptive content creators. We collected the data via Amazon Mechanical Turk.[2] Turkers were led to a separate website where keylogging was enabled, and asked to write truthful and deceptive texts ($\geq$ 100 words) on one of three top-

[1] Available at http://www3.cs.stonybrook.edu/~junkang/keystroke/

[2] https://www.mturk.com/mturk

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1469–1473,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

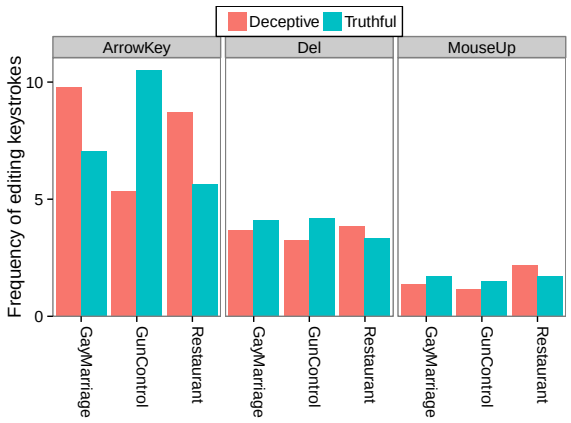


Figure 1: Number of keystrokes corresponding to the three types of edit patterns ($\mathcal{E}_3$): (a) use of arrow keys, (b) deletion (Delete and Backspace) and (c) text selection with mouse.

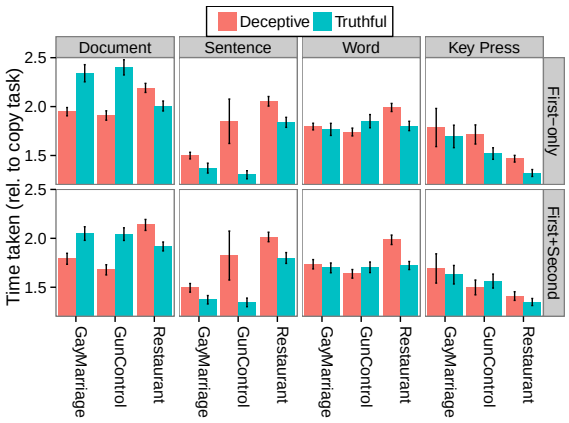


Figure 2: Average normalized timespan $\widehat{\delta}(e)$ for documents, sentences, words and key presses. The top row considers only the first text, while the bottom row considers both flows.

ics: *restaurant review*, *gay marriage* and *gun control*. Each Turker was required to agree to their typing being logged. Since copy/paste operations defeat our purpose of studying keystrokes in the typing process, they were disabled. This restriction also acts as a hindrance to plagiarism. All texts were reviewed manually, and those not meeting the requirements (due to the being too short, plagiarized content, etc.) were disregarded.

Writing task design: The task was designed such that each Turker wrote a pair of texts, one truthful and one deceptive, on the same topic. For restaurant reviews, they were asked to write a truthful review of a restaurant they liked, and a deceptive review of a restaurant they have never been to or did not like. For the other two topics – 'gun control' and 'gay marriage' – we asked their opinion: *support*, *neutral*, or *against*. Then, they were asked to write a truthful and a deceptive essay articulating, respectively, their actual opinion and its opposite.[3] The tasks further were divided into two 'flows': writing the truthful text before the deceptive one, and vice versa. Each Turker was assigned only one flow, and was not allowed to participate in the other. After completing this, each Turker was asked to *copy* their own typing, *i.e.,* re-type the two texts.

Finally, in order to get an idea of the cognitive burden associated with truthful and deceptive writing, we asked the Turkers which task was easier for them. Of the 196 participants, 152 answered "truthful", 40 answered "deceptive" and only 4 opted for "not sure".

What are logged: We deployed a keylogger to capture the mouse and keyboard events in the "text area". The events *KeyUp*, *KeyDown* and *MouseUp*, along with the keycode and timestamp were logged.[4] For the three topics *restaurant review*, *gay marriage* and *gun control* we obtained 1000, 800 and 800 texts, respectively.

In the remainder of this paper, k^{dn} and k^{up} denote the KeyDown and KeyUp events for a key k. For any event e, its *timespan*, *i.e.,* the time interval between the beginning and end of e, is denoted by $\delta(e)$.

[3]To prevent a change in opinion depending on task availability, Turkers were redirected to other tasks if their opinion was neutral, or if we had enough essays of their opinion.

[4]Printable (*e.g.,* alphanumeric characters) as well as non-printable keystrokes like (*e.g.,* 'Backspace'), are logged.

4 Feature Design

Keystroke logging enables the study of two types of information that go beyond conventional linguistic analysis. First, it captures editing processes (*e.g.,* deletions, insertions made by changing cursor position, etc.). Second, it reveals the temporal aspect of text generation (*e.g.,* duration, latency). Our exploration of these features and their application in deception detection is motivated by the similarities between text and speech generation. Editing patterns, for instance, can be viewed as attempts to veil *incoherence* in deceptive writing and temporal patterns like latency or pause can be treated as analogous to *disfluency*.

Different people, of course, have varying typing skills, and some may type faster than others. In order to control for this variation, we normalize all event timespans $\delta(e)$ with respect to the corresponding event timespan in the *copy* task: $\widehat{\delta}(e) = \delta(e)/\delta(e_{copy})$.

4.1 Editing Patterns

In this work, we treat keys that are used only for editing as different from others. Text editing is done by employing a small subset of available keys: deletion keys ('Backspace' and 'Delete'), arrow keys ($\leftarrow$, $\rightarrow$, $\uparrow$ and $\downarrow$) and by using the mouse for text selection (*i.e.,* the 'MouseUp' event). The three types of editing keystrokes are collectively denoted by

$$\mathcal{E}_3 = \langle |\text{DEL}|, |\text{MSELECT}|, |\text{ARROW}| \rangle$$

where

(i) $|\text{DEL}|$ = number of deletion keystrokes
(ii) $|\text{MSELECT}|$ = number of 'MouseUp' events, and
(iii) $|\text{ARROW}|$ = number of arrow keystrokes

The editing differences between truthful and deceptive writing across all three topics are shown in Fig. 1.

4.2 Temporal Aspects

Each event is logged with a keycode and a timestamp. In order to study the temporal aspects of digital writing, we calculate the timespan of different linguistic

Topic	Features	Flow	
		First + Second	First-only
Restaurants	BoW	73.9	78.8
	BoW + $\mathcal{T}_6$	74.3	79.1
	BoW + $\mathcal{T}_6 + \mathcal{E}_3$	74.6	**80.3***
Gun Control (Support)	BoW	86.5	80.0
	BoW + $\mathcal{T}_6$	86.8	82.5*
	BoW + $\mathcal{T}_6 + \mathcal{E}_3$	**88.0**§	83.5*
Gun Control (Oppose)	BoW	88.5	88.0
	BoW + $\mathcal{T}_6$	89.8	87.5
	BoW + $\mathcal{T}_6 + \mathcal{E}_3$	**90.8***	89.1
Gay Marriage (Support)	BoW	92.5	92.0
	BoW + $\mathcal{T}_6$	93.8	92.5
	BoW + $\mathcal{T}_6 + \mathcal{E}_3$	**94.3***	92.0
Gay Marriage (Oppose)	BoW	84.5	86.5
	BoW + $\mathcal{T}_6$	85.0	**87.0**
	BoW + $\mathcal{T}_6 + \mathcal{E}_3$	85.3	86.8

Table 1: SVM classifier performance for truthful vs. deceptive writing. Statistically significant improvements over the baseline are marked * ($p < 0.05$) and § ($p < 0.1$). $\mathcal{E}_3 = \langle |\text{DEL}|, |\text{MSELECT}|, |\text{ARROW}| \rangle$ denotes the editing keystrokes, and $\mathcal{T}_6$ is the set of normalized timespans of documents, words (plus preceding keystroke), all keystrokes, spaces, non-whitespace keystrokes and inter-word intervals: $\mathcal{T}_6 = \{\widehat{\delta}(\text{D}), \widehat{\delta}(k), \widehat{\delta}(\text{SP}), \widehat{\delta}(\neg\text{SP}), \widehat{\delta}(\neg\text{W}), \widehat{\delta}(k_{prv} + \text{W})\}$

units such as words, sentences and even entire documents. Further, we separately inspect the timespans of different parts of speech, function words and content words. In addition to event timespans, intervals between successive events (*e.g.,* inter-word and inter-sentence pauses) and pauses preceding or succeeding and event (*e.g.,* time interval before and after a function word) are measured as well.

5 Experimental Results

This section describes our experimental setup and presents insights based on the obtained results. All classification experiments use 5-fold cross validation with 80/20 division for training and testing. In addition to experimenting on the entire dataset, we also separately analyze the texts written first (of the two texts in each 'flow'). This additional step is taken in order to eliminate the possibility of a text being primed by its preceding text.

Deception cues in keystroke patterns: To empirically check whether keystroke features can help distinguish between truthful and deceptive writing, we design binary SVM classifiers.[5] Unigrams with `tf-idf` encoding is used as the baseline. The average baseline accuracy across all topics is 82.58% when considering both texts of a flow, and 83.62% when considering only the first text of each flow. The better performance in the latter possibly indicates that the second text of a flow exhibits some amount of lexical priming with the first.

The high accuracy of the baseline is not surprising. Previous work by Ott et al. (2011) reported similar performance of unigram models. The focus of our work is to explore the completely new feature space of typographic patterns in deception detection. We draw motivation from parallels between the text generation and speech generation processes. Prosodic concepts such as *speed*, *disfluency* and *coherence* can be realized in typographic behavior by analyzing timestamp of keystrokes, pauses and editing patterns, respectively.

[5]We use the LIBLINEAR (Fan et al., 2008) package.

$\widehat{\delta}(\text{W})$		$\widehat{\delta}(k_{prv} + \text{W})$	
D > T	T > D	D > T	T > D
our	best	when	one
if	get	quality	other
when	well	even	get
were	your	on	service
it's	fresh	by	been
quality	not	me	their
dishes	my	has	not
the	one	also	with
i've	had	go	friendly
on	hat	we	great
they	of	had	an
we	other	is	our
friendly	very	at	are
has	love	which	really
at	service	from	but
wait	great	dishes	favorite
an	really	or	very
go	you	re	about
is	but	would	will
which	been	just	here

Table 2: Top 20 words in **restaurant reviews** with greatest timespan difference between deceptive and truthful writing.

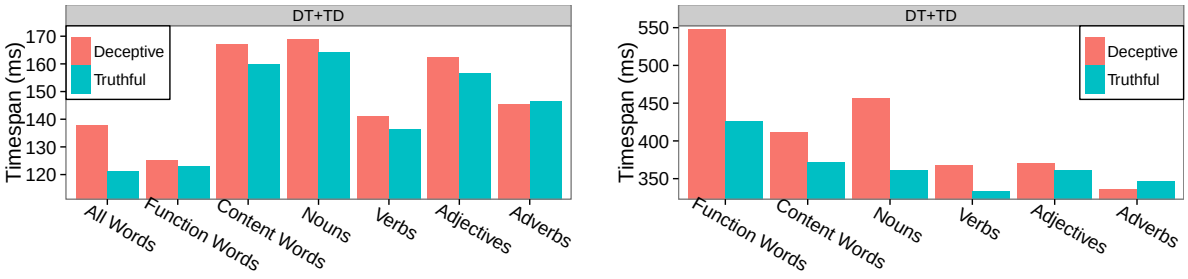
Based on the differences in the temporal aspects of keystrokes, we extract six timespan features to improve this baseline. This set, denoted by $\mathcal{T}_6$, comprises of

(i) $\widehat{\delta}(\text{D})$ = timespan of entire document
(ii) $\widehat{\delta}(k_{prv}+\text{W})$ = average timespan of word plus preceding keystroke
(iii) $\widehat{\delta}(k)$ = average keystroke timespan
(iv) $\widehat{\delta}(\text{SP})$ = average timespan of spaces
(v) $\widehat{\delta}(\neg\text{SP})$ = average timespan of non whitespace keystrokes
(vi) $\widehat{\delta}(\neg\text{W})$ = average interval between words.

The improvements attained by adding $\mathcal{T}_6$ to the baseline are shown in Table 1. Adding the edit patterns ($\mathcal{E}_3$) (cf. § 4.1) further improves the performance (with the exception of two cases) by 0.7–3.5%.

Writing speed, pauses and revisions: To study the temporal aspect of language units across all topics, we first consider all texts, and then restrict to only the first of each 'flow'. The timespan measurements are presented in Fig. 2, showing the average duration of typing documents, sentences, words and individual keystrokes. The timespans are measured as the interval between the first and the last keystrokes. The sentence timespan, for instance, does not include the gap between a sentence end and the first keystroke marking the beginning of the next.

The sentence timespans for "gay marriage" and "gun



(a) (b)

Figure 3: Event timespans in restaurant reviews: (a) language units, and (b) language units including their preceding k^{dn}.

control" are lower in truthful writing, even though the document timespans are higher. This difference implies that the writer is spending a longer period of time to think before commencing the next sentence, but once started, the actual typing proceeds rapidly.

Apart from restaurant reviews, truthful writers have typed slower. This may be due to exercising better care while expressing their honest opinion.

For restaurant reviews, the document, sentence and word timespans are significantly higher in deceptive writing. This, however, is not the case for documents and words in the other two topics. We conjecture that this is because deception is harder to write for product reviews, due to their dependence on factual details. Gun control and gay marriage, on the other hand, are topics well discussed in media, and it is possible that the writers are aware of the arguments that go against their personal belief. The frequency of revisional occurrences (*i.e.,* keys used for editing) shown in Fig. 1, too, supports the thought that writing fake reviews may be harder than adopting a fake stance on well-known issues. Deceptive reviews exhibit a higher number of revisions than truthful ones, but essays show the opposite trend. Our findings align with previous studies (Ott et al., 2011) which showed that deception cues are domain dependent.

Writing speed variations over word categories: Next, we investigate whether there is any quantitative difference in the writing rate over different words with respect to the deceptive and truthful intent of the author. In an attempt to understand this, we analyze words which show the highest timespan difference between deceptive and truthful writings.

Table 2 presents words in the restaurant review topic for which deceptive writers took a lot longer than truthful writers, and vice versa. Some word categories exhibit common trends across all three topics. Highly subjective words, for instance (*e.g.,* "love", "best", "great") are words over which truthful writers spent more time.

Deceptive and truthful texts differ in the typing rate of first- and second-person pronouns. Deceptive reviews reveal more time spent in using 2^{nd}-person pronouns, as shown by "you" and "your". This finding throws some light on how people perceive text cues. Toma and Hancock (2012) showed that readers perform poorly at deception detection because they rely on unrelated text cues such as 2^{nd}-person pronouns. Our analysis indicates that people associate the use of 2^{nd}-person pronouns more with deception not only while reading, but while writing as well.

Deceptive reviews also exhibit longer time spans for 1^{st}-person pronouns (*e.g.,* "we", "me"), which have been known to be useful in deception detection (Newman et al., 2003; Ott et al., 2011). Newman et al. (2003) attributed the less frequent usage of 1^{st}-person pronouns to *psychological distancing*. The longer time taken by deceptive writers in our data is a possible sign of increased cognitive burden when the writer is unable to maintain the psychological distance. Deceptive reviewers also paused a lot more around relative clauses, e.g., "if", "when", and "which".

In essays, however, the difference in timespans of 1^{st}-person and 2^{nd}-person pronouns as well as the timespan difference in relative clauses were insignificant ($< 50\,ms$).

A broader picture of the temporal difference in using different types of words is presented in Fig. 3, which shows deceptive reviewers spending less time on adverbs as compared to truthful writers, but more time on nouns, verbs, adjectives, function words and content words. They also exhibited significantly longer pauses before nouns, verbs and function words.

6 Conclusion

In this paper, we investigated the use of typographic style in deception detection and presented distinct temporal and revisional aspects of keystroke patterns that improve the characterization of deceptive writing. Our study provides novel empirically supported insights into the writing and editing processes of truthful and deceptive writers. It also presents the first application of keylogger data used to distinguish between true and fake texts, and opens up a new range of questions to better understand what affects these different keystroke patterns and what they exhibit. It also suggests new possibilities for making use of keystroke information as an extended linguistic signal to accompany writings.

Acknowledgements

This research is supported in part by gift from Google.

References

Francesco Bergadano, Daniele Gunetti, and Claudia Picardi. 2002. User Authentication through Keystroke Dynamics. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):367–397.

Charles F Bond and Bella M DePaulo. 2006. Accuracy of Deception Judgments. *Personality and Social Psychology Review*, 10(3):214–234.

Sungzoon Cho, Chigeun Han, Dae Hee Han, and Hyung-Il Kim. 2000. Web-based Keystroke Dynamics Identity Verification Using Neural Network. *Journal of Organizationl Computing and Electronic Commerce*, 10(4):295–307.

Bella M DePaulo, James J Lindsay, Brian E Malone, Laura Muhlenbruck, Kelly Charlton, and Harris Cooper. 2003. Cues to Deception. *Psychological Bulletin*, 129(1):74.

Paul Ekman, Maureen O'Sullivan, Wallace V Friesen, and Klaus R Scherer. 1991. Invited Article: Face, Voice and Body in Detecting Deceit. *Journal of Nonverbal Behavior*, 15(2):125–135.

Paul Ekman. 2003. Darwin, Deception, and Facial Expression. *Annals of the New York Academy of Sciences*, 1000(1):205–221.

Clayton Epp, Michael Lippold, and Regan L Mandryk. 2011. Identifying Emotional States Using Keystroke Dynamics. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 715–724. ACM.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Geli Fei, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Exploiting Burstiness in Reviews for Review Spammer Detection. In *ICWSM*, pages 175–184.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic Stylometry for Deception Detection. In *Proc. 50th Annual Meeting of the ACL*, pages 171–175. ACL.

George E Forsen, Mark R Nelson, and Raymond J Staron Jr. 1977. Personal Attributes Authentication Techniques. Technical report, DTIC Document.

Stephanie Gokhman, Jeff Hancock, Poornima Prabhu, Myle Ott, and Claire Cardie. 2012. In Search of a Gold Standard in Studies of Deception. In *Computational Approaches to Deception Detection*, pages 23–30. ACL.

Jeffrey T Hancock, L Curry, Saurabh Goorha, and Michael T Woodworth. 2004. Lies in Conversation: An Examination of Deception Using Automated Linguistic Analysis. In *Annual Conference of the Cognitive Science Society*, volume 26, pages 534–540.

Victoria Johansson. 2009. *Developmental Aspects of Text Production in Writing and Speech*. Ph.D. thesis, Lund University.

Kevin S Killourhy and Roy A Maxion. 2009. Comparing Anomaly-Detection Algorithms for Keystroke Dynamics. In *Dependable Systems & Networks, 2009. DSN'09.*, pages 125–134. IEEE.

John Leggett and Glen Williams. 1988. Verifying Identity Via Keystroke Characteristics. *International Journal of Man-Machine Studies*, 28(1):67–76.

Ann Matsuhashi. 1981. Pausing and Planning: The Tempo of Written Discourse Production. *Research in the Teaching of English*, pages 113–134.

Steven A McCornack. 1997. The Generation of Deceptive Messages: Laying the Groundwork for a Viable Theory of Interpersonal Deception. In John O Greene, editor, *Message Production: Advances in Communication Theory*. Erlbaum, Mahwah, NJ.

Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. 2013. What Yelp Fake Review Filter Might be Doing. In *ICSWM*, pages 409–418.

Matthew L Newman, James W Pennebaker, Diane S Berry, and Jane M Richards. 2003. Lying Words: Predicting Deception from Linguistic Styles. *Personality and Social Psychology Bulletin*, 29(5):665–675.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding Deceptive Opinion Spam by Any Stretch of the Imagination. In *Proc. 49th Annual Meeting of the ACL: HLT*, pages 309–319. ACL.

Catalina L Toma and Jeffrey T Hancock. 2012. What Lies Beneath: The Linguistic Traces of Deception in Online Dating Profiles. *Journal of Communication*, 62(1):78–97.

Paul V Trovillo. 1939. A History of Lie Detection. *Journal of Criminal Law and Criminology (1931-1951)*, 29:848–881.

Janet G van Hell, Ludo Verhoeven, and Liesbeth M van Beijsterveldt. 2008. Pause Time Patterns in Writing Narrative and Expository Texts by Children and Adults. *Discourse Processes*, 45(4-5):406–427.

Lisa M Vizer, Lina Zhou, and Andrew Sears. 2009. Automated Stress Detection Using Keystroke and Linguistic Features: An Exploratory Study. *International Journal of Human-Computer Studies*, 67(10):870–886.

Aldert Vrij, Ronald Fisher, Samantha Mann, and Sharon Leal. 2006. Detecting Deception by Manipulating Cognitive Load. *Trends in Cognitive Sciences*, 10(4):141–142.

Putri Zulkifli. 2013. *Applying Pause Analysis to Explore Cognitive Processes in the Copying of Sentences by Second Language Users*. Ph.D. thesis, University of Sussex.

Leveraging Effective Query Modeling Techniques for Speech Recognition and Summarization

Kuan-Yu Chen[*†], Shih-Hung Liu[*], Berlin Chen[#], Ea-Ee Jan[+], Hsin-Min Wang[*], Wen-Lian Hsu[*], and Hsin-Hsi Chen[†]

[*]Institute of Information Science, Academia Sinica, Taiwan
[†]National Taiwan University, Taiwan
[#]National Taiwan Normal University, Taiwan
[+]IBM Thomas J. Watson Research Center, USA

{kychen, journey, whm, hsu}@iis.sinica.edu.tw,
berlin@ntnu.edu.tw, hhchen@csie.ntu.edu.tw, ejan@us.ibm.com

Abstract

Statistical language modeling (LM) that purports to quantify the acceptability of a given piece of text has long been an interesting yet challenging research area. In particular, language modeling for information retrieval (IR) has enjoyed remarkable empirical success; one emerging stream of the LM approach for IR is to employ the pseudo-relevance feedback process to enhance the representation of an input query so as to improve retrieval effectiveness. This paper presents a continuation of such a general line of research and the main contribution is threefold. First, we propose a principled framework which can unify the relationships among several widely-used query modeling formulations. Second, on top of the successfully developed framework, we propose an extended query modeling formulation by incorporating critical query-specific information cues to guide the model estimation. Third, we further adopt and formalize such a framework to the speech recognition and summarization tasks. A series of empirical experiments reveal the feasibility of such an LM framework and the performance merits of the deduced models on these two tasks.

1 Introduction

Along with the rapidly growing popularity of the Internet and the ubiquity of social web communications, tremendous volumes of multimedia contents, such as broadcast radio and television programs, digital libraries and so on, are made available to the public. Research on multimedia content understanding and organization has witnessed a booming interest over the past decade. By virtue of the developed techniques, a variety of functionalities were created to help distill important content from multimedia collections, or provide locations of important speech segments in a video accompanied with their corresponding transcripts, for users to listen to or to digest. Statistical language modeling (LM) (Jelinek, 1999; Jurafsky and Martin, 2008; Zhai, 2008), which manages to quantify the acceptability of a given word sequence in a natural language or capture the statistical characteristics of a given piece of text, has been proved to offer both efficient and effective modeling abilities in many practical applications of natural language processing and speech recognition (Ponte and Croft, 1998; Jelinek, 1999; Huang, *et al.*, 2001; Zhai and Lafferty, 2001[a]; Jurafsky and Martin, 2008; Furui *et al.*, 2012; Liu and Hakkani-Tur, 2011).

The LM approach was first introduced for the information retrieval (IR) problems in the late 1990s, indicating very good potential, and was subsequently extended in a wide array of follow-up studies. One typical realization of the LM approach for IR is to access the degree of relevance between a query and a document by computing the likelihood of the query generated by the document (usually referred to as the query-likelihood approach) (Zhai, 2008; Baeza-Yates and Ribeiro-Neto, 2011). A document is deemed to be relevant to a given query if the corresponding document model is more likely to generate the query. On the other hand, the Kullback-Leibler divergence measure (denoted by KLM for short hereafter), which quantifies the degree of relevance between a document and a query from a more rigorous information-theoretic perspective, has been proposed (Lafferty and Zhai, 2001; Zhai and Lafferty, 2001[b]; Baeza-Yates and Ribeiro-Neto, 2011). KLM not only can be thought as a natural generalization of the query-likelihood approach, but also has the additional merit of being able to accommodate extra information cues to improve the performance of document ranking. For example, a main challenge facing such a measure is that since a given query usually consists of few words, the true information need is hard to be inferred from the surface statistics of a query. As such, one emerging stream of thought for KLM is to employ the

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1474–1480,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

pseudo-relevance feedback process to construct an enhanced query model (or representation) so as to achieve better retrieval effectiveness (Hiemstra *et al.*, 2004; Lv and Zhai, 2009; Carpineto and Romano, 2012; Lee and Croft, 2013).

Following this line of research, the major contribution of this paper is three-fold: 1) we analyze several widely-used query models and then propose a principled framework to unify the relationships among them; 2) on top of the successfully developed query models, we propose an extended modeling formulation by incorporating additional query-specific information cues to guide the model estimation; 3) we explore a novel use of these query models by adapting them to the speech recognition and summarization tasks. As we will see, a series of experiments indeed demonstrate the effectiveness of the proposed models on these two tasks.

2 Language Modeling Framework

2.1 Kullback-Leibler Divergence Measure

A promising realization of the LM approach to IR is the Kullback-Leibler divergence measure (KLM), which determines the degree of relevance between a document and a query from a rigorous information-theoretic perspective. Two different language models are involved in KLM: one for the document and the other for the query. The divergence of the document model with respect to the query model is defined by

$$\mathrm{KL}(Q \| D) = \Sigma_{w \in V} P(w|Q) \log \frac{P(w|Q)}{P(w|D)}. \quad (1)$$

KLM not only can be thought as a natural generalization of the traditional query-likelihood approach (Yi and Allan, 2009; Baeza-Yates and Ribeiro-Neto, 2011), but also has the additional merit of being able to accommodate extra information cues to improve the estimation of its component models in a systematic way for better document ranking (Zhai, 2008).

Due to that a query usually consists of only a few words, the true query model $P(w|Q)$ might not be accurately estimated by the simple ML estimator (Jelinek, 1991). There are several studies devoted to estimating a more accurate query modeling, saying that it can be approached with the pseudo-relevance feedback process (Lavrenko and Croft, 2001; Zhai and Lafferty, 2001[b]). However, the success depends largely on the assumption that the set of top-ranked documents, $\mathbf{D}_{Top}=\{D_1,D_2,...,D_r,...\}$, obtained from an initial round of retrieval, are relevant and can be used to estimate a more accurate query language model.

2.2 Relevance Modeling

Under the notion of relevance modeling (RM, often referred to as RM-1), each query Q is assumed to be associated with an unknown relevance class R_Q, and documents that are relevant to the semantic content expressed in query are samples drawn from the relevance class R_Q. Since there is no prior knowledge about R_Q, we may use the top-ranked documents $\mathbf{D}_{Top}$ to approximate the relevance class R_Q. The corresponding relevance model can be estimated using the following equation (Lavrenko and Croft, 2001; Lavrenko, 2004):

$$P_{\mathrm{RM}}(w|Q) = \frac{\Sigma_{D_r \in \mathbf{D}_{Top}} P(D_r) P(w|D_r) \Pi_{w' \in Q} P(w'|D_r)}{\Sigma_{D_r'' \in \mathbf{D}_{Top}} P(D_r'') \Pi_{w' \in Q} P(w'|D_r'')}. \quad (2)$$

2.3 Simple Mixture Model

Another perspective of estimating an accurate query model with the top-ranked documents is the simple mixture model (SMM), which assumes that words in $\mathbf{D}_{Top}$ are drawn from a two-component mixture model: 1) One component is the query-specific topic model $P_{\mathrm{SMM}}(w|Q)$**,** and 2) the other is a generic background model $P(w|BG)$. By doing so, the SMM model $P_{\mathrm{SMM}}(w|Q)$ can be estimated by maximizing the likelihood over all the top-ranked documents (Zhai and Lafferty, 2001[b]; Tao and Zhai, 2006):

$$L = \prod_{D_r \in \mathbf{D}_{Top}} \prod_{w \in V} \left(\alpha \cdot P_{\mathrm{SMM}}(w|Q) + (1-\alpha) \cdot P(w|BG)\right)^{c(w,D_r)}, \quad (3)$$

where α is a pre-defined weighting parameter used to control the degree of reliance between $P_{\mathrm{SMM}}(w|Q)$ and $P(w|BG)$. This estimation will enable more specific words to receive more probability mass, thereby leading to a more discriminative query model $P_{\mathrm{SMM}}(w|Q)$.

Although the SMM modeling aims to extract extra word usage cues for enhanced query modeling, it may confront two intrinsic problems. One is the extraction of word usage cues from $\mathbf{D}_{Top}$ is not guided by the original query. The other is that the mixing coefficient α is fixed across all top-ranked documents albeit that different documents would potentially contribute different amounts of word usage cues to the enhanced query model. To mitigate these two problems, the regularized simple mixture model has been proposed and can be estimated by maximizing the likelihood function (Tao and Zhai, 2006; Dillon and Collins-Thompson, 2010)

$$L = \prod_{w \in V} P_{\mathrm{RSMM}}(w|Q)^{\mu \cdot P(w|Q)} \times \prod_{D_r \in \mathbf{D}_{Top}} \prod_{w \in V} \left(\alpha_{D_r} \cdot P_{\mathrm{RSMM}}(w|Q) + (1-\alpha_{D_r}) \cdot P(w|BG)\right)^{c(w,D_r)}, \quad (4)$$

where μ is a weighting factor indicating the confidence on the prior information.

3 The Proposed Modeling Framework

3.1 Fundamentals

It is obvious that the major difference among the

representative query models mentioned above is how to capitalize on the set of top-ranked documents and the original query. Several subtle relationships can be deduced through the following in-depth analysis. First, a direct inspiration of the LM-based query reformulation framework can be drawn from the celebrated Rocchio's formulation, while the former can be viewed as a probabilistic counterpart of the latter (Robertson, 1990; Ponte and Croft, 1998; Baeza-Yates and Ribeiro-Neto, 2011). Second, after some mathematical manipulation, the formulation of the RM model (*c.f.* Eq. (2)) can be rewritten as

$$P_{\mathrm{RM}}(w|Q) = \sum_{D_r \in \mathbf{D}_{Top}} P(w|D_r) \frac{P(Q|D_r)P(D_r)}{\sum_{D_r'' \in \mathbf{D}_{Top}} P(Q|D_r'')P(D_r'')}. \quad (5)$$

It becomes evident that the RM model is composed by mixing a set of document models $P(w|D_r)$. As such, the RM model bears a close resemblance to the Rocchio's formulation. Furthermore, based on Eq. (5), we can recast the estimation of the RM model as an optimization problem, and the likelihood (or objective) function is formulated as

$$L = \prod_{w \in V} \left(\sum_{D_r \in \mathbf{D}_{Top}} P(w|D_r)P(D_r|Q) \right)^{c(w,Q)}, \quad (6)$$
$$s.t. \sum_{D_r \in \mathbf{D}_{Top}} P(D_r|Q) = 1$$

where the document models $P(w|D_r)$ are known in advance; the conditional probability $P(D_r|Q)$ of each document D_r is unknown and leave to be estimated. Finally, a principled framework can be obtained to unify all of these query models, including RM (*c.f.* Eq. (6)), SMM (c.f. Eq. (3)) and RSMM (*c.f.* Eq. (4))), by using a generalized objective likelihood function:

$$L = \prod_{w \in V} \prod_{E_i \in \mathbf{E}} \left(\sum_{M_r \in \mathbf{M}} P(w|M_r)P(M_r) \right)^{c(w,E_i)}, \quad (7)$$
$$s.t. \sum_{M_r \in \mathbf{M}} P(M_r) = 1$$

where **E** represents a set of observations which we want to maximize their likelihood, and **M** denotes a set of mixture components.

3.2 Query-specific Mixture Modeling

The SMM model and the RSMM model are intended to extract useful word usage cues from $\mathbf{D}_{Top}$, which are not only relevant to the original query Q but also external to those already captured by the generic background model. However, we argue in this paper that the "generic information" should be carefully crafted for each query due mainly to the fact that users' information needs may be very diverse from one another. To crystallize the idea, a query-specific background model $P_Q(w|\mathrm{BG})$ for each query Q can be derived from $\mathbf{D}_{Top}$ directly. Another consideration is that since the original query model $P(w|Q)$ cannot be accurately estimated, it thus may not necessarily be the best choice for use in defining a conjugate Dirichlet prior for the enhanced query model to be estimated. We propose to use the RM model as a prior to guide the estimation of the enhanced query model. The enhanced query model is termed query-specific mixture model (QMM), and its corresponding training objective function can be expressed as

$$L = \prod_{w \in V} P_{\mathrm{QMM}}(w|Q)^{\mu \cdot P_{\mathrm{RM}}(w|Q)} \times \prod_{D_r \in \mathbf{D}_{Top}} \prod_{w \in V} \left(\alpha_{D_r} \cdot P_{\mathrm{QMM}}(w|Q) + (1-\alpha_{D_r}) \cdot P_Q(w|BG) \right)^{c(w,D_r)}. \quad (8)$$

4 Applications

4.1 Speech Recognition

Language modeling is a critical and integral component in any large vocabulary continuous speech recognition (LVCSR) system (Huang *et al.*, 2001; Jurafsky and Martin, 2008; Furui *et al.*, 2012). More concretely, the role of language modeling in LVCSR can be interpreted as calculating the conditional probability $P(w|H)$, in which H is a search history, usually expressed as a sequence of words $H=h_1, h_2,\ldots, h_L$, and w is one of its possible immediately succeeding words. Once the various aforementioned query modeling methods are applied to speech recognition, for a search history H, we can conceptually regard it as a query and each of its immediately succeeding words w as a (single-word) document. Then, we may leverage an IR procedure that takes H as a query and poses it to a retrieval system to obtain a set of top-ranked documents from a contemporaneous (or in-domain) corpus. Finally, the enhanced query model (that is $P(w|H)$ in speech recognition) can be estimated by RM, SMM, RSMM or QMM, and further combined with the background n-gram (e.g., trigram) language model to form an adaptive language model to guide the speech recognition process.

4.2 Speech Summarization

On the other hand, extractive speech summarization aims at producing a concise summary by selecting salient sentences or paragraphs from the original spoken document according to a predefined target summarization ratio (Carbonell and Goldstein, 1998; Mani and Maybury, 1999; Nenkova and McKeown, 2011; Liu and Hakkani-Tur, 2011). Intuitively, this task could be framed as an ad-hoc IR problem, where the spoken document is treated as an information need and each sentence of the document is regarded as a candidate information unit to be retrieved according to its relevance to the information need. Therefore, KLM can be used to quantify how close the document D and one of its sentences S are: the closer the sentence model $P(w|S)$ to the document model $P(w|D)$, the more

likely the sentence would be part of the summary. Due to that each sentence S of a spoken document D to be summarized usually consists of only a few words, the corresponding sentence model $P(w|S)$ might not be appropriately estimated by the ML estimation. To alleviate the deficiency, we can leverage the merit of the above query modeling techniques to estimate an accurate sentence model for each sentence to enhance the summarization performance.

5 Experimental Setup

The speech corpus consists of about 196 hours of Mandarin broadcast news collected by the Academia Sinica and the Public Television Service Foundation of Taiwan between November 2001 and April 2003 (Wang *et al.*, 2005), which is publicly available and has been segmented into separate stories and transcribed manually. Each story contains the speech of one studio anchor, as well as several field reporters and interviewees. A subset of 25-hour speech data compiled during November 2001 to December 2002 was used to bootstrap the acoustic model training. The vocabulary size is about 72 thousand words. The background language model was estimated from a background text corpus consisting of 170 million Chinese characters collected from the Chinese Gigaword Corpus released by LDC.

The dataset for use in the speech recognition experiments is compiled by a subset of 3-hour speech data from the corpus within 2003 (1.5 hours for development and 1.5 hours for test). The contemporaneous (in-domain) text corpus used for training the various LM adaptation methods was collected between 2001 and 2003 from the corpus (excluding the test set), which consists of one million Chinese characters of the orthographic broadcast news transcripts. In this paper, all the LM adaptation experiments were performed in word graph rescoring. The associated word graphs of the speech data were built beforehand with a typical LVCSR system (Ortmanns *et al.*, 1997; Young *et al.*, 2006).

In addition, the summarization task also employs the same broadcast news corpus as well. A subset of 205 broadcast news documents compiled between November 2001 and August 2002 was reserved for the summarization experiments (185 for development and 20 for test). A subset of about 100,000 text news documents, compiled during the same period as the documents to be summarized, was employed to estimate the related summarization models compared in this paper. We adopted three variants of the widely-used ROUGE metric (i.e., ROUGE-1, ROGUE-2 and ROUGE-L) for the assessment of summarization performance (Lin, 2003). The summarization ratio, defined as the ratio of the number of words in the automatic (or manual) summary to that in the reference transcript of a spoken document, was set to 10% in this research.

6 Experimental Results

In the first part of experiments, we evaluate the effectiveness of the various query models applied to the speech recognition task. The corresponding results with respect to different numbers of top-ranked documents being used for estimating their component models are shown in Table 1. Also worth mentioning is that the baseline system with the background trigram language model, which was trained with the SRILM toolkit (Stolcke, 2005) and Good-Turing smoothing (Jelinek, 1999), results in a Chinese character error rate (CER) of 20.08% on the test set. Consulting Table 1 we notice two particularities. One is that there is more fluctuation in the CER results of SMM than in those of RM. The reason might be that, for SMM, the extraction of relevance information from the top-ranked documents is conducted with no involvement of the test utterance (i.e., the query; or its corresponding search histories), as elaborated earlier in Section 2. When too many feedback documents are being used, there would be a concern for SMM to be distracted from being able to appropriate model the test utterance, which is probably caused by some dominant distracting (or irrelevant) feedback documents. The other interesting observation is that RSMM only achieves a comparable (even worse) result when compared to SMM. A possible reason is that the prior constraint of the RSMM may contain too much noisy information so as to bias the model estimation. Furthermore, it is evident that the proposed QMM is the best-performing method among all the query models compared in the paper. Although the improvements made by QMM are not as pronounced as expected, we believe that QMM has demonstrated its potential to be applied to other related applications. On the other hand, we compare the various query models with two well-practiced language models, namely the cache model (Cache) (Kuhn and Mori, 1990; Jelinek *et al.*, 1991) and the latent Dirichlet allocation (LDA) (Liu and Liu, 2007; Tam and Schultz, 2005). The CER results of these two models are also shown in Table 1, respectively. For the cache model, bigram cache was used since it can yield better results than the unigram and trigram cache models in our experiments. It is worthy to notice that the LDA model was trained with the entire set of contemporaneous text document collection (*c.f.* Section 4), while all of the query models explored in the paper were estimated based on a subset of the corpus selected by an initial round of retrieval. The results reveal that most of these query models can achieve superior performance over the two conventional language models.

In the second part of experiments, we evaluate the utilities of the various query models as applied to the speech summarization task. At the outset, we assess the performance level of the baseline KLM method by comparison with two well-practiced unsupervised methods, viz. the vector space model (VSM) (Gong and Liu, 2001), and its extension, maximal marginal relevance (MMR) (Carbonell and Goldstein, 1998). The corresponding results are shown in Table 2 and can be aligned with several related literature reviews. By looking at the results, we find that KLM outperforms VSM by a large margin, confirming the applicability of the language modeling framework for speech summarization. Furthermore, MMR that presents an extension of VSM performs on par with KLM for the text summarization task (TD) and exhibits superior performance over KLM for the speech summarization task (SD). We now turn to evaluate the effectiveness of the various query models (viz. RM, SMM, RSMM and QMM) in conjunction with the pseudo-relevance feedback process for enhancing the sentence model involved in the KLM method. The corresponding results are also shown in Table 2. Two noteworthy observations can be drawn from Table 2. One is that all these query models can considerably improve the summarization performance of the KLM method, which corroborates the advantage of using them for enhanced sentence representations. The other is that QMM is the best-performing one among all the formulations studied in this paper for both the TD and SD cases.

Going one step further, we explore to use extra prosodic features that are deemed complementary to the LM cue provided by QMM for speech summarization. To this end, a support vector machine (SVM) based summarization model is trained to integrate a set of 28 commonly-used prosodic features (Liu and Hakkani-Tur, 2011) for representing each spoken sentence, since SVM is arguably one of the state-of-the-art supervised methods that can make use of a diversity of indicative features for text or speech summarization (Xie and Liu, 2010; Chen *et al.*, 2013). The sentence ranking scores derived by QMM and SVM are in turn integrated through a simple log-linear combination. The corresponding results are shown in Table 2, demonstrating consistent improvements with respect to all the three variants of the ROUGE metric as compared to that using either QMM or SVM in isolation. We also investigate using SVM to additionally integrate a richer set of lexical and relevance features to complement QMM and further enhance the summarization effectiveness. However, due to space limitation, we omit the details here. As a side note, there is a sizable gap between the TD and SD cases, indicating room for further improvements. We may seek remedies, such as robust indexing schemes, to compensate for imperfect speech recognition.

Table 1. The speech recognition results (in CER (%)) achieved by various language models along with different numbers of latent topics/pseudo-relevance feedback documents.

	16	32	64	128
Baseline	20.08			
Cache	19.86			
LDA	19.29	19.30	19.28	19.15
RM	19.26	19.26	19.26	19.26
SMM	19.19	19.00	19.14	19.10
RSMM	19.18	19.14	19.15	19.19
QMM	**19.05**	**18.97**	**19.00**	**18.99**

Table 2. The summarization results (in F-scores) achieved by various language models along with text and spoken documents.

	Text Documents (TD)			Spoken Documents (SD)		
	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-1	ROUGE-2	ROUGE-L
VSM	0.347	0.228	0.290	0.342	0.189	0.287
MMR	0.407	0.294	0.358	0.381	0.226	0.331
KLM	0.411	0.298	0.361	0.364	0.210	0.307
RM	0.453	0.335	0.403	0.382	0.239	0.331
SMM	0.439	0.320	0.388	0.383	0.229	0.327
RSMM	0.472	0.365	0.423	0.381	0.235	0.329
QMM	**0.486**	**0.382**	**0.435**	**0.395**	**0.256**	**0.349**
SVM	0.441	0.334	0.396	0.370	0.222	0.326
QMM+ SVM	**0.492**	**0.395**	**0.448**	**0.398**	**0.261**	**0.358**

7 Conclusion and Outlook

In this paper, we have presented a systematic and thorough analysis of a few well-practiced query models for IR and extended their novel applicability to speech recognition and summarization in a principled way. Furthermore, we have proposed an extension of this research line by introducing query-specific mixture modeling; the utilities of the deduced model have been extensively compared with several existing query models. As to future work, we would like to investigate jointly integrating proximity and other different kinds of relevance and lexical/semantic information cues into the process of feedback document selection so as to improve the empirical effectiveness of such query modeling.

Acknowledgements

This research is supported in part by the "Aim for the Top University Project" of National Taiwan Normal University (NTNU), sponsored by the Ministry of Education, Taiwan, and by the Ministry of Science and Technology, Taiwan, under Grants MOST 103-2221-E-003-016-MY2, NSC 101-2221-E-003-024-MY3, NSC 102-2221-E-003-014-, NSC 101-2511-S-003-057-MY3, NSC 101-2511-S-003-047-MY3 and NSC 103-2911-I-003-301.

References

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 2011. Modern information retrieval: the concepts and technology behind search, ACM Press.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, pp.993–1022.

David M. Blei and John Lafferty. 2009. Topic models. In A. Srivastava and M. Sahami, (eds.), *Text Mining: Theory and Applications*. Taylor and Francis.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversitybased reranking for reordering documents and producing summaries. In *Proc. SIGIR*, pp. 335–336.

Claudio Carpineto and Giovanni Romano. 2012. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys*, vol. 44, pp.1–56.

Stephane Clinchant and Eric Gaussier. 2013. A theoretical analysis of pseudo-relevance feedback models. In *Proc. ICTIR.*

Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. 2008. Selecting good expansion terms for pseudo-relevance feedback. In *Proc. SIGIR*, pp. 243–250.

Berlin Chen, Shih-Hsiang Lin, Yu-Mei Chang, and Jia-Wen Liu. 2013. Extractive speech summarization using evaluation metric-related training criteria. *Information Processing & Managemen*t, 49(1), pp. 1cess

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39(1), pp. 1–38.

Joshua V. Dillon and Kevyn Collins-Thompson. 2010. A unified optimization framework for robust pseudo-relevance feedback algorithms. In *Proc. CIKM*, pp. 1069–1078.

Sadaoki Furui, Li Deng, Mark Gales, Hermann Ney, and Keiichi Tokuda. 2012. Fundamental technologies in modern speech recognition. *IEEE Signal Processing Magazine*, 29(6), pp. 16–17.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proc. SIGIR*, pp. 19–25.

Djoerd Hiemstra, Stephen Robertson, and Hugo Zaragoza. 2004. Parsimonious language models for information retrieval. In *Proc. SIGIR*, pp. 178–185.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proc. SIGIR*, pp. 50–57.

Thomas Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42, pp. 177–196.

Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. 2001. Spoken language processing: a guide to theory, algorithm, and system development. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Frederick Jelinek, Bernard Merialdo, Salim Roukos, and M. Strauss. 1991. A dynamic language model for speech recognition. In *Proc. the DARPA workshop on speech and natural language*, pp. 293–295.

Frederick Jelinek. 1999. Statistical methods for speech recognition. MIT Press.

Daniel Jurafsky and James H. Martin. 2008. Speech and language processing. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Roland Kuhn and Renato D. Mori. 1990. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6), pp. 570–583.

Solomon Kullback and Richard A. Leibler. 1951. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1), pp. 79–86.

Chin-Yew Lin. 2003. ROUGE: Recall-oriented Understudy for Gisting Evaluation. Available: http://haydn.isi.edu/ROUGE/.

Feifan Liu and Yang Liu. 2007. Unsupervised language model adaptation incorporating named entity information. In *Proc. ACL*, pp. 672–769.

Yang Liu and Dilek Hakkani-Tur. 2011. Speech summarization. *Chapter 13* in *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, G. Tur and R. D. Mori (Eds), New York: Wiley.

John Lafferty and Chengxiang Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proc. SIGIR*, pp. 111–119.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance-based language models. In *Proc. SIGIR*, pp. 120–127.

Victor Lavrenko. 2004. A Generative Theory of Relevance. PhD thesis, University of Massachusetts, Amherst.

Shasha Xie and Yang Liu. 2010. Improving supervised learning for meeting summarization using sampling and regression. *Computer Speech & Language*, 24(3), pp. 495–514.

Yuanhua Lv and Chengxiang Zhai. 2009. A comparative study of methods for estimating query language models with pseudo feedback. In *Proc. CIKM*, pp. 1895–1898.

Yuanhua Lv and Chengxiang Zhai. 2010. Positional relevance model for pseudo-relevance feedback. In *Proc. SIGIR*, pp. 579–586.

Kyung Soon Lee, W. Bruce Croft, and James Allan. 2008. A cluster-based resampling method for pseudo-relevance feedback. In *Proc. SIGIR*, pp. 235–242.

Kyung Soon Lee and W. Bruce Croft. 2013. A deterministic resampling method using overlapping document clusters for pseudo-relevance feedback. *Inf. Process. Manage. 49(4)*, pp. 792–806.

Inderjeet Mani and Mark T. Maybury (Eds.). 1999. Advances in automatic text summarization. Cambridge, MA: MIT Press.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2–3), pp. 103–233.

Stefan Ortmanns, Hermann Ney, and Xavier Aubert. 1997. A word graph algorithm for large vocabulary continuous speech recognition. *Computer Speech and Language*, pp. 43–72.

Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proc. SIGIR*, pp. 275–281.

Stephen E. Robertson. 1990. On term selection for query expansion. *Journal of Documentation*, 46(4), pp. 359–364.

Andreas Stolcke. 2005. SRILM - An extensible language modeling toolkit. In *Proc. INTERSPEECH*, pp.901–904.

Tao Tao and Chengxiang Zhai. 2006. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *Proc. SIGIR*, pp. 162–169.

Yik-Cheung Tam and Tanja Schultz. 2005. Dynamic language model adaptation using variational Bayes inference. In *Proc. INTERSPEECH*, pp. 5–8.

Xuanhui Wang, Hui Fang, and Chengxiang Zhai. 2008. A study of methods for negative relevance feedback. In *Proc. SIGIR*, pp. 219–226.

Hsin-Min Wang, Berlin Chen, Jen-Wei Kuo, and Shih-Sian Cheng. 2005. MATBN: A Mandarin Chinese broadcast news corpus. *International Journal of Computational Linguistics & Chinese Language Processing*, 10(2), pp. 219–236.

Xing Yi and James Allan. 2009. A comparative study of utilizing topic models for information retrieval. In *Proc. ECIR*, pp. 29–41.

Steve Young, Dan Kershaw, Julian Odell, Dave Ollason, Valtcho Valtchev, and Phil Woodland. 2006. The HTK book version 3.4. Cambridge University Press.

Chengxiang Zhai and John Lafferty. 2001[a]. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proc. SIGIR*, pp. 334–342.

Chengxiang Zhai and John Lafferty. 2001[b]. Model-based feedback in the language modeling approach to information retrieval. In *Proc. CIKM*, pp. 403–410.

Chengxiang Zhai. 2008. Statistical language models for information retrieval: a critical review. Foundations and Trends in Information Retrieval, 2 (3), pp. 137–213.

Yi Zhang, Jamie Callan, and Thomas Minka. 2002. Novelty and redundancy detection in adaptive filtering. In *Proc. SIGIR*, pp. 81–88.

Staying on Topic: An Indicator of Power in Political Debates

Vinodkumar Prabhakaran
Dept. of Computer Science
Columbia University
New York, NY, USA
vinod@cs.columbia.edu

Ashima Arora
Dept. of Computer Science
Columbia University
New York, NY, USA
aa3470@columbia.edu

Owen Rambow
CCLS
Columbia University
New York, NY, USA
rambow@ccls.columbia.edu

Abstract

We study the topic dynamics of interactions in political debates using the 2012 Republican presidential primary debates as data. We show that the tendency of candidates to shift topics changes over the course of the election campaign, and that it is correlated with their relative power. We also show that our topic shift features help predict candidates' relative rankings.

1 Introduction

The field of computational social sciences has created many interesting applications for natural language processing in recent years. One of the areas where NLP techniques have shown great promise is in the analysis of political speech. For example, researchers have applied NLP techniques to political texts for a variety of tasks such as predicting voting patterns (Thomas et al., 2006), identifying markers of persuasion (Guerini et al., 2008), capturing cues that signal charisma (Rosenberg and Hirschberg, 2009), and detecting ideological positions (Sim et al., 2013). Our work also analyzes political speech, more specifically, presidential debates. The contribution of this paper is to show that the topic shifting tendency of a presidential candidate changes over the course of the election campaign, and that it is correlated with his or her relative power. We also show that this insight can help computational systems that predict the candidates' relative rankings based on their interactions in the debates.

2 Motivation

The motivation for this paper stems from prior work done by the first author in collaboration with other researchers (Prabhakaran et al., 2013a; Prabhakaran et al., 2013b). Prabhakaran et al. (2013a) introduced the notion of power in the domain of presidential debates, and Prabhakaran et al. (2013b) followed it up with an automatic power ranker system based on interactions within the debates. The power that a candidate had at a certain point in the election campaign was modeled based on his or her recent poll standings: in elections, popularity is power. Those studies analyzed the 2012 Republican presidential primary debates and found that a candidate's power at the time of a debate correlates with the structure of interactions within the debate (e.g., turn frequency and interruption patterns). Another finding was that the candidates' power correlates with the distribution of topics they speak about in the debates: candidates with more power spoke significantly more about certain topics (e.g., economy) and less about certain other topics (e.g., energy). However, these findings relate to the specific election cycle that was analyzed and will not carry over to political debates in general.

A further dimension with relevance beyond a specific election campaign is how topics evolve during the course of an interaction (e.g., who attempts to shift topics). In (Prabhakaran et al., 2014), we explored this dimension and found that candidates with higher power introduce significantly more topics in the debates, but attempt to shift topics significantly less often while responding to a moderator. We used the basic LDA topic modeling method (with a filter for substantivity of turns) to assign topics to turns, which were then used to detect shifts in topics. However, segmenting interactions into coherent topic segments is an active area of research and a variety of topic modeling approaches have been proposed for that purpose. In this paper, we explore the utility of one such topic modeling approach to tackle this problem.

While most of the early approaches for topic segmenting in interactions have focused on the

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1481–1486,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

content of the contribution, Nguyen et al. (2012) introduced a system called Speaker Identity for Topic Segmentation (SITS) which also takes into account the topic shifting tendencies of the participants of the conversation. In later work, Nguyen et al. (2013) demonstrated the SITS system's utility in detecting influencers in Crossfire debates and Wikipedia discussions. They also applied the SITS system to the domain of political debates. However they were able to perform only a qualitative analysis of its utility in the debates domain since the debates data did not have influence annotations. In this paper, we use the SITS system to assign topics to turns and perform a quantitative analysis of how the topic shift features calculated using the SITS system relate to the notion of power as captured by (Prabhakaran et al., 2013a).

The SITS system associates each debate participant with a constant scalar value that captures his or her tendency to shift topics. However, since we want to investigate how each candidate's topic shifting tendency relates to his or her changing power over the course of the campaign, we introduce a variation of the SITS analysis in which we represent a different "persona" for each candidate in each debate. Once equipped with this notion of "persona", we find that the topic shifting tendency of a candidate does indeed show a great deal of fluctuation during the election campaign period. We also find that this fluctuation in topic shifting tendencies is significantly correlated with the candidates' power.

As an additional contribution of this paper, we demonstrate the utility of our topic shift features extracted using both types of SITS-based analyses in improving the performance of the automatic power ranker system presented in (Prabhakaran et al., 2013b). We also investigated the utility of topic shifting features described in (Prabhakaran et al., 2014) extracted using LDA based topic modeling. However, they did not improve the performance of the ranker, and hence we do not discuss them in detail in this paper.

3 Data

We use the presidential debates corpus released by Prabhakaran et al. (2013a), which contains manual transcripts of 20 debates held between May 2011 and February 2012 as part of the 2012 Republican presidential primaries. The corpus also captures each candidate's power at the time of each debate, computed based on their relative standing in recent public polls. The poll numbers capture how successful candidates are in convincing the electorate of their candidature, which in turn affects their confidence within the debates. These debates serve as a rich domain to explore manifestations of power since they are a medium through which candidates pursue and maintain power over other candidates. Prabhakaran et al. (2013b) offers a detailed description of how the relative standings in national and state-level polls from various sources are aggregated to obtain candidates' power.

The transcripts are originally obtained from The American Presidency Project, where each turn of the conversation is manually demarcated and their speakers identified. The turns in the corpus are preprocessed using the Stanford CoreNLP package to perform basic NLP steps such as tokenization, sentence segmentation, parts-of-speech tagging and lemmatization.

4 Modeling Topic Shifts

Topic segmentation, the task of segmenting interactions into coherent topic segments, is an important step in analyzing interactions. In addition to its primary purpose, topic segmentation also identifies the speaker turn where the conversation changed from one topic to another, i.e., where the topic shifted, which may shed light on the characteristics of the speaker who changed the topic. We use the SITS approach proposed by (Nguyen et al., 2012) to detect topic shifts. We also propose a different way of using SITS to obtain an analysis of our corpus, which we call SITS^{var}. We discuss both in turn, and then provide a discussion.

4.1 Segmentation using SITS

Most computational approaches towards automatic topic segmentation have focused mainly on the content of the contribution without taking into account the social aspects or speaker characteristics. Different discourse participants may have different tendencies to introduce or shift topics in interactions. In order to address this shortcoming, Nguyen et al. (2012) proposed a new topic segmentation model called Speaker Identity for Topic Segmentation (SITS), in which they explicitly model the individual's tendency to introduce new topics.

Like traditional topic modeling approaches, the SITS system also considers each turn to be a

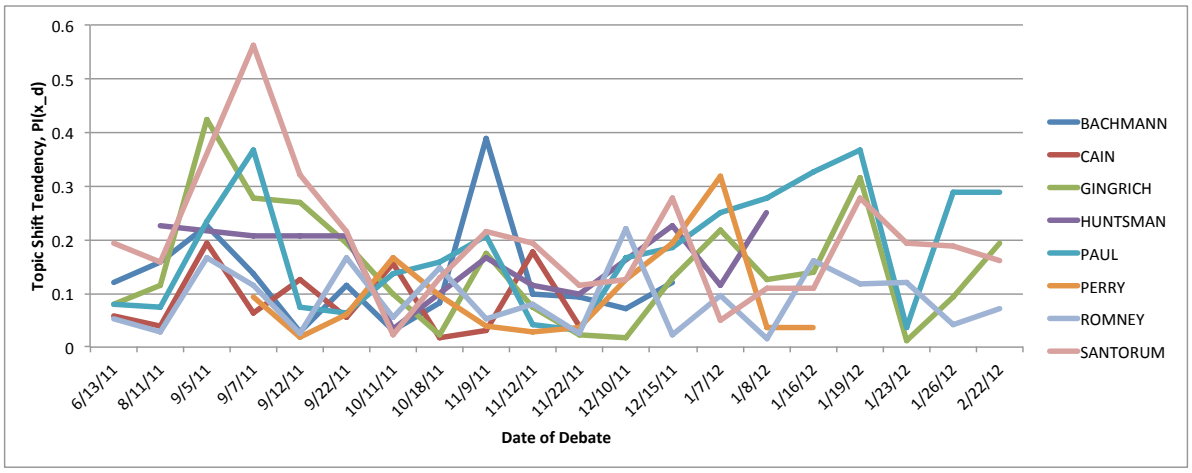


Figure 1: SITS$^{\text{var}}$ Topic shift tendency values across debates

bag of words generated from a mixture of topics. These topics themselves are multinomial distributions over terms. In order to account for the topic shifts that happen during the course of an interaction, they introduce a binary latent variable $l_{d;t}$ called the topic shift to indicate whether the speaker changed the topic or not in conversation d at turn t. To capture the individual speaker's topic shifting tendency, they introduced another latent variable called topic shift tendency (π_x) of speaker x. The π_x value represents the propensity of speaker x to perform a topic shift.

4.2 Segmentation using SITS$^{\text{var}}$

Within the SITS formulation, the topic shifting tendency of an individual (π_x) is considered a constant across conversations. While an individual may have an inherent propensity to shift topics or not, we argue that the topic shifting tendency he or she displays can vary based on the social settings in which he or she interacts and his or her status within those settings. In other words, the same discourse participant may behave differently in different social situations and at different points in time. This is especially relevant in the context of our dataset, where the debates happen over a period of 10 months, and the power and status of each candidate in the election campaign vary greatly within that time period.

We propose a variant of SITS which takes this issue into account. We consider each candidate to have a different "persona" in each debate. To accomplish this, we create new identities for each candidate x for each debate d, denoted by x_d. For example, 'ROMNEY_08-11-2011' denotes the persona of the candidate ROMNEY in the debate held on 08-11-2011. Running the SITS system using this formulation, we obtain different π_{x_d} values for candidate x for different debates, capturing different topic shift tendencies of x.

4.3 Execution

We perform both the SITS and SITS$^{\text{var}}$ analyses on the 20 debates in our corpus. We used the non-parametric version of SITS for both runs, since it systemically estimates the number of topics in the data. We set the maximum number of iterations at 5000, sample lag at 100 and initial number of topics at 25. We refer the reader to (Nguyen et al., 2013) for details on these parameters.

For each candidate, we calculate the mean and standard deviation of the topic shift tendency (π_{x_d}) of his or her personas across all debates he or she participated in. We then average these means and standard deviations, and obtain an average mean of 0.14 and an average standard deviation of 0.09. This shows that the topic shift tendencies of candidates vary by a considerable amount across debates. Figure 1 shows the π_{x_d} value fluctuating across different debates.

5 Analysis of Topic Shift Features

Nguyen et al. (2013) used the SITS analysis as a means to model influence in multi party conversations. They propose two features to detect influencers: Total Topic Shifts (TTS) and Weighted Topic Shifts (WTS). TTS(x, d) captures the expected number of topic shifts the individual x makes in conversation d. This expectation is calculated through the empirical average of samples

Feature Set	Feature	Correlation
TopSh	Total Topic Shifts (TTS)	0.12
	Weighted Topic Shifts (WTS)	0.16
$\text{TopSh}^{\text{var}}$	Total Topic Shifts (TTS^{var})	0.12
	Weighted Topic Shifts (WTS^{var})	0.15
	Topic Shift Tendency (PI^{var})	**-0.27**

Table 1: Pearson Correlations for Topical Features
boldface denotes statistical significance ($p < 0.05$)

from the Gibbs sampler, after a burn-in period. We refer the reader to (Nguyen et al., 2013) for more details on how this value is computed. WTS(x, d) is the value of TTS(x, d) weighted by $1 - \pi_x$. The intuition here is that a topic shift by a speaker with low topic shift tendency must be weighted higher than that by a speaker with a high topic shift tendency. We use these two features as well, and denote the set of these two features as TopSh.

We also extract the TTS and WTS features using our SITS^{var} variation of topic segmentation analysis and denote them as TTS^{var} and WTS^{var} respectively. In addition, we also use a feature $\text{PI}^{\text{var}}(x, d)$ which is the π_{x_d} value obtained by the SITS^{var} for candidate x in debate d. It captures the topic shifting tendency of candidate x in debate d. (We do not include the SITS π_x value in our correlation analysis since it is constant across debates.) We denote the set of these three features obtained from the SITS^{var} run as $\text{TopSh}^{\text{var}}$.

Table 1 shows the Pearson's product correlation between each topical feature and candidate's power. We obtain a highly significant ($p = 0.002$) negative correlation between topic shift tendency of a candidate (PI) and his/her power. In other words, the variation in the topic shifting tendencies is significantly correlated with the candidates' recent poll standings. Candidates who are higher up in the polls tend to stay on topic while the candidates with less power attempt to shift topics more often. This is in line with our previous findings from (Prabhakaran et al., 2014) that candidates with higher power attempt to shift topics less often than others when responding to moderators. It is also in line with the findings by Prabhakaran et al. (2013a) that candidates with higher power tend not to interrupt others. On the other hand, we did not obtain any significant correlation for the features proposed by Nguyen et al. (2013).

6 Topic Shift Features in Power Ranker

In this section, we investigate the utility of the SITS and SITS^{var} based topic shift features described above in the problem of automatically ranking the participants of debates based on their power. Prabhakaran et al. (2013b) define the problem as follows: given a debate d with a set of participants $C_d = \{x_1, x_2, ...x_n\}$ and corresponding power indices $P(x_i)$ for $1 < i < n$, find a ranking function $r : C_d \rightarrow \{1...n\}$ such that for all $1 < i, j < n$, $r(x_i) > r(x_j) \iff P(x_i) > P(x_j)$. For our experiments, we use the SVM^{rank} based supervised learned power ranker presented in that work to estimate this ranking function.

As we do in (Prabhakaran et al., 2013b), we here report Kendall's Tau and Normalized Discounted Cumulative Gain values (NDCG and NDCG@3) on 5-fold cross validation (at the debate level). All three metrics are based on the number of rank inversions between original and predicted ranking. While Tau treats all rank inversions equal, NDCG and NDCG@3 penalize the inversions happening in the top of the ranked list more than those happening in the bottom. NDCG@3 focuses only on the top 3 positions in the ranked list.

We use the best performing feature set of (Prabhakaran et al., 2013b) as the baseline (BL), which contains three features: Words Deviation (WD), Question Deviation (QD) and Mention Percentage (MP). WD and QD capture the deviation of percentage of words spoken by the candidate and questions addressed to the candidate from the expected fair share of those measures in the particular debate. The fair share for debate d is $1/|C_d|$ — the percentage each candidate would have gotten for each feature if it was equally distributed. This deviation measure is used instead of the raw per-

	Kendall's Tau	NDCG	NDCG@3
BL	0.55	0.962	0.932
TopSh	0.36	0.907	0.830
TopSh^{var}	0.39	0.919	0.847
BL + TopSh	0.59	0.967	0.929
BL + TopSh^{var}	**0.60**	**0.970**	**0.937**
BL + TopSh + TopSh^{var}	0.59	0.968	0.934

Table 2: Power Ranker results using topic shift features on 5-fold cross validation
BL: Baseline system (Prabhakaran et al., 2013b)
NDCG: Normalized Discounted Cumulative Gain

centage in order to handle the fact that the percentage values are dependent on the number of participants in a debate, which varied from 9 to 4. MP captures the percentage of mentions of the candidate within a debate.

Table 2 shows the results obtained using the baseline features (BL) as well as combinations of TopSh and TopSh^{var} features. The baseline system obtained a Kendall Tau of 0.55, NDCG of 0.962 and NDCG@3 of 0.932. The topic shift features by themselves performed much worse, with TopSh^{var} posting marginally better results than TopSh. Combining the topic shift and baseline features increases performance considerably. TopSh^{var} obtained better performance than TopSh across the board. BL + TopSh^{var} posted the overall best system obtaining a Tau of 0.60, NDCG of 0.970, and NDCG@3 of 0.937. These results demonstrates the utility of topic shift features in the power ranking problem, especially using the SITS^{var} formulation. We also experimented with all subsets of TopSh and TopSh^{var}; the best results were obtained using all features in each set.

7 Related Work

Studies in sociolinguistics (e.g., (Ng et al., 1993; Ng et al., 1995; Reid and Ng, 2000)) have long established that dialog structure in interactions relates to power and influence. Researchers in the NLP community have studied power and influence in various genres of interactions, such as organizational email threads (Bramsen et al., 2011; Gilbert, 2012; Prabhakaran and Rambow, 2013; Prabhakaran and Rambow, 2014), online discussion forums (Danescu-Niculescu-Mizil et al., 2012; Biran et al., 2012) and online chat dialogs (Strzalkowski et al., 2012). The correlates analyzed in these studies range from word and phrase patterns, to derivatives of such patterns such as linguistic coordination, to deeper dialogic features such as argumentation and dialog acts. Our work differs from these studies in that we study the correlates of power in topic dynamics. Furthermore, we analyze spoken interactions.

8 Conclusion

In this paper, we studied how topic shift patterns in the 2012 Republican presidential debates correlate with the power of candidates. We proposed an alternate formulation of the SITS topic segmentation system that captures fluctuations in each candidate's topic shifting tendencies, which we found to be correlated with their power. We also showed that features based on topic shift improve the prediction of the relative rankings of candidates. In future work, we will explore a model that captures individuals' inherent topic shift propensities, while also capturing their fluctuations due to social factors.

Acknowledgments

This paper is based upon work supported by the DARPA DEFT Program. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. We also thank the anonymous reviewers for their constructive feedback.

References

Or Biran, Sara Rosenthal, Jacob Andreas, Kathleen McKeown, and Owen Rambow. 2012. Detecting influencers in written online conversations. In *Proceedings of the Second Workshop on Language in Social Media*, pages 37–45, Montréal, Canada, June. Association for Computational Linguistics.

Philip Bramsen, Martha Escobar-Molano, Ami Patel, and Rafael Alonso. 2011. Extracting social power relationships from natural language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 773–782, Portland, Oregon, USA, June. Association for Computational Linguistics.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: language effects and power differences in social interaction. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, New York, NY, USA. ACM.

Eric Gilbert. 2012. Phrases that signal workplace hierarchy. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, CSCW '12, pages 1037–1046, New York, NY, USA. ACM.

Marco Guerini, Carlo Strapparava, and Oliviero Stock. 2008. Corps: A corpus of tagged political speeches for persuasive communication processing. *Journal of Information Technology & Politics*, 5(1):19–32.

Sik Hung Ng, Dean Bell, and Mark Brooke. 1993. Gaining turns and achieving high in influence ranking in small conversational groups. *British Journal of Social Psychology*, pages 32, 265–275.

Sik Hung Ng, Mark Brooke, and Michael Dunne. 1995. Interruption and in influence in discussion groups. *Journal of Language and Social Psychology*, pages 14(4),369–381.

Viet-An Nguyen, Jordan Boyd-Graber, and Philip Resnik. 2012. Sits: A hierarchical nonparametric model using speaker identity for topic segmentation in multiparty conversations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 78–87, Jeju Island, Korea, July. Association for Computational Linguistics.

Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, Deborah A. Cai, Jennifer E. Midberry, and Yuanxin Wang. 2013. Modeling topic control to detect influence in conversations using nonparametric topic models. *Machine Learning*, pages 1–41.

Vinodkumar Prabhakaran and Owen Rambow. 2013. Written dialog and social power: Manifestations of different types of power in dialog behavior. In *Proceedings of the IJCNLP*, pages 216–224, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Vinodkumar Prabhakaran and Owen Rambow. 2014. Predicting power relations between participants in written dialog from a single thread. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 339–344, Baltimore, Maryland, June. Association for Computational Linguistics.

Vinodkumar Prabhakaran, Ajita John, and Dorée D. Seligmann. 2013a. Power dynamics in spoken interactions: a case study on 2012 republican primary debates. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 99–100. International World Wide Web Conferences Steering Committee.

Vinodkumar Prabhakaran, Ajita John, and Dorée D. Seligmann. 2013b. Who had the upper hand? ranking participants of interactions based on their relative power. In *Proceedings of the IJCNLP*, pages 365–373, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Vinodkumar Prabhakaran, Ashima Arora, and Owen Rambow. 2014. Power of confidence: How poll scores impact topic dynamics in political debates. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, page 49, Baltimore, MD, USA, June. Association for Computational Linguistics.

Scott A. Reid and Sik Hung Ng. 2000. Conversation as a resource for in influence: evidence for prototypical arguments and social identification processes. *European Journal of Social Psych.*, pages 30, 83–100.

Andrew Rosenberg and Julia Hirschberg. 2009. Charisma perception from text and speech. *Speech Communication*, 51(7):640–655.

Yanchuan Sim, Brice D. L. Acree, Justin H. Gross, and Noah A. Smith. 2013. Measuring ideological proportions in political speeches. In *Proceedings of the 2013 Conference on EMNLP*, pages 91–101, Seattle, Washington, USA, October. Association for Computational Linguistics.

Tomek Strzalkowski, Samira Shaikh, Ting Liu, George Aaron Broadwell, Jenny Stromer-Galley, Sarah Taylor, Umit Boz, Veena Ravishankar, and Xiaoai Ren. 2012. Modeling leadership and influence in multi-party online discourse. In *Proceedings of COLING*, pages 2535–2552, Mumbai, India, December. The COLING 2012 Organizing Committee.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 327–335, Sydney, Australia, July. Association for Computational Linguistics.

Language Modeling with Power Low Rank Ensembles

Ankur P. Parikh
School of Computer Science
Carnegie Mellon University
apparikh@cs.cmu.edu

Avneesh Saluja
Electrical & Computer Engineering
Carnegie Mellon University
avneesh@cs.cmu.edu

Chris Dyer
School of Computer Science
Carnegie Mellon University
cdyer@cs.cmu.edu

Eric P. Xing
School of Computer Science
Carnegie Mellon University
epxing@cs.cmu.edu

Abstract

We present power low rank ensembles (PLRE), a flexible framework for n-gram language modeling where ensembles of low rank matrices and tensors are used to obtain smoothed probability estimates of words in context. Our method can be understood as a generalization of n-gram modeling to non-integer n, and includes standard techniques such as absolute discounting and Kneser-Ney smoothing as special cases. PLRE training is efficient and our approach outperforms state-of-the-art modified Kneser Ney baselines in terms of perplexity on large corpora as well as on BLEU score in a downstream machine translation task.

1 Introduction

Language modeling is the task of estimating the probability of sequences of words in a language and is an important component in, among other applications, automatic speech recognition (Rabiner and Juang, 1993) and machine translation (Koehn, 2010). The predominant approach to language modeling is the n-gram model, wherein the probability of a word sequence $P(w_1, \ldots, w_\ell)$ is decomposed using the chain rule, and then a Markov assumption is made: $P(w_1, \ldots, w_\ell) \approx \prod_{i=1}^{\ell} P(w_i | w_{i-n+1}^{i-1})$. While this assumption substantially reduces the modeling complexity, parameter estimation remains a major challenge. Due to the power-law nature of language (Zipf, 1949), the maximum likelihood estimator massively overestimates the probability of rare events and assigns zero probability to legitimate word sequences that happen not to have been observed in the training data (Manning and Schütze, 1999).

Many smoothing techniques have been proposed to address the estimation challenge. These reassign probability mass (generally from overestimated events) to unseen word sequences, whose probabilities are estimated by interpolating with or backing off to lower order n-gram models (Chen and Goodman, 1999).

Somewhat surprisingly, these widely used smoothing techniques differ substantially from techniques for coping with data sparsity in other domains, such as collaborative filtering (Koren et al., 2009; Su and Khoshgoftaar, 2009) or matrix completion (Candès and Recht, 2009; Cai et al., 2010). In these areas, *low rank* approaches based on matrix factorization play a central role (Lee and Seung, 2001; Salakhutdinov and Mnih, 2008; Mackey et al., 2011). For example, in recommender systems, a key challenge is dealing with the sparsity of ratings from a single user, since typical users will have rated only a few items. By projecting the low rank representation of a user's (sparse) preferences into the original space, an estimate of ratings for new items is obtained. These methods are attractive due to their computational efficiency and mathematical well-foundedness.

In this paper, we introduce **power low rank ensembles** (PLRE), in which low rank tensors are used to produce smoothed estimates for n-gram probabilities. Ideally, we would like the low rank structures to discover semantic and syntactic relatedness among words and n-grams, which are used to produce smoothed estimates for word sequence probabilities. In contrast to the few previous low rank language modeling approaches, PLRE is not orthogonal to n-gram models, but rather a general framework where existing n-gram smoothing methods such as Kneser-Ney smoothing are special cases. A key insight is that PLRE does not compute low rank approximations of the original

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1487–1498,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

joint count matrices (in the case of bigrams) or tensors i.e. multi-way arrays (in the case of 3-grams and above), but instead altered quantities of these counts based on an element-wise power operation, similar to how some smoothing methods modify their lower order distributions.

Moreover, PLRE has two key aspects that lead to easy scalability for large corpora and vocabularies. First, since it utilizes the original n-grams, the ranks required for the low rank matrices and tensors tend to be remain tractable (e.g. around 100 for a vocabulary size $V \approx 1 \times 10^6$) leading to fast training times. This differentiates our approach over other methods that leverage an underlying latent space such as neural networks (Bengio et al., 2003; Mnih and Hinton, 2007; Mikolov et al., 2010) or soft-class models (Saul and Pereira, 1997) where the underlying dimension is required to be quite large to obtain good performance. Moreover, at test time, the probability of a sequence can be queried in time $O(\kappa_{max})$ where κ_{max} is the maximum rank of the low rank matrices/tensors used. While this is larger than Kneser Ney's virtually constant query time, it is substantially faster than conditional exponential family models (Chen and Rosenfeld, 2000; Chen, 2009; Nelakanti et al., 2013) and neural networks which require $O(V)$ for exact computation of the normalization constant. See Section 7 for a more detailed discussion of related work.

Outline: We first review existing n-gram smoothing methods (§2) and then present the intuition behind the key components of our technique: **rank** (§3.1) and **power** (§3.2). We then show how these can be interpolated into an ensemble (§4). In the experimental evaluation on English and Russian corpora (§5), we find that PLRE outperforms Kneser-Ney smoothing and all its variants, as well as class-based language models. We also include a comparison to the log-bilinear neural language model (Mnih and Hinton, 2007) and evaluate performance on a downstream machine translation task (§6) where our method achieves consistent improvements in BLEU.

2 Discount-based Smoothing

We first provide background on absolute discounting (Ney et al., 1994) and Kneser-Ney smoothing (Kneser and Ney, 1995), two common n-gram smoothing methods. Both methods can be formulated as back-off or interpolated models; we describe the latter here since that is the basis of our low rank approach.

2.1 Notation

Let $c(w)$ be the count of word w, and similarly $c(w, w_{i-1})$ for the joint count of words w and w_{i-1}. For shorthand we will define w_i^j to denote the word sequence $\{w_i, w_{i+1}, ..., w_{j-1}, w_j\}$. Let $\widehat{P}(w_i)$ refer to the maximum likelihood estimate (MLE) of the probability of word w_i, and similarly $\widehat{P}(w_i|w_{i-1})$ for the probability conditioned on a history, or more generally, $\widehat{P}(w_i|w_{i-n+1}^{i-1})$.

Let $N_-(w_i) := |\{w : c(w_i, w) > 0\}|$ be the number of distinct words that appear before w_i. More generally, let $N_-(w_{i-n+1}^i) = |\{w : c(w_{i-n+1}^i, w) > 0\}|$. Similarly, let $N_+(w_{i-n+1}^{i-1}) = |\{w : c(w, w_{i-n+1}^{i-1}) > 0\}|$. V denotes the vocabulary size.

2.2 Absolute Discounting

Absolute discounting works on the idea of interpolating higher order n-gram models with lower-order n-gram models. However, first some probability mass must be "subtracted" from the higher order n-grams so that the leftover probability can be allocated to the lower order n-grams. More specifically, define the following discounted conditional probability:

$$\widehat{P}_D(w_i|w_{i-n+1}^{i-1}) = \frac{\max\{c(w_i, w_{i-n+1}^{i-1}) - D, 0\}}{c(w_{i-n+1}^{i-1})}$$

Then absolute discounting $P_{\text{abs}}(\cdot)$ uses the following (recursive) equation:

$$P_{\text{abs}}(w_i|w_{i-n+1}^{i-1}) = \widehat{P}_D(w_i|w_{i-n+1}^{i-1}) + \gamma(w_{i-n+1}^{i-1}) P_{\text{abs}}(w_i|w_{i-n+2}^{i-1})$$

where $\gamma(w_{i-n+1}^{i-1})$ is the leftover weight (due to the discounting) that is chosen so that the conditional distribution sums to one: $\gamma(w_{i-n+1}^{i-1}) = \frac{D}{c(w_{i-n+1}^{i-1})} N_+(w_{i-n+1}^{i-1})$. For the base case, we set $P_{\text{abs}}(w_i) = \widehat{P}(w_i)$.

Discontinuity: Note that if $c(w_{i-n+1}^{i-1}) = 0$, then $\gamma(w_{i-n+1}^{i-1}) = \frac{0}{0}$, in which case $\gamma(w_{i-n+1}^{i-1})$ is set to 1. We will see that this discontinuity appears in PLRE as well.

2.3 Kneser Ney Smoothing

Ideally, the smoothed probability should preserve the observed unigram distribution:

$$\widehat{P}(w_i) = \sum_{w_{i-n+1}^{i-1}} P_{\text{sm}}(w_i|w_{i-n+1}^{i-1})\widehat{P}(w_{i-n+1}^{i-1}) \quad (1)$$

where $P_{\text{sm}}(w_i|w_{i-n+1}^{i-1})$ is the smoothed conditional probability that a model outputs. Unfortunately, absolute discounting does not satisfy this property, since it exclusively uses the unaltered MLE unigram model as its lower order model. In practice, the lower order distribution is only utilized when we are unsure about the higher order distribution (i.e., when $\gamma(\cdot)$ is large). Therefore, the unigram model should be altered to condition on this fact.

This is the inspiration behind Kneser-Ney (KN) smoothing, an elegant algorithm with robust performance in n-gram language modeling. KN smoothing defines alternate probabilities $P^{\text{alt}}(\cdot)$:

$$P_D^{\text{alt}}(w_i|w_{i-n'+1}^{i-1}) = \begin{cases} \widehat{P}_D(w_i|w_{i-n'+1}^{i-1}), & \text{if } n' = n \\ \frac{\max\{N_-(w_{i-n'+1}^{i})-D,0\}}{\sum_{w_i} N_-(w_{i-n'+1}^{i})}, & \text{if } n' < n \end{cases}$$

The base case for unigrams reduces to $P^{\text{alt}}(w_i) = \frac{N_-(w_i)}{\sum_{w_i} N_-(w_i)}$. Intuitively $P^{\text{alt}}(w_i)$ is proportional to the number of unique words that precede w_i. Thus, words that appear in many different contexts will be given higher weight than words that consistently appear after only a few contexts. These alternate distributions are then used with absolute discounting:

$$P_{\text{kn}}(w_i|w_{i-n+1}^{i-1}) = P_D^{\text{alt}}(w_i|w_{i-n+1}^{i-1}) + \gamma(w_{i-n+1}^{i-1})P_{\text{kn}}(w_i|w_{i-n+2}^{i-1}) \quad (2)$$

where we set $P_{\text{kn}}(w_i) = P^{\text{alt}}(w_i)$. By definition, KN smoothing satisfies the marginal constraint in Eq. 1 (Kneser and Ney, 1995).

3 Power Low Rank Ensembles

In n-gram smoothing methods, if a bigram count $c(w_i, w_{i-1})$ is zero, the unigram probabilities are used, which is equivalent to assuming that w_i and w_{i-1} are independent (and similarly for general n). However, in this situation, instead of backing off to a 1-gram, we may like to back off to a "1.5-gram" or more generally an order between 1 and 2 that captures a coarser level of dependence between w_i and w_{i-1} and does not assume full independence.

Inspired by this intuition, our strategy is to construct an ensemble of matrices and tensors that not only consists of MLE-based count information, but also contains quantities that represent levels of dependence in-between the various orders in the model. We call these combinations power low rank ensembles (PLRE), and they can be thought of as n-gram models with non-integer n. Our approach can be recursively formulated as:

$$\begin{aligned} P_{\text{plre}}(w_i|w_{i-n+1}^{i-1}) &= P_{\boldsymbol{D}_0}^{\text{alt}}(w_i|w_{i-n+1}^{i-1}) \\ &+ \gamma_0(w_{i-n+1}^{i-1})\Big(\boldsymbol{Z}_{\boldsymbol{D}_1}(w_i|w_{i-n+1}^{i-1}) + ... \\ &+ \gamma_{\eta-1}(w_{i-n+1}^{i-1})\Big(\boldsymbol{Z}_{\boldsymbol{D}_\eta}(w_i|w_{i-n+1}^{i-1}) \\ &+ \gamma_\eta(w_{i-n+1}^{i-1})\Big(P_{\text{plre}}(w_i|w_{i-n+2}^{i-1})\Big)\Big)...\Big) \end{aligned} \quad (3)$$

where $\boldsymbol{Z}_1, ..., \boldsymbol{Z}_\eta$ are conditional probability matrices that represent the intermediate n-gram orders[1] and $\boldsymbol{D}$ is a discount function (specified in §4).

This formulation begs answers to a few critical questions. How to construct matrices that represent conditional probabilities for intermediate n? How to transform them in a way that generalizes the altered lower order distributions in KN smoothing? How to combine these matrices such that the marginal constraint in Eq. 1 still holds? The following propose solutions to these three queries:

1. **Rank** (Section 3.1): *Rank* gives us a concrete measurement of the dependence between w_i and w_{i-1}. By constructing low rank approximations of the bigram count matrix and higher-order count tensors, we obtain matrices that represent coarser dependencies, with a rank one approximation implying that the variables are independent.

2. **Power** (Section 3.2): In KN smoothing, the lower order distributions are not the original counts but rather altered estimates. We propose a continuous generalization of this alteration by taking the element-wise *power* of the counts.

[1] with a slight abuse of notation, let $\boldsymbol{Z}_{\boldsymbol{D}_j}$ be shorthand for $\boldsymbol{Z}_{j,\boldsymbol{D}_j}$

3. **Creating the Ensemble** (Section 4): Lastly, PLRE also defines a way to interpolate the specifically constructed intermediate n-gram matrices. Unfortunately a constant discount, as presented in Section 2, will not in general preserve the lower order marginal constraint (Eq. 1). We propose a generalized discounting scheme to ensure the constraint holds.

3.1 Rank

We first show how rank can be utilized to construct quantities between an n-gram and an $n-1$-gram. In general, we think of an n-gram as an n^{th} order tensor i.e. a multi-way array with n indices $\{i_1, ..., i_n\}$. (A vector is a tensor of order 1, a matrix is a tensor of order 2 etc.) Computing a special rank one approximation of slices of this tensor produces the $n-1$-gram. Thus, taking rank κ approximations in this fashion allows us to represent dependencies between an n-gram and $n-1$-gram.

Consider the bigram count matrix $\boldsymbol{B}$ with N counts which has rank V. Note that $\widehat{P}(w_i|w_{i-1}) = \frac{\boldsymbol{B}(w_i,w_{i-1})}{\sum_w \boldsymbol{B}(w,w_{i-1})}$. Additionally, $\boldsymbol{B}$ can be considered a random variable that is the result of sampling N tuples of (w_i, w_{i-1}) and agglomerating them into a count matrix. Assuming w_i and w_{i-1} are independent, the expected value (with respect to the empirical distribution) $\mathbb{E}[\boldsymbol{B}] = NP(w_i)P(w_{i-1})$, which can be rewritten as being proportional to the outer product of the unigram probability vector with itself, and is thus rank one.

This observation extends to higher order n-grams as well. Let $\boldsymbol{C}^n$ be the n^{th} order tensor where $\boldsymbol{C}^n(w_i, ..., w_{i-n+1}) = c(w_i, ..., w_{i-n+1})$. Furthermore denote $\boldsymbol{C}^n(:, \tilde{w}_{i-n+2}^{i-1}, :)$ to be the $V \times V$ matrix slice of $\boldsymbol{C}^n$ where $w_{i-n+2}, ..., w_{i-1}$ are held fixed to a particular sequence $\tilde{w}_{i-n+2}, ..., \tilde{w}_{i-1}$. Then if w_i is conditionally independent of w_{i-n+1} given w_{i-n+2}^{i-1}, then $\mathbb{E}[\boldsymbol{C}^n(:, \tilde{w}_{i-n+2}^{i-1}, :)]$ is rank one $\forall \tilde{w}_{i-n+2}^{i-1}$.

However, it is rare that these matrices are actually rank one, either due to sampling variance or the fact that w_i and w_{i-1} are not independent. What we would really like to say is that the *best* rank one approximation $\boldsymbol{B}^{(1)}$ (under some norm) of $\boldsymbol{B}$ is $\propto \widehat{P}(w_i)\widehat{P}(w_{i-1})$. While this statement is not true under the ℓ_2 norm, it is true under generalized KL divergence (Lee and Seung, 2001): $gKL(\boldsymbol{A}||\boldsymbol{B}) = \sum_{ij}\Big(\boldsymbol{A}_{ij}\log(\frac{\boldsymbol{A}_{ij}}{\boldsymbol{B}_{ij}}) - \boldsymbol{A}_{ij} + \boldsymbol{B}_{ij})\Big)$.

In particular, generalized KL divergence preserves row and column sums: *if $\boldsymbol{M}^{(\kappa)}$ is the best rank κ approximation of $\boldsymbol{M}$ under gKL then the row sums and column sums of $\boldsymbol{M}^{(\kappa)}$ and $\boldsymbol{M}$ are equal* (Ho and Van Dooren, 2008). Leveraging this property, it is straightforward to prove the following lemma:

Lemma 1. *Let $\boldsymbol{B}^{(\kappa)}$ be the best rank κ approximation of $\boldsymbol{B}$ under gKL. Then $\boldsymbol{B}^{(1)} \propto \widehat{P}(w_i)\widehat{P}(w_{i-1})$ and $\forall w_{i-1}$ s.t. $c(w_{i-1}) \neq 0$:*

$$\widehat{P}(w_i) = \frac{\boldsymbol{B}^{(1)}(w_i, w_{i-1})}{\sum_w \boldsymbol{B}^{(1)}(w, w_{i-1})}$$

For more general n, let $\boldsymbol{C}^{n,(\kappa)}_{i-1,...,i-n+2}$ be the best rank κ approximation of $\boldsymbol{C}^n(:, \tilde{w}_{i-n+2}^{i-1}, :)$ under gKL. Then similarly, $\forall w_{i-n+1}^{i-1}$ s.t. $c(w_{i-n+1}^{i-1}) > 0$:

$$\widehat{P}(w_i|w_{i-1}, ..., w_{i-n+2}) = \frac{\boldsymbol{C}^{n,(1)}_{i-1,...,i-n+2}(w_i, w_{i-n+1}^{i-1})}{\sum_w \boldsymbol{C}^{n,(1)}_{i-1,...,i-n+2}(w, w_{i-n+1}^{i-1})} \quad (4)$$

Thus, by selecting $1 < \kappa < V$, we obtain count matrices and tensors between n and $n-1$-grams. The condition that $c(w_{i-n+1}^{i-1}) > 0$ corresponds to the discontinuity discussed in §2.2.

3.2 Power

Since KN smoothing alters the lower order distributions instead of simply using the MLE, varying the rank is not sufficient in order to generalize this suite of techniques. Thus, PLRE computes low rank approximations of altered count matrices. Consider taking the elementwise power ρ of the bigram count matrix, which is denoted by $\boldsymbol{B}^{\cdot\rho}$. For example, the observed bigram count matrix and associated row sum:

$$\boldsymbol{B}^{\cdot 1} = \begin{pmatrix} 1.0 & 2.0 & 1.0 \\ 0 & 5.0 & 0 \\ 2.0 & 0 & 0 \end{pmatrix} \overset{\text{row sum}}{\rightarrow} \begin{pmatrix} 4.0 \\ 5.0 \\ 2.0 \end{pmatrix}$$

As expected the row sum is equal to the unigram counts (which we denote as $\boldsymbol{u}$). Now consider $\boldsymbol{B}^{\cdot 0.5}$:

$$\boldsymbol{B}^{\cdot 0.5} = \begin{pmatrix} 1.0 & 1.4 & 1.0 \\ 0 & 2.2 & 0 \\ 1.4 & 0 & 0 \end{pmatrix} \overset{\text{row sum}}{\rightarrow} \begin{pmatrix} 3.4 \\ 2.2 \\ 1.4 \end{pmatrix}$$

Note how the row sum vector has been altered. In particular since w_1 (corresponding to the first

row) has a more diverse history than w_2, it has a higher row sum (compared to in $\boldsymbol{u}$ where w_2 has the higher row sum). Lastly, consider the case when $p = 0$:

$$\boldsymbol{B}^{\cdot 0} = \begin{pmatrix} 1.0 & 1.0 & 1.0 \\ 0 & 1.0 & 0 \\ 1.0 & 0 & 0 \end{pmatrix} \overset{\text{row sum}}{\rightarrow} \begin{pmatrix} 3.0 \\ 1.0 \\ 1.0 \end{pmatrix}$$

The row sum is now the number of unique words that precede w_i (since $\boldsymbol{B}^0$ is binary) and is thus equal to the (unnormalized) Kneser Ney unigram. This idea also generalizes to higher order n-grams and leads us to the following lemma:

Lemma 2. *Let $\boldsymbol{B}^{(\rho,\kappa)}$ be the best rank κ approximation of $\boldsymbol{B}^{\cdot\rho}$ under gKL. Then $\forall w_{i-1}$ s.t. $c(w_{i-1}) \neq 0$:*

$$P^{alt}(w_i) = \frac{\boldsymbol{B}^{(0,1)}(w_i, w_{i-1})}{\sum_w \boldsymbol{B}^{(0,1)}(w, w_{i-1})}$$

For more general n, let $\boldsymbol{C}^{n,(\rho,\kappa)}_{i-1,\ldots,i-n+2}$ be the best rank κ approximation of $\boldsymbol{C}^{n,(\rho)}(:, \tilde{w}^{i-1}_{i-n+2}, :)$ under gKL. Similarly, $\forall w^{i-1}_{i-n+1}$ s.t. $c(w^{i-1}_{i-n+1}) > 0$:

$$P^{alt}(w_i|w_{i-1}, ..., w_{i-n+2}) = \frac{\boldsymbol{C}^{n,(0,1)}_{i-1,\ldots,i-n+2}(w_i, w^{i-1}_{i-n+1})}{\sum_w \boldsymbol{C}^{n,(0,1)}_{i-1,\ldots,i-n+2}(w, w^{i-1}_{i-n+1})} \quad (5)$$

4 Creating the Ensemble

Recall our overall formulation in Eq. 3; a naive solution would be to set $\boldsymbol{Z}_1, ..., \boldsymbol{Z}_\eta$ to low rank approximations of the count matrices/tensors under varying powers, and then interpolate through constant absolute discounting. Unfortunately, the marginal constraint in Eq. 1 will generally not hold if this strategy is used. Therefore, we propose a generalized discounting scheme where each nonzero n-gram count is associated with a different discount $\boldsymbol{D}_j(w_i, w^{i-1}_{i-n'+1})$. The low rank approximations are then computed on the discounted matrices, leaving the marginal constraint intact.

For clarity of exposition, we focus on the special case where $n = 2$ with only one low rank matrix before stating our general algorithm:

$$P_{\text{plre}}(w_i|w_{i-1}) = \widehat{P}_{\boldsymbol{D}_0}(w_i|w_{i-1}) + \gamma_0(w_{i-1})\Big(\boldsymbol{Z}_{\boldsymbol{D}_1}(w_i|w_{i-1}) + \gamma_1(w_{i-1})P^{\text{alt}}(w_i)\Big) \quad (6)$$

Our goal is to compute $\boldsymbol{D}_0, \boldsymbol{D}_1$ and $\boldsymbol{Z}_1$ so that the following lower order marginal constraint holds:

$$\widehat{P}(w_i) = \sum_{w_{i-1}} P_{\text{plre}}(w_i|w_{i-1})\widehat{P}(w_{i-1}) \quad (7)$$

Our solution can be thought of as a two-step procedure where we compute the discounts $\boldsymbol{D}_0, \boldsymbol{D}_1$ (and the $\gamma(w_{i-1})$ weights as a by-product), followed by the low rank quantity $\boldsymbol{Z}_1$. First, we construct the following intermediate ensemble of powered, but full rank terms. Let $\boldsymbol{Y}^{\rho_j}$ be the matrix such that $\boldsymbol{Y}^{\rho_j}(w_i, w_{i-1}) := c(w_i, w_{i-1})^{\rho_j}$. Then define

$$P_{\text{pwr}}(w_i|w_{i-1}) := \boldsymbol{Y}^{(\rho_0=1)}_{\boldsymbol{D}_0}(w_i|w_{i-1}) + \gamma_0(w_{i-1})\Big(\boldsymbol{Y}^{(\rho_1)}_{\boldsymbol{D}_1}(w_i|w_{i-1}) + \gamma_1(w_{i-1})\boldsymbol{Y}^{(\rho_2=0)}(w_i|w_{i-1})\Big) \quad (8)$$

where with a little abuse of notation:

$$\boldsymbol{Y}^{\rho_j}_{\boldsymbol{D}_j}(w_i|w_{i-1}) = \frac{c(w_i, w_{i-1})^{\rho_j} - \boldsymbol{D}_j(w_i, w_{i-1})}{\sum_{w_i} c(w_i, w_{i-1})^{\rho_j}}$$

Note that $P^{\text{alt}}(w_i)$ has been replaced with $\boldsymbol{Y}^{(\rho_2=0)}(w_i|w_{i-1})$, based on Lemma 2, and will equal $P^{\text{alt}}(w_i)$ once the low rank approximation is taken as discussed in § 4.2).

Since we have only combined terms of different power (but all full rank), it is natural choose the discounts so that the result remains unchanged i.e., $P_{\text{pwr}}(w_i|w_{i-1}) = \widehat{P}(w_i|w_{i-1})$, since the low rank approximation (not the power) will implement smoothing. Enforcing this constraint gives rise to a set of linear equations that can be solved (in closed form) to obtain the discounts as we now show below.

4.1 Step 1: Computing the Discounts

To ensure the constraint that $P_{\text{pwr}}(w_i|w_{i-1}) = \widehat{P}(w_i|w_{i-1})$, it is sufficient to enforce the following two local constraints:

$$\boldsymbol{Y}^{(\rho_j)}(w_i|w_{i-1}) = \boldsymbol{Y}^{(\rho_j)}_{\boldsymbol{D}_j}(w_i|w_{i-1}) + \gamma_j(w_{i-1})\boldsymbol{Y}^{(\rho_{j+1})}(w_i|w_{i-1}) \text{ for } j = 0, 1 \quad (9)$$

This allows each $\boldsymbol{D}_j$ to be solved for independently of the other $\{\boldsymbol{D}_{j'}\}_{j' \neq j}$. Let $c_{i,i-1} = c(w_i, w_{i-1})$, $c^j_{i,i-1} = c(w_i, w_{i-1})^{\rho_j}$, and $d^j_{i,i-1} =$

$\boldsymbol{D}_j(w_i, w_{i-1})$. Expanding Eq. 9 yields that $\forall w_i, w_{i-1}$:

$$\frac{c^j_{i,i-1}}{\sum_i c^j_{i,i-1}} = \frac{c^j_{i,i-1} - d^j_{i,i-1}}{\sum_i c^j_{i,i-1}} + \left(\frac{\sum_i d^j_{i,i-1}}{\sum_i c^j_{i,i-1}}\right) \frac{c^{j+1}_{i,i-1}}{\sum_i c^{j+1}_{i,i-1}} \quad (10)$$

which can be rewritten as:

$$-d^j_{i,i-1} + \left(\sum_i d^j_{i,i-1}\right) \frac{c^{j+1}_{i,i-1}}{\sum_i c^{j+1}_{i,i-1}} = 0 \quad (11)$$

Note that Eq. 11 decouples across w_{i-1} since the only $d^j_{i,i-1}$ terms that are dependent are the ones that share the preceding context w_{i-1}.

It is straightforward to see that setting $d^j_{i,i-1}$ proportional to $c^{j+1}_{i,i-1}$ satisfies Eq. 11. Furthermore it can be shown that all solutions are of this form (i.e., the linear system has a null space of exactly one). Moreover, we are interested in a particular subset of solutions where a single parameter d_* (independent of w_{i-1}) controls the scaling as indicated by the following lemma:

Lemma 3. *Assume that $\rho_j \geq \rho_{j+1}$. Choose any $0 \leq d_* \leq 1$. Set $d^j_{i,i-1} = d_* c^{j+1}_{i,i-1}\ \forall i, j$. The resulting discounts satisfy Eq. 11 as well as the inequality constraints $0 \leq d^j_{i,i-1} \leq c^j_{i,i-1}$. Furthermore, the leftover weight γ_j takes the form:*

$$\gamma_j(w_{i-1}) = \frac{\sum_i d^j_{i,i-1}}{\sum_i c^j_{i,i-1}} = \frac{d_* \sum_i c^{j+1}_{i,i-1}}{\sum_i c^j_{i,i-1}}$$

Proof. Clearly this choice of $d^j_{i,i-1}$ satisfies Eq. 11. The largest possible value of $d^j_{i,i-1}$ is $c^{j+1}_{i,i-1}$. $\rho_j \geq \rho_{j+1}$, implies $c^j_{i,i-1} \geq c^{j+1}_{i,i-1}$. Thus the inequality constraints are met. It is then easy to verify that γ takes the above form. □

The above lemma generalizes to longer contexts (i.e. $n > 2$) as shown in Algorithm 1. Note that if $\rho_j = \rho_{j+1}$ then Algorithm 1 is equivalent to scaling the counts e.g. deleted-interpolation/Jelinek Mercer smoothing (Jelinek and Mercer, 1980). On the other hand, when $\rho_{j+1} = 0$, Algorithm 1 is equal to the absolute discounting that is used in Kneser-Ney. Thus, depending on ρ_{j+1}, our method generalizes different types of interpolation schemes to construct an ensemble so that the marginal constraint is satisfied.

Algorithm 1 Compute $\boldsymbol{D}$

In: Count tensor $\boldsymbol{C}^n$, powers ρ_j, ρ_{j+1} such that $\rho_j \geq \rho_{j+1}$, and parameter d_*.

Out: Discount $\boldsymbol{D}_j$ for powered counts $\boldsymbol{C}^{n,(\rho_j)}$ and associated leftover weight γ_j

1: Set $\boldsymbol{D}_j(w_i, w^{i-1}_{i-n+1}) = d_* c(w_i, w^{i-1}_{i-n+1})^{\rho_{j+1}}$.

2: $$\gamma_j(w_i, w^{i-1}_{i-n+1}) = \frac{d_* \sum_{w_i} c(w_i, w^{i-1}_{i-n+1})^{\rho_{j+1}}}{\sum_{w_i} c(w_i, w^{i-1}_{i-n+1})^{\rho_j}}$$

Algorithm 2 Compute $\boldsymbol{Z}$

In: Count tensor $\boldsymbol{C}^n$, power ρ, discounts $\boldsymbol{D}$, rank κ

Out: Discounted low rank conditional probability table $\boldsymbol{Z}^{(\rho,\kappa)}_{\boldsymbol{D}}(w_i|w^{i-1}_{i-n+1})$ (represented implicitly)

1: Compute powered counts $\boldsymbol{C}^{n,(\cdot\rho)}$.

2: Compute denominators $\sum_{w_i} c(w_i, w^{i-1}_{i-n+1})^\rho$ $\forall w^{i-1}_{i-n+1}$ s.t. $c(w^{i-1}_{i-n+1}) > 0$.

3: Compute discounted powered counts $\boldsymbol{C}^{n,(\cdot\rho)}_{\boldsymbol{D}} = \boldsymbol{C}^{n,(\cdot\rho)} - \boldsymbol{D}$.

4: For each slice $\boldsymbol{M}_{\tilde{w}^{i-1}_{i-n+2}} := \boldsymbol{C}^{n,(\cdot\rho)}_{\boldsymbol{D}}(:, \tilde{w}^{i-1}_{i-n+2}, :)$ compute

$$\boldsymbol{M}^{(\kappa)} := \min_{\boldsymbol{A} \geq 0 : rank(\boldsymbol{A}) = \kappa} \|\boldsymbol{M}_{\tilde{w}^{i-1}_{i-n+2}} - \boldsymbol{A}\|_{KL}$$

(stored implicitly as $\boldsymbol{M}^{(\kappa)} = \boldsymbol{LR}$)

Set $\boldsymbol{Z}^{(\rho,\kappa)}_{\boldsymbol{D}}(:, \tilde{w}^{i-1}_{i-n+2}, :) = \boldsymbol{M}^{(\kappa)}$

5: Note that

$$\boldsymbol{Z}^{(\rho,\kappa)}_{\boldsymbol{D}}(w_i|w^{i-1}_{i-n+1}) = \frac{\boldsymbol{Z}^{(\rho,\kappa)}_{\boldsymbol{D}}(w_i, w^{i-1}_{i-n+1})}{\sum_{w_i} c(w_i, w^{i-1}_{i-n+1})^\rho}$$

4.2 Step 2: Computing Low Rank Quantities

The next step is to compute low rank approximations of $\boldsymbol{Y}^{(\rho_j)}_{\boldsymbol{D}_j}$ to obtain $\boldsymbol{Z}_{\boldsymbol{D}_j}$ such that the intermediate marginal constraint in Eq. 7 is preserved. This constraint trivially holds for the intermediate ensemble $P_{\text{pwr}}(w_i|w_{i-1})$ due to how the discounts were derived in § 4.1. For our running bigram example, define $\boldsymbol{Z}^{(\rho_j,\kappa_j)}_{\boldsymbol{D}_j}$ to be the best rank κ_j approximation to $\boldsymbol{Y}^{(\rho_j,\kappa_j)}_{\boldsymbol{D}_j}$ according to gKL and let

$$\boldsymbol{Z}^{\rho_j,\kappa_j}_{\boldsymbol{D}_j}(w_i|w_{i-1}) = \frac{\boldsymbol{Z}^{\rho_j,\kappa_j}_{\boldsymbol{D}_j}(w_i, w_{i-1})}{\sum_{w_i} c(w_i, w_{i-1})^{\rho_j}}$$

Note that $\boldsymbol{Z}^{\rho_j,\kappa_j}_{\boldsymbol{D}_j}(w_i|w_{i-1})$ is a valid (discounted) conditional probability since gKL preserves row/column sums so the denominator remains unchanged under the low rank approximation. Then

using the fact that $\boldsymbol{Z}^{(0,1)}(w_i|w_{i-1}) = P^{\text{alt}}(w_i)$ (Lemma 2) we can embellish Eq. 6 as

$$P_{\text{plre}}(w_i|w_{i-1}) = P_{\boldsymbol{D}_0}(w_i|w_{i-1}) + \gamma_0(w_{i-1})\left(\boldsymbol{Z}_{\boldsymbol{D}_1}^{(\rho_1,\kappa_1)}(w_i|w_{i-1}) + \gamma_1(w_{i-1})P_{\text{alt}}(w_i)\right)$$

Leveraging the form of the discounts and row/column sum preserving property of gKL, we then have the following lemma (the proof is in the supplementary material):

Lemma 4. *Let $P_{plre}(w_i|w_{i-1})$ indicate the PLRE smoothed conditional probability as computed by Eq. 6 and Algorithms 1 and 2. Then, the marginal constraint in Eq. 7 holds.*

4.3 More general algorithm

In general, the principles outlined in the previous sections hold for higher order n-grams. Assume that the discounts are computed according to Algorithm 1 with parameter d_* and $\boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}$ is computed according to Algorithm 2. Note that, as shown in Algorithm 2, for higher order n-grams, the $\boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}$ are created by taking low rank approximations of slices of the (powered) count tensors (see Lemma 2 for intuition). Eq. 3 can now be embellished:

$$\begin{aligned} P_{\text{plre}}(w_i|w_{i-n+1}^{i-1}) &= P_{\boldsymbol{D}_0}^{\text{alt}}(w_i|w_{i-n+1}^{i-1}) \\ &+ \gamma_0(w_{i-n+1}^{i-1})\left(\boldsymbol{Z}_{\boldsymbol{D}_1}^{(\rho_1,\kappa_1)}(w_i|w_{i-n+1}^{i-1}) + ...\right. \\ &+ \gamma_{\eta-1}(w_{i-n+1}^{i-1})\left(\boldsymbol{Z}_{\boldsymbol{D}_\eta}^{(\rho_\eta,\kappa_\eta)}(w_i|w_{i-n+1}^{i-1})\right. \\ &\left.\left. + \gamma_\eta(w_{i-n+1}^{i-1})\left(P_{\text{plre}}(w_i|w_{i-n+2}^{i-1})\right)\right)...\right) \quad (12) \end{aligned}$$

Lemma 4 also applies in this case and is given in Theorem 1 in the supplementary material.

4.4 Links with KN Smoothing

In this section, we explicitly show the relationship between PLRE and KN smoothing. Rewriting Eq. 12 in the following form:

$$\begin{aligned} P_{\text{plre}}(w_i|w_{i-n+1}^{i-1}) &= P_{\text{plre}}^{\text{terms}}(w_i|w_{i-n+1}^{i-1}) \\ &+\gamma_{0:\eta}(w_{i-n+1}^{i-1})P_{\text{plre}}(w_i|w_{i-n+2}^{i-1}) \quad (13) \end{aligned}$$

where $P_{\text{plre}}^{\text{terms}}(w_i|w_{i-n+1}^{i-1})$ contains the terms in Eq. 12 except the last, and $\gamma_{0:\eta}(w_{i-n+1}^{i-1}) = \prod_{h=0}^{\eta}\gamma_h(w_{i-n+1}^{i-1})$, we can leverage the form of the discount, and using the fact that $\rho_{\eta+1} = 0$[2]:

$$\gamma_{0:\eta}(w_{i-n-1}^{i-1}) = \frac{{d_*}^{\eta+1}N_+(w_{i-n+1}^{i-1})}{c(w_{i-n+1}^{i-1})}$$

With this form of $\gamma(\cdot)$, Eq. 13 is remarkably similar to KN smoothing (Eq. 2) if KN's discount parameter D is chosen to equal $(d_*)^{\eta+1}$.

The difference is that $P^{\text{alt}}(\cdot)$ has been replaced with the alternate estimate $P_{\text{plre}}^{\text{terms}}(w_i|w_{i-n+1}^{i-1})$, which have been enriched via the low rank structure. Since these alternate estimates were constructed via our ensemble strategy they contain both very fine-grained dependencies (the original n-grams) as well as coarser dependencies (the lower rank n-grams) and is thus fundamentally different than simply taking a single matrix/tensor decomposition of the trigram/bigram matrices.

Moreover, it provides a natural way of setting d_* based on the Good-Turing (GT) estimates employed by KN smoothing. In particular, we can set d_* to be the $(\eta+1)^{\text{th}}$ root of the KN discount D that can be estimated via the GT estimates.

4.5 Computational Considerations

PLRE scales well even as the order n increases. To compute a low rank bigram, one low rank approximation of a $V \times V$ matrix is required. For the low rank trigram, we need to compute a low rank approximation of each slice $\boldsymbol{C}_{\boldsymbol{D}}^{n,(\cdot p)}(:, \tilde{w}_{i-1}, :)\ \forall \tilde{w}_{i-1}$. While this may seem daunting at first, in practice the *size* of each slice (number of non-zero rows/columns) is usually much, much smaller than V, keeping the computation tractable.

Similarly, PLRE also evaluates conditional probabilities at evaluation time efficiently. As shown in Algorithm 2, the normalizer can be precomputed on the sparse powered matrix/tensor. As a result our test complexity is $\mathcal{O}(\sum_{i=1}^{\eta_{\text{total}}}\kappa_i)$ where η_{total} is the total number of matrices/tensors in the ensemble. While this is larger than Kneser Ney's practically constant complexity of $\mathcal{O}(n)$, it is much faster than other recent methods for language modeling such as neural networks and conditional exponential family models where exact computation of the normalizing constant costs $O(V)$.

5 Experiments

To evaluate PLRE, we compared its performance on English and Russian corpora with several vari-

[2] for derivation see proof of Lemma 4 in the supplementary material

ants of KN smoothing, class-based models, and the log-bilinear neural language model (Mnih and Hinton, 2007). We evaluated with perplexity in most of our experiments, but also provide results evaluated with BLEU (Papineni et al., 2002) on a downstream machine translation (MT) task. We have made the code for our approach publicly available [3].

To build the hard class-based LMs, we utilized `mkcls`[4], a tool to train word classes that uses the maximum likelihood criterion (Och, 1995) for classing. We subsequently trained trigram class language models on these classes (corresponding to 2nd-order HMMs) using SRILM (Stolcke, 2002), with KN-smoothing for the class transition probabilities. SRILM was also used for the baseline KN-smoothed models.

For our MT evaluation, we built a hierarchical phrase translation (Chiang, 2007) system using `cdec` (Dyer et al., 2010). The KN-smoothed models in the MT experiments were compiled using KenLM (Heafield, 2011).

5.1 Datasets

For the perplexity experiments, we evaluated our proposed approach on 4 datasets, 2 in English and 2 in Russian. In all cases, the singletons were replaced with "<unk>" tokens in the training corpus, and any word not in the vocabulary was replaced with this token during evaluation. There is a general dearth of evaluation on large-scale corpora in morphologically rich languages such as Russian, and thus we have made the processed Large-Russian corpus available for comparison [3].

- **Small-English**: APNews corpus (Bengio et al., 2003): Train - 14 million words, Dev - 963,000, Test - 963,000. Vocabulary- 18,000 types.
- **Small-Russian**: Subset of Russian news commentary data from 2013 WMT translation task[5]: Train- 3.5 million words, Dev - 400,000 Test - 400,000. Vocabulary - 77,000 types.
- **Large-English**: English Gigaword, Training - 837 million words, Dev - 8.7 million, Test - 8.7 million. Vocabulary- 836,980 types.
- **Large-Russian**: Monolingual data from WMT 2013 task. Training - 521 million words, Validation - 50,000, Test - 50,000. Vocabulary- 1.3 million types.

[3] http://www.cs.cmu.edu/∼apparikh/plre.html

[4] http://code.google.com/p/giza-pp/

[5] http://www.statmt.org/wmt13/training-monolingual-nc-v8.tgz

For the MT evaluation, we used the parallel data from the WMT 2013 shared task, excluding the Common Crawl corpus data. The newstest2012 and newstest2013 evaluation sets were used as the development and test sets respectively.

5.2 Small Corpora

For the class-based baseline LMs, the number of classes was selected from $\{32, 64, 128, 256, 512, 1024\}$ (Small-English) and $\{512, 1024\}$ (Small-Russian). We could not go higher due to the computationally laborious process of hard clustering. For Kneser-Ney, we explore four different variants: back-off (BO-KN) interpolated (int-KN), modified back-off (BO-MKN), and modified interpolated (int-MKN). Good-Turing estimates were used for discounts. All models trained on the small corpora are of order 3 (trigrams).

For PLRE, we used one low rank bigram and one low rank trigram in addition to the MLE n-gram estimates. The powers of the intermediate matrices/tensors were fixed to be 0.5 and the discounts were set to be square roots of the Good Turing estimates (as explained in § 4.4). The ranks were tuned on the development set. For Small-English, the ranges were $\{1e-3, 5e-3\}$ (as a fraction of the vocabulary size) for both the low rank bigram and low rank trigram models. For Small-Russian the ranges were $\{5e-4, 1e-3\}$ for both the low rank bigram and the low rank trigram models.

The results are shown in Table 1. The best class-based LM is reported, but is not competitive with the KN baselines. PLRE outperforms all of the baselines comfortably. Moreover, PLRE's performance over the baselines is highlighted in Russian. With larger vocabulary sizes, the low rank approach is more effective as it can capture linguistic similarities between rare and common words.

Next we discuss how the maximum n-gram order affects performance. Figure 1 shows the relative percentage improvement of our approach over int-MKN as the order is increased from 2 to 4 for both methods. The Small-English dataset has a rather small vocabulary compared to the number of tokens, leading to lower data sparsity in the bigram. Thus the PLRE improvement is small for order $= 2$, but more substantial for order $= 3$. On the other hand, for the Small-Russian dataset, the vocabulary size is much larger and consequently the bigram counts are sparser. This leads to sim-

Dataset	class-1024(3)	BO-KN(3)	int-KN(3)	BO-MKN(3)	int-MKN(3)	PLRE(3)
Small-English Dev	115.64	99.20	99.73	99.95	95.63	**91.18**
Small-English Test	119.70	103.86	104.56	104.55	100.07	**95.15**
Small-Russian Dev	286.38	281.29	265.71	287.19	263.25	**241.66**
Small-Russian Test	284.09	277.74	262.02	283.70	260.19	**238.96**

Table 1: Perplexity results on small corpora for all methods.

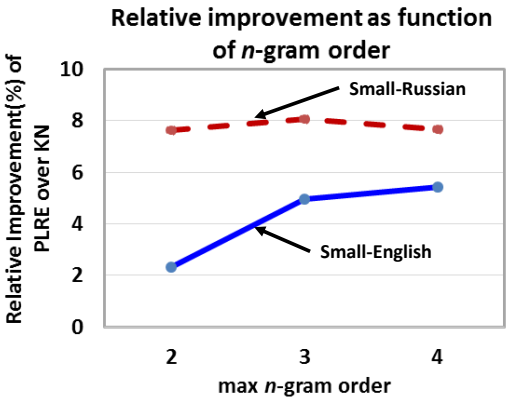


Figure 1: Relative percentage improvement of PLRE over int-MKN as the maximum n-gram order for both methods is increased.

ilar improvements for all orders (which are larger than that for Small-English).

On both these datasets, we also experimented with tuning the discounts for int-MKN to see if the baseline could be improved with more careful choices of discounts. However, this achieved only marginal gains (reducing the perplexity to 98.94 on the Small-English test set and 259.0 on the Small-Russian test set).

Comparison to LBL (Mnih and Hinton, 2007): Mnih and Hinton (2007) evaluate on the Small-English dataset (but remove end markers and concatenate the sentences). They obtain perplexities 117.0 and 107.8 using contexts of size 5 and 10 respectively. With this preprocessing, a 4-gram (context 3) PLRE achieves 108.4 perplexity.

5.3 Large Corpora

Results on the larger corpora for the top 2 performing methods "PLRE" and "int-MKN" are presented in Table 2. Due to the larger training size, we use 4-gram models in these experiments. However, including the low rank 4-gram tensor provided little gain and therefore, the 4-gram PLRE only has additional low rank bigram and low rank trigram matrices/tensors. As above, ranks were tuned on the development set. For Large-English, the ranges were $\{1e-4, 5e-4, 1e-3\}$ (as a fraction of the vocabulary size) for both the low rank bigram and low rank trigram models. For Small-Russian the ranges were $\{1e-5, 5e-5, 1e-4\}$ for both the low rank bigram and the low rank trigram models. For statistical validity, 10 test sets of size equal to the original test set were generated by randomly sampling sentences with replacement from the original test set. Our method outperforms "int-MKN" with gains similar to that on the smaller datasets. As shown in Table 3, our method obtains fast training times even for large datasets.

Dataset	int-MKN(4)	PLRE(4)
Large-English Dev	73.21	**71.21**
Large-English Test	77.90 $\pm$ 0.203	**75.66** $\pm$ 0.189
Large-Russian Dev	326.9	**297.11**
Large-Russian Test	289.63 $\pm$ 6.82	**264.59** $\pm$ 5.839

Table 2: Mean perplexity results on large corpora, with standard deviation.

Dataset	PLRE Training Time
Small-English	3.96 min (order 3) / 8.3 min (order 4)
Small-Russian	4.0 min (order 3) / 4.75 min (order 4)
Large-English	3.2 hrs (order 4)
Large-Russian	8.3 hrs (order 4)

Table 3: PLRE training times for a fixed parameter setting[6]. 8 Intel Xeon CPUs were used.

Method	BLEU
int-MKN(4)	17.63 $\pm$ 0.11
PLRE(4)	17.79 $\pm$ 0.07
Smallest Diff	PLRE+0.05
Largest Diff	PLRE+0.29

Table 4: Results on English-Russian translation task (mean $\pm$ stdev). See text for details.

6 Machine Translation Task

Table 4 presents results for the MT task, translating from English to Russian[7]. We used MIRA (Chiang et al., 2008) to learn the feature weights. To control for the randomness in MIRA, we avoid retuning when switching LMs - the set of feature weights obtained using int-MKN is the same, only the language model changes. The

[6]As described earlier, only the ranks need to be tuned, so only 2-3 low rank bigrams and 2-3 low rank trigrams need to be computed (and combined depending on the setting).

[7]the best score at WMT 2013 was 19.9 (Bojar et al., 2013)

procedure is repeated 10 times to control for optimizer instability (Clark et al., 2011). Unlike other recent approaches where an additional feature weight is tuned for the proposed model and used in conjunction with KN smoothing (Vaswani et al., 2013), our aim is to show the improvements that PLRE provides as a substitute for KN. On average, PLRE outperforms the KN baseline by 0.16 BLEU, and this improvement is consistent in that PLRE never gets a worse BLEU score.

7 Related Work

Recent attempts to revisit the language modeling problem have largely come from two directions: Bayesian nonparametrics and neural networks. Teh (2006) and Goldwater et al. (2006) discovered the connection between interpolated Kneser Ney and the hierarchical Pitman-Yor process. These have led to generalizations that account for domain effects (Wood and Teh, 2009) and unbounded contexts (Wood et al., 2009).

The idea of using neural networks for language modeling is not new (Miikkulainen and Dyer, 1991), but recent efforts (Mnih and Hinton, 2007; Mikolov et al., 2010) have achieved impressive performance. These methods can be quite expensive to train and query (especially as the vocabulary size increases). Techniques such as noise contrastive estimation (Gutmann and Hyvärinen, 2012; Mnih and Teh, 2012; Vaswani et al., 2013), subsampling (Xu et al., 2011), or careful engineering approaches for maximum entropy LMs (which can also be applied to neural networks) (Wu and Khudanpur, 2000) have improved training of these models, but querying the probability of the next word given still requires explicitly normalizing over the vocabulary, which is expensive for big corpora or in languages with a large number of word types. Mnih and Teh (2012) and Vaswani et al. (2013) propose setting the normalization constant to 1, but this is approximate and thus can only be used for downstream evaluation, not for perplexity computation. An alternate technique is to use word-classing (Goodman, 2001; Mikolov et al., 2011), which can reduce the cost of exact normalization to $O(\sqrt{V})$. In contrast, our approach is much more scalable, since it is trivially parallelized in training and does not require explicit normalization during evaluation.

There are a few low rank approaches (Saul and Pereira, 1997; Bellegarda, 2000; Hutchinson et al., 2011), but they are only effective in restricted settings (e.g. small training sets, or corpora divided into documents) and do not generally perform comparably to state-of-the-art models. Roark et al. (2013) also use the idea of marginal constraints for re-estimating back-off parameters for heavily-pruned language models, whereas we use this concept to estimate n-gram specific discounts.

8 Conclusion

We presented power low rank ensembles, a technique that generalizes existing n-gram smoothing techniques to non-integer n. By using ensembles of sparse as well as low rank matrices and tensors, our method captures both the fine-grained and coarse structures in word sequences. Our discounting strategy preserves the marginal constraint and thus generalizes Kneser Ney, and under slight changes can also extend other smoothing methods such as deleted-interpolation/Jelinek-Mercer smoothing. Experimentally, PLRE convincingly outperforms Kneser-Ney smoothing as well as class-based baselines.

Acknowledgements

This work was supported by NSF IIS1218282, NSF IIS1218749, NSF IIS1111142, NIH R01GM093156, the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533, the NSF Graduate Research Fellowship Program under Grant No. 0946825 (NSF Fellowship to APP), and a grant from Ebay Inc. (to AS).

References

Jerome R. Bellegarda. 2000. Large vocabulary speech recognition with multispan statistical language models. *IEEE Transactions on Speech and Audio Processing*, 8(1):76–84.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August. Association for Computational Linguistics.

Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. 2010. A singular value thresholding algorithm for

matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982.

Emmanuel J Candès and Benjamin Recht. 2009. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Stanley F Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for me models. *Speech and Audio Processing, IEEE Transactions on*, 8(1):37–50.

Stanley F. Chen. 2009. Shrinking exponential language models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 468–476, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 224–233. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *Comput. Linguist.*, 33(2):201–228, June.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 176–181.

Chris Dyer, Jonathan Weese, Hendra Setiawan, Adam Lopez, Ferhan Ture, Vladimir Eidelman, Juri Ganitkevitch, Phil Blunsom, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12. Association for Computational Linguistics.

Sharon Goldwater, Thomas Griffiths, and Mark Johnson. 2006. Interpolating between types and tokens by estimating power-law generators. In *Advances in Neural Information Processing Systems*, volume 18.

Joshua Goodman. 2001. Classes for fast maximum entropy training. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 561–564. IEEE.

Michael Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.

Ngoc-Diep Ho and Paul Van Dooren. 2008. Non-negative matrix factorization with fixed row and column sums. *Linear Algebra and its Applications*, 429(5):1020–1025.

Brian Hutchinson, Mari Ostendorf, and Maryam Fazel. 2011. Low rank language models for small training sets. *Signal Processing Letters, IEEE*, 18(9):489–492.

Frederick Jelinek and Robert Mercer. 1980. Interpolated estimation of markov source parameters from sparse data. *Pattern recognition in practice*.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Daniel D. Lee and H. Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13:556–562.

Lester Mackey, Ameet Talwalkar, and Michael I Jordan. 2011. Divide-and-conquer matrix factorization. *arXiv preprint arXiv:1107.0789*.

Christopher D Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*, volume 999. MIT Press.

Risto Miikkulainen and Michael G. Dyer. 1991. Natural language processing with modular pdp networks and distributed lexicon. *Cognitive Science*, 15:343–399.

Tom Mikolov, Martin Karafit, Luk Burget, Jan ernock, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, volume 2010, pages 1045–1048. International Speech Communication Association.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, JH Cernocky, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the International Conference on Machine Learning*.

Anil Kumar Nelakanti, Cedric Archambeau, Julien Mairal, Francis Bach, and Guillaume Bouchard. 2013. Structured penalties for log-linear language models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 233–243, Seattle, Washington, USA, October. Association for Computational Linguistics.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On Structuring Probabilistic Dependencies in Stochastic Language Modelling. *Computer Speech and Language*, 8:1–38.

Franz Josef Och. 1995. Maximum-likelihood-schätzung von wortkategorien mit verfahren der kombinatorischen optimierung. Bachelor's thesis (Studienarbeit), University of Erlangen.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. pages 311–318.

Lawrence Rabiner and Biing-Hwang Juang. 1993. Fundamentals of speech recognition.

Brian Roark, Cyril Allauzen, and Michael Riley. 2013. Smoothed marginal distribution constraints for language modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 43–52.

Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM.

Lawrence Saul and Fernando Pereira. 1997. Aggregate and mixed-order markov models for statistical language processing. In *Proceedings of the second conference on empirical methods in natural language processing*, pages 81–89. Somerset, New Jersey: Association for Computational Linguistics.

Andreas Stolcke. 2002. SRILM - An Extensible Language Modeling Toolkit. In *Proceedings of the International Conference in Spoken Language Processing*.

Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4.

Yee Whye Teh. 2006. A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 985–992. Association for Computational Linguistics.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, Seattle, Washington, USA, October. Association for Computational Linguistics.

Frank Wood and Yee Whye Teh. 2009. A hierarchical nonparametric Bayesian approach to statistical language model domain adaptation. In *Artificial Intelligence and Statistics*, pages 607–614.

Frank Wood, Cédric Archambeau, Jan Gasthaus, Lancelot James, and Yee Whye Teh. 2009. A stochastic memoizer for sequence data. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1129–1136. ACM.

Jun Wu and Sanjeev Khudanpur. 2000. Efficient training methods for maximum entropy language modeling. In *Interspeech*, pages 114–118.

Puyang Xu, Asela Gunawardana, and Sanjeev Khudanpur. 2011. Efficient subsampling for training complex language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1128–1136, Stroudsburg, PA, USA. Association for Computational Linguistics.

George Zipf. 1949. Human behaviour and the principle of least-effort. Addison-Wesley, Cambridge, MA.

Modeling Biological Processes for Reading Comprehension

Jonathan Berant[*], **Vivek Srikumar**[*], **Pei-Chun Chen, Brad Huang and Christopher D. Manning**
Stanford University, Stanford

Abby Vander Linden and Brittany Harding
University of Washington, Seattle

Abstract

Machine reading calls for programs that read and understand text, but most current work only attempts to extract facts from redundant web-scale corpora. In this paper, we focus on a new reading comprehension task that requires complex reasoning over a single document. The input is a paragraph describing a biological process, and the goal is to answer questions that require an understanding of the relations between entities and events in the process. To answer the questions, we first predict a rich structure representing the process in the paragraph. Then, we map the question to a formal query, which is executed against the predicted structure. We demonstrate that answering questions via predicted structures substantially improves accuracy over baselines that use shallower representations.

1 Introduction

The goal of machine reading is to develop programs that read text to learn about the world and make decisions based on accumulated knowledge. Work in this field has focused mostly on macro-reading, i.e., processing large text collections and extracting knowledge bases of facts (Etzioni et al., 2006; Carlson et al., 2010; Fader et al., 2011). Such methods rely on redundancy, and are thus suitable for answering common factoid questions which have ample evidence in text (Fader et al., 2013). However, reading a single document (micro-reading) to answer comprehension questions that require deep reasoning is currently beyond the scope of state-of-the-art systems.

In this paper, we introduce a task where given a paragraph describing a process, the goal is to answer reading comprehension questions that test understanding of the underlying structure. In particular, we consider processes in biology textbooks such as this excerpt and the question that follows:

> "... **Water is split**, providing a source of electrons and protons (hydrogen ions, H^+) and giving off O_2 as a by-product. **Light absorbed** by chlorophyll drives a **transfer of the electrons and hydrogen ions** from water to an acceptor called $NADP^+$ ..."

Q What can the splitting of water lead to?

a Light absorption

b Transfer of ions

This excerpt describes a process in which a complex set of events and entities are related to one another. A system trying to answer this question must extract a rich structure spanning multiple sentences and reason that *water splitting* combined with *light absorption* leads to *transfer of ions*. Note that shallow methods, which rely on lexical overlap or text proximity, will fail. Indeed, both answers are covered by the paragraph and the wrong answer is closer in the text to the question.

We propose a novel method that tackles this challenging problem (see Figure 1). First, we train a supervised structure predictor that learns to extract entities, events and their relations describing the biological process. This is a difficult problem because events have complex interactions that span multiple sentences. Then, treating this structure as a small knowledge-base, we map questions to formal queries that are executed against the structure to provide the answer.

Micro-reading is an important aspect of natural language understanding (Richardson et al., 2013; Kushman et al., 2014). In this work, we focus specifically on modeling processes, where events and entities relate to one another through complex interactions. While we work in the biology

[*] Both authors equally contributed to the paper.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1499–1510,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

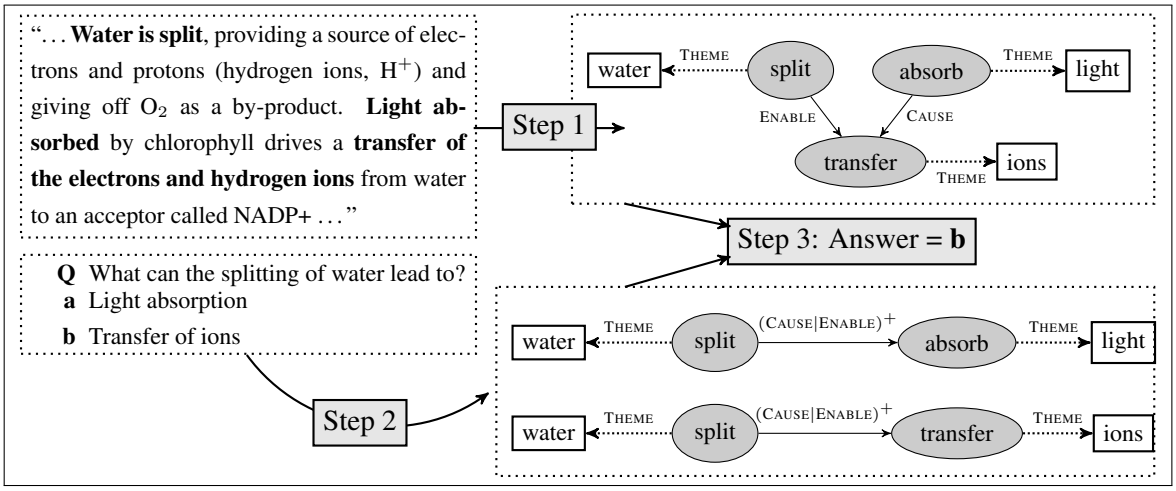


Figure 1: An overview of our reading comprehension system. First, we predict a structure from the input paragraph (the top right portion shows a partial structure skipping some arguments for brevity). Circles denote events, squares denote arguments, solid arrows represent event-event relations, and dashed arrows represent event-argument relations. Second, we map the question paired with each answer into a query that will be answered using the structure. The bottom right shows the query representation. Last, the two queries are executed against the structure, and a final answer is returned.

domain, processes are abundant in domains such as chemistry, economics, manufacturing, and even everyday events like shopping or cooking, and our model can be applied to these domains as well.

The contributions of this paper are:

1. We propose a reading comprehension task which requires deep reasoning over structures that represent complex relations between multiple events and entities.
2. We present PROCESSBANK, a new dataset consisting of descriptions of biological processes, fully-annotated with rich process structures, and accompanied by multiple-choice questions.
3. We present a novel method for answering questions, by predicting process structures and mapping questions to queries. We demonstrate that by predicting structures we can improve reading comprehension accuracy over baselines that do not exploit the underlying structure.

The data and code for this paper are available at `http://www-nlp.stanford.edu/software/bioprocess`.

2 Task Definition and Setup

This section describes the reading comprehension task we address and the accompanying dataset. We will use the example in Figure 1 as our running example throughout the paper.

Our goal is to tackle a complex reading comprehension setting that centers on understanding the underlying meaning of a process description. We target a multiple-choice setting in which each input consists of a paragraph of text describing a biological process, a question, and two possible answers. The goal is to identify the correct answer using the text (Figure 1, left). We used the 148 paragraphs from the textbook *Biology* (Campbell and Reece, 2005) that were manually identified by Scaria et al. (2013). We extended this set to 200 paragraphs by including additional paragraphs that describe biological processes. Each paragraph in the collection represents a single biological process and describes a set of events, their participants and their interactions.

Because we target understanding of paragraph meaning, we use the following desiderata for building the corpus of questions and answers:

1. The questions should focus on the events and entities participating in the process described in the paragraph, and answering the questions should require reasoning about the relations between those events and entities.
2. Both answers should have similar lexical overlap with the paragraph. Moreover, names of entities and events in the question and answers should appear as in the paragraph and not using synonyms. This is to ensure that the task revolves around reading comprehension rather than lexical variability.[1]

A biologist created the question-answer part of

[1]Lexical variability is an important problem in NLP, but is not the focus of this task.

the corpus comprising of 585 questions spread over the 200 paragraphs. A second annotator validated 326 randomly chosen questions and agreed on the correct answer with the first annotator in 98.1% of cases. We provide the annotation guidelines in the supplementary material.

Figure 1 (left) shows an excerpt of a paragraph describing a process and an example of a question based on it. In general, questions test an understanding of the interactions between multiple events (such as causality, inhibition, temporal ordering), or between events and entities (i.e., roles of entities in events), and require complex reasoning about chains of event-event and event-entity relations.

3 The Structure of Processes

A natural first step for answering reading comprehension questions is to identify a structured representation of the text. In this section, we define this structure. We broadly follow the definition of Scaria et al. (2013), but modify important aspects, highlighted at the end of this section.

A paragraph describing a process is a sequence of tokens that describes events, entities and their relations (see Figure 1, top right). A *process* is a directed graph $(\mathcal{T}, \mathcal{A}, \mathcal{E}_{tt}, \mathcal{E}_{ta})$, where the nodes $\mathcal{T}$ are labeled event triggers, the nodes $\mathcal{A}$ are arguments, $\mathcal{E}_{tt}$ are labeled edges describing event-event relations, and $\mathcal{E}_{ta}$ are labeled edges from triggers to arguments denoting semantic roles (see Figure 1 top right for a partial structure of the running example). The goal of process extraction is to generate the process graph given the input paragraph.

Triggers and arguments A trigger is a token span denoting the occurrence of an event. In Figure 1, *split*, *absorbed* and *transfer* are event triggers. In rare cases, a trigger denotes the *non-occurrence* of an event. For example, in *"sympatric speciation can occur when gene flow is blocked"*, *sympatric speciation* occurs if *gene flow* does *not* happen. Thus, nodes in $\mathcal{T}$ are labeled as either a T-YES or T-NO to distinguish triggers of events that occur from triggers of events that do not occur. Arguments are token spans denoting entities that participate in the process (such as *water*, *light* and *ions* in Figure 1).

Semantic roles The edges $\mathcal{E}_{ta}$ from triggers to arguments are labeled by the semantic roles AGENT, THEME, SOURCE, DESTINATION, LOCATION, RESULT, and OTHER for all other roles. Our running example shows three THEME semantic roles for the three triggers. For brevity, the figure does not show the RESULT of the event *split*, namely, both *source of electrons and protons (hydrogen ions, H^+)* and O_2.

Event-event relations The directed edges $\mathcal{E}_{tt}$ between triggers are labeled by one of eight possible event-event relations. These relations are central to answering reading comprehension questions, which test understanding of the dependencies and causal relations between the process events. We first define three relations that express a dependency between two event triggers u and v.

1. CAUSE denotes that u starts before v, and if u happens then v happens (Figure 1).
2. ENABLE denotes that u creates conditions necessary for the occurrence of v. This means that u starts before v and v can only happen if u happens (Figure 1).[2]
3. PREVENT denotes that u starts before v and if u happens, then v does not happen.

In processes, events sometimes depend on more than one other event. For example, in Figure 1 (right top) *transfer of ions* depends on both *water splitting* as well as *light absorption*. Conversely, in Figure 2, the *shifting* event results in either one of two events but not both. To express both conjunctions and disjunctions of related events we add the relations CAUSE-OR, ENABLE-OR and PREVENT-OR, which express disjunctions, while the default CAUSE, ENABLE, and PREVENT express conjunction (Compare the CAUSE-OR relations in Figure 2 with the relations in Figure 1).

We define the SUPER relation to denote that event u is part of event v. (In Figure 2, *slippage* is a sub-event of *replication*.) Last, we use the event coreference relation SAME to denote two event mentions referring to the same event.

Notice that the assignments of relation labels interact across different pairs of events. As an example, if event u causes event v, then v can not cause u. Our inference algorithm uses such structural constraints when predicting process structure (Section 4).

[2] In this work, we do not distinguish causation from facilitation, where u can help v but is not absolutely required. We instructed the annotators to ignore the inherent uncertainty in these cases and use CAUSE.

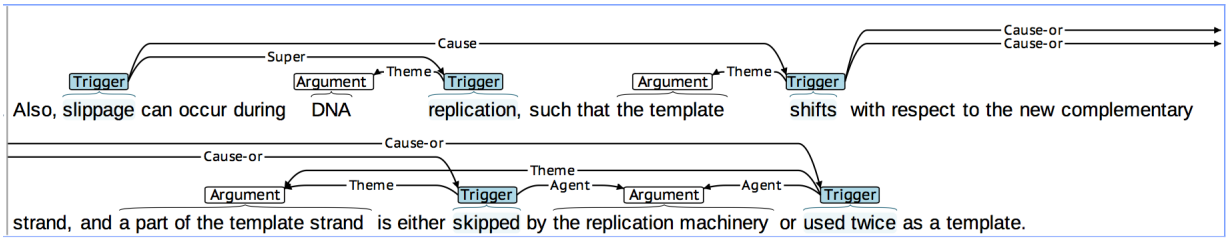


Figure 2: Partial example of a process, as annotated in our dataset.

	Avg	Min	Max
# of triggers	7.0	2	18
# of arguments	11.3	1	36
# of relation	7.9	1	37

Table 1: Statistics of triggers, arguments and relations over the 200 annotated paragraphs.

Three biologists annotated the same 200 paragraphs described in Section 2 using the brat annotation tool (Stenetorp et al., 2012). For each paragraph, one annotator annotated the process, and a second validated its correctness. Importantly, the questions and answers were authored separately by a different annotator, thus ensuring that the questions and answers are independent from the annotated structures. Table 1 gives statistics over the dataset. The annotation guidelines are included in the supplementary material.

Relation to Scaria et al. (2013) Scaria et al. (2013) also defined processes as graphs where nodes are events and edges describe event-event relations. Our definition differs in a few important aspects.

First, the set of event-event relations in that work included temporal relations in addition to causal ones. In this work, we posit that because events in a process are inter-related, causal dependencies are sufficient to capture the relevant temporal ordering between them. Figure 1 illustrates this phenomenon, where the temporal ordering between the events of *water splitting* and *light absorption* is unspecified. It does not matter whether one happens before, during, or after the other. Furthermore, the incoming causal links to *transfer* imply that the event should happen after *splitting* and *absorption*.

A second difference is that Scaria et al. (2013) do not include disjunctions and conjunctions of events in their formulation. Last, Scaria et al. (2013) predict only relations given input triggers, while we predict a full process structure.

4 Predicting Process Structures

We now describe the first step of our algorithm. Given an input paragraph we predict events, their arguments and event-event relations (Figure 1, top). We decompose this into three sub-problems:

1. Labeling trigger candidates using a multi-class classifier (Section 4.1).
2. For each trigger, identifying an over-complete set of possible arguments, using a classifier tuned for high recall (Section 4.2).
3. Jointly assigning argument labels and relation labels for all trigger pairs (Section 4.3).

The event-event relations CAUSE, ENABLE, CAUSE-OR and ENABLE-OR, form a semantic cluster: If (u, v) is labeled by one of these, then the occurrence of v depends on the occurrence of u. Since our dataset is small, we share statistics by collapsing all four labels to a single ENABLE label. Similarly, we collapse the PREVENT and PREVENT-OR labels, overall reducing the number of relations to four.

For brevity, in what follows we only provide a flavor of the features we extract, and refer the reader to the supplementary material for details.

4.1 Predicting Event Triggers

The first step is to identify the events in the process. We model the trigger detector as a multi-class classifier that labels all content words in the paragraph as one of T-YES, T-NO or NOT-TRIGGER (Recall that a word can trigger an event that occurred, an event that did not occur, or not be a trigger at all). For simplicity, we model triggers as single words, but in the gold annotation about 14% are phrases (such as *gene flow*). Thus, we evaluate trigger prediction by taking heads of gold phrases. To train the classifier, we extract

the lemma and POS tag of the word and adjacent words, dependency path to the root, POS tag of children and parent in the dependency tree, and clustering features from WordNet (Fellbaum, 1998), Nomlex (Macleod et al., 1998), Levin verb classes (Levin, 1993), and a list of biological processes compiled from Wikipedia.

4.2 Filtering Argument Candidates

Labeling trigger-argument edges is similar to semantic role labeling. Following the standard approach (Punyakanok et al., 2008), for each trigger we collect all constituents in the same sentence to build an over-complete set of plausible candidate arguments. This set is pruned with a binary classifier that is tuned for high recall (akin to the argument identifier in SRL systems). On the development set we filter more than half of the argument candidates, while achieving more than 99% recall. This classifier is trained using argument identification features from Punyakanok et al. (2008).

At the end of this step, each trigger has a set of candidate arguments which will be labeled during joint inference. In further discussion, the argument candidates for trigger t are denoted by $\mathcal{A}_t$.

4.3 Predicting Arguments and Relations

Given the output of the trigger classifier, our goal is to jointly predict event-argument and event-event relations. We model this as an integer linear program (ILP) instance described below. We first describe the inference setup assuming a model that scores inference decisions and defer description of learning to Section 4.4. The ILP has two types of decision variables: arguments and relations.

Argument variables These variables capture the decision that a candidate argument a, belonging to the set $\mathcal{A}_t$ of argument candidates, takes a label A (from Section 3). We denote the Boolean variables by $y_{t,a,A}$, which are assigned a score $b_{t,a,A}$ by the model. We include an additional label NULL-ARG, indicating that the candidate is not an argument for the trigger.

Event-event relation variables These variables capture the decision that a pair of triggers t_1 and t_2 are connected by a directed edge (t_1, t_2) labeled by the relation R. We denote these variables by $z_{t_1,t_2,R}$, which are associated with a score $c_{t_1,t_2,R}$. Again, we introduce a label NULL-REL to indicate triggers that are not connected by an edge.

Name	Description
Unique labels	Every argument candidate and trigger pair has exactly one label.
Argument overlap	Two arguments of the same trigger cannot overlap.
Relation symmetry	The SAME relation is symmetric. All other relations are anti-symmetric, i.e., for any relation label other than SAME, at most one of (t_i, t_j) or (t_j, t_i) can take that label and the other is assigned the label NULL-REL.
Max arguments per trigger	Every trigger can have no more than two arguments with the same label.
Max triggers per argument	The same span of text can not be an argument for more than two triggers.
Connectivity	The triggers must form a connected graph, framed as flow constraints as in Magnanti and Wolsey (1995) and Martins et al. (2009).
Shared arguments	If the same span of text is an argument of two triggers, then the triggers must be connected by a relation that is not NULL-REL. This ensures that triggers that share arguments are related.
Unique parent	For any trigger, at most one outgoing edge can be labeled SUPER.

Table 2: Constraints for joint inference.

Formulation Given the two sets of variables, the objective of inference is to find a global assignment that maximizes the score. That is, the objective can be stated as follows:

$$\max_{\mathbf{y},\mathbf{z}} \sum_{t,a\in\mathcal{A}_t,A} b_{t,a,A} \cdot y_{t,a,A} + \sum_{t_1,t_2,R} c_{t_1,t_2,R} \cdot z_{t_1,t_2,R}$$

Here, $\mathbf{y}$ and $\mathbf{z}$ refer to all the argument and relation variables respectively.

Clearly, all possible assignments to the inference variables are not feasible and there are both structural as well as prior knowledge constraints over the output space. Table 2 states the constraints we include, which are expressed as linear inequalities over output variables using standard techniques (e.g., (Roth and Yih, 2004)).

4.4 Learning in the Joint Model

We train both the trigger classifier and the argument identifier using L_2-regularized logistic regression. For the joint model, we use a linear model for the scoring functions, and train jointly using the structured averaged perceptron algorithm (Collins, 2002).

Since argument labeling is similar to semantic role labeling (SRL), we extract standard SRL features given the trigger and argument from the syntactic tree for the corresponding sentence. In addition, we add features extracted from an off-the-shelf SRL system. We also include all feature conjunctions. For event relations, we include the features described in Scaria et al. (2013), as well as context features for both triggers, and the dependency path between them, if one exists.

5 Question Answering via Structures

This section describes our question answering system that, given a process structure, a question and two answers, chooses the correct answer (steps 2 and 3 in Figure 1).

Our strategy is to treat the process structure as a small knowledge-base. We map each answer along with the question into a *structured query* that we compare against the structure. The query can prove either the correctness or incorrectness of the answer being considered. That is, either we get a valid match for an answer (proving that the corresponding answer is correct), or we get a refutation in the form of a contradicted causal chain (thus proving that the *other answer* is correct). This is similar to theorem proving approaches suggested in the past for factoid question answering (Moldovan et al., 2003).

The rest of this section is divided into three parts: Section 5.1 defines the queries we use, Section 5.2 describes a rule-based algorithm for converting a question and an answer into a query and finally, 5.3 describes the overall algorithm.

5.1 Queries over Processes

We model a query as a directed graph path with regular expressions over edge labels. The bottom right portion of Figure 1 shows examples of queries for our running example. In general, given a question and one of the answer candidates, one end of the path is populated by a trigger/argument found in the question and the other is populated with a trigger/ argument from the answer.

We define a query to consist of three parts:

1. A regular expression over relation labels, describing permissible paths,
2. A source trigger/argument node, and
3. A target trigger/argument node.

For example, the bottom query in Figure 1 looks for paths labeled with CAUSE or ENABLE edges from the event *split* to the event *transfer*.

Note that the representation of questions as directed paths is a modeling choice and did not influence the authoring of the questions. Indeed, while most questions do fit this model, there are rare cases that require a more complex query structure.

5.2 Query Generation

Mapping a question and an answer into a query involves identifying the components of the query listed above. We do this in two phases: (1) In the *alignment phase*, we align triggers and arguments in the question and answer to the process structure to give us candidate source and target nodes. (2) In the *query construction phase*, we identify the regular expression and the direction of the query using the question, the answer and the alignment.

We identify three broad categories of QA pairs (see Table 3) that can be identified using simple lexical rules: (a) *Dependency questions* ask which event or argument depends on another event or argument, (b) *Temporal questions* ask about temporal ordering of events, and (c) *True-false questions* ask whether some fact is true. Below, we describe the two phases of query generation primarily in the context of dependency questions with a brief discussion about temporal and true-false questions at the end of the section.

Alignment Phase We align triggers in the structure to the question and the answer by matching lemmas or nominalizations. In case of multiple matches, we use the context to disambiguate and resolve ties using the highest matching candidate in the syntactic dependency tree.

We align arguments in the question and the answer in a similar manner. Since arguments are typically several words long, we prefer maximal spans. Additionally, if a question (or an answer) contains an aligned trigger, we prefer to align words to its arguments.

Query Construction Phase We construct a query using the aligned question and answer triggers/arguments. We will explain query construction using our running example (reproduced as the dependency question in Table 3).

First, we identify the source and the target of the query. We select either the source or the target to be a question node and populate the other end of the query path with an answer node. To make the choice between source or target for the question node, we use the main verb in the question, its voice and relative position of the question word with respect to the main verb. In our example, the main verb *lead to* is in active voice and the question word *what* is not in subject position. This places the trigger from the question as the source of the query path (see both queries in the bottom right portion of the running example). In contrast, had the verb been *require*, the trigger would be the target of the query. We construct two verb clusters that indicate query direction using a small seed set

Type	Example	# (%)
Dependency	**Q**: What can the splitting of water lead to? **a**: Light absorption **b**: Transfer of ions	407 (69.57%)
Temporal	**Q**: What is the correct order of events? **a**: PDGF binds to tyrosine kinases, then cells divide, then wound healing **b**: Cells divide, then PDGF binds to tyrosine kinases, then wound healing	57 (9.74%)
True-False	**Q**: Cdk associates with MPF to become cyclin **a**: True **b**: False	121 (20.68%)

Table 3: Examples and statistics for each of the three coarse types of questions.

Is main verb trigger?

Yes

Condition	Regular Exp.
Wh- word subjective?	AGENT
Wh- word object?	THEME

No

Condition	Regular Exp.
default	(ENABLE|SUPER)$^+$
DIRECT	(ENABLE|SUPER)
PREVENT	(ENABLE|SUPER)* PREVENT(ENABLE|SUPER)*

Figure 3: Rules for determining the regular expressions for queries concerning two triggers. In each table, the **condition** column decides the regular expression to be chosen. In the left table, we make the choice based on the path from the root to the *Wh-* word in the question. In the right table, if the word *directly* modifies the main trigger, the DIRECT regular expression is chosen. If the main verb in the question is in the synset of *prevent*, *inhibit*, *stop* or *prohibit*, we select the PREVENT regular expression. Otherwise, the default one is chosen. We omit the relation label SAME from the expressions, but allow going through any number of edges labeled by SAME when matching expressions to the structure.

that we expand using WordNet.

The final step in constructing the query is to identify the regular expression for the path connecting the source and the target. Due to paucity of data, we do not map a question and an answer to arbitrary regular expressions. Instead, we construct a small set of regular expressions, and build a rule-based system that selects one. We used the training set to construct the regular expressions and we found that they answer most questions (see Section 6.4). We determine the regular expression based on whether the main verb in the sentence is a trigger and whether the source and target of the path are triggers or arguments. Figure 3 shows the possible regular expressions and the procedure for choosing one when both the source and target are triggers. If either of them are argument nodes, we append the appropriate semantic role to the regular expression, based on whether the argument is the source or the target of the path (or both).

True-false questions are treated similarly, except that both source and target are chosen from the question. For temporal questions, we seek to identify the ordering of events in the answers. We use the keywords *first*, *then*, or *simultaneously* to identify the implied order in the answer. We use the regular expression SUPER$^+$ for questions asking about simultaneous events and ENABLE$^+$ for those asking about sequential events.

5.3 Answering Questions

We match the query of an answer to the process structure to identify the answer. In case of a match, the corresponding answer is chosen. The matching path can be thought of as a *proof* for the answer.

If neither query matches the graph (or both do), we check if either answer contradicts the structure. To do so, we find an undirected path from the source to the target. In the event of a match, if the matching path traverses any ENABLE edge in the incorrect direction, we treat this as a *refutation* for the corresponding answer and select the other one. In our running example, in addition to the valid path for the second query, for the first query we see that there is an undirected path from *split* to *absorb* through *transfer* that matches the first query. This tells us that *light absorption* cannot be the answer because it is not along a causal path from *split*.

Finally, if none of the queries results in a match, we look for any unlabeled path between the source and the target, before backing off to a dependency-based proximity baseline described in Section 6. When there are multiple aligning nodes in the question and answer, we look for any proof or refutation before backing off to the baselines.

6 Empirical Evaluation

In this section we aim to empirically evaluate whether we can improve reading comprehension accuracy by predicting process structures. We first provide details of the experimental setup.

6.1 Experimental setup

We used 150 processes (435 questions) for training and 50 processes (150 questions) as the test set. For development, we randomly split the training set 10 times (80%/20%), and tuned hyper-parameters by maximizing average accuracy on question answering. We preprocessed the paragraphs with the Stanford CoreNLP pipeline version 3.4 (Manning et al., 2014) and Illinois SRL (Punyakanok et al., 2008; Clarke et al., 2012). We used the Gurobi optimization package[3] for inference.

We compare our system PROREAD to baselines that do not have access to the process structure:

1. BOW: For each answer, we compute the proportion of content word lemmas covered by the paragraph and choose the one with higher coverage. For true-false questions, we compute the coverage of the question statement, and answer "True" if it is higher than a threshold tuned on the development set.
2. TEXTPROX: For dependency questions, we align content word lemmas in both the question and answer against the text and select the answer whose aligned tokens are closer to the aligned tokens of the question. For temporal questions, we return the answer for which the order of events is identical to their order in the paragraph. For true-false questions, we return "True" if the number of bigrams from the question covered in the text is higher than a threshold tuned on the development set.
3. SYNTPROX: For dependency questions, we use proximity as in TEXTPROX, except that distance is measured using dependency tree edges. To support multiple sentences we connect roots of adjacent sentences with bidirectional edges. For temporal questions this baseline is identical to TEXTPROX. For true-false questions, we compute the number of dependency tree edges in the question statement covered by edges in the paragraph (an edge has a source lemma, relation, and target lemma), and answer "True" if the coverage is higher than a threshold tuned on the training set.

To separate the contribution of process structures from the performance of our structure predictor, we also run our QA system given manually annotated gold standard structures (GOLD).[4]

Method	Depen.	Temp.	True-false	All
PROREAD	**68.1**	**80.0**	**55.6**	**66.7**
SYNTPROX	61.9	70.0	48.1	60.0
TEXTPROX	58.4	70.0	33.3	54.7
BOW	47.8	40.0	44.4	46.7
GOLD	77.9	80.0	70.4	76.7

Table 4: Reading comprehension test set accuracy. The **All** column shows overall accuracy across all questions. The first three columns show accuracy for each coarse type.

6.2 Reading Comprehension Task

We evaluate our system using accuracy, i.e., the proportion of questions answered correctly. Table 4 presents test set results, where we break down questions by their coarse-type.

PROREAD improves accuracy compared to the best baseline by 6.7 absolute points (last column). Most of the gain is due to improvement on dependency questions, which are the most common question type. The performance of BOW indicates that lexical coverage alone does not distinguish the correct answer from the wrong answer. In fact, guessing the answer with higher lexical overlap results in performance that is slightly lower than random. Text proximity and syntactic proximity provide a stronger cue, but exploiting predicted process structures substantially outperforms these baselines.

Examining results using gold information highlights the importance of process structures independently of the structure predictor. Results of GOLD demonstrate that given gold structures we can obtain a dramatic improvement of almost 17 points compared to the baselines, using our simple deterministic QA system.

Results on true-false questions are low for PROREAD and all the baselines. True-false questions are harder for two main reasons. First, in dependency and temporal questions, we create a query for both answers, and can find a proof or a refutation for either one of them. In true-false

[3] http://www.gurobi.com/

[4] We also ran an experiment where gold triggers are given and arguments and relations are predicted. We found that this results in slightly higher performance compared to PROREAD.

	Precision	Recall	F_1
Triggers	75.4	73.9	74.6
Arguments	43.4	34.4	38.3
Relations	27.0	22.5	24.6

Table 5: Structured prediction test set results.

questions we must determine given a single statement whether it holds. Second, an analysis of true-false questions reveals that they focus less on relations between events and entities in the process, and require modeling lexical variability.[5]

6.3 Structure Prediction Task

Our evaluation demonstrates that gold structures improve accuracy substantially more than predicted structures. To examine this, we now directly evaluate the structure predictor by comparing micro-average precision, recall and F_1 between predicted and gold structures (Table 5).

While performance for trigger identification is reasonable, performance on argument and relation prediction is low. This explains the higher performance obtained in reading comprehension given gold structures. Note that errors in trigger prediction propagate to argument and relation prediction – a relation cannot be predicted correctly if either one of the related triggers is not previously identified. One reason for low performance is the small size of the dataset. Thus, training process predictors with less supervision is an important direction for future work. Furthermore, the task of process prediction is inherently difficult, because often relations are expressed only indirectly in text. For example, in Figure 1 the relation between *water splitting* and *transfer of ions* is only recoverable by understanding that water provides the ions that need to be transferred.

Nevertheless, we find that questions can often be answered correctly even if the structure contains some errors. For example, the gold structure for the sentence *"Some ... radioisotopes have long half-lives, allowing ..."*, contains the trigger *long half-lives*, while we predict *have* as a trigger and *long half-lives* as an argument. This is good enough to answer questions related to this part of the structure correctly, and overall, to improve performance using predicted structures.

[5]The low performance of TEXTPROX and SYNTPROX on true-false questions can also be attributed to the fact that we tuned a threshold parameter on the training set, and this did not generalize well to the test set.

Reason	GOLD	PROREAD
Alignment	35%	15%
Missing from annotation	25%	10%
Entity coreference	20%	10%
Missing regular expression	10%	
Lexical variability	5%	10%
Error in predicted structure		55%
Other	5%	

Table 6: Error analysis results. An explanation of the various categories are in the body of the paper.

6.4 Error Analysis

This section presents the results of an analysis of 20 sampled errors of GOLD (gold structures), and 20 errors of PROREAD (predicted structures). We have categorized the primary reason for error in Table 6.

As expected, the main problem when using predicted structures, is structure errors which account for more than half of the errors.

Errors in GOLD are distributed across various categories, which we briefly describe. *Alignment* errors occur due to multiple words aligning to multiple triggers and arguments. For example, in the question *"What is the result of gases being produced in the lysosome?"*, the answer *"engulfed pathogens are poisoned"* is incorrectly aligned to the trigger *engulfed* rather than to *poisoned*.

Another reason for errors are cases where questions are asked about parts of the paragraph that are *missing from annotation*. This is possible since questions were authored independently of structure annotation. Two other causes for errors are *entity coreference* errors, where a referent for an entity is missing from the structure, and *lexical variability*, where the author of questions uses names for triggers or arguments that are missing from the paragraph, and so alignment fails.

Last, in 10% of the cases in GOLD we found that the answer could not be retrieved using the set of regular expressions that are currently used by our QA system.

7 Discussion

This work touches on several strands of work in NLP including information extraction, semantic role labeling, semantic parsing and reading comprehension.

Event and relation extraction have been studied via the ACE data (Doddington et al., 2004) and related work. The BioNLP shared tasks (Kim et al., 2009; Kim et al., 2011; Riedel and McCal-

lum, 2011) focused on biomedical data to extract events and their arguments. Event-event relations have been mostly studied from the perspective of temporal ordering; e.g., (Chambers and Jurafsky, 2008; Yoshikawa et al., 2009; Do et al., 2012; McClosky and Manning, 2012). The process structure predicted in this work differs from these lines of work in two important ways: First, we predict events, arguments *and* their interactions from multiple sentences, while most earlier work focused on one or two of these components. Second, we model processes, and thus target causal relations between events, rather than temporal order only.

Our semantic role annotation is similar to existing SRL schemes such as PropBank (Palmer et al., 2005), FrameNet (Ruppenhofer et al., 2006) and BioProp (Chou et al., 2006). However, in contrast to PropBank and FrameNet, we do not allow all verbs to trigger events and instead let the annotators decide on biologically important triggers, which are not restricted to verbs (unlike BioProp, where 30 pre-specified verbs were selected for annotation). Like PropBank and BioProp, the argument labels are not trigger specific.

Mapping questions to queries is effectively a semantic parsing task. In recent years, several lines of work addressed semantic parsing using various formalisms and levels of supervision (Zettlemoyer and Collins, 2005; Wong and Mooney, 2006; Clarke et al., 2010; Berant et al., 2013). In particular, Krishnamurthy and Kollar (2013) learned to map natural language utterances to referents in an image by constructing a KB from the image and then mapping the utterance to a query over the KB. This is analogous to our process of constructing a process structure and performing QA by querying that structure. In our work, we parse questions into graph-based queries, suitable for modeling processes, using a rule-based heuristic. Training a statistical semantic parser that will replace the QA system is an interesting direction for future research.

Multiple choice reading comprehension tests are a natural choice for evaluating machine reading. Hirschman et al. (1999) presented a bag-of-words approach to retrieving sentences for reading comprehension. Richardson et al. (2013) recently released the MCTest reading comprehension dataset that examines understanding of fictional stories. Their work shares our goal of advancing micro-reading, but they do not focus on process understanding.

Developing programs that perform deep reasoning over complex descriptions of processes is an important step on the road to fulfilling the higher goals of machine reading. In this paper, we present an end-to-end system for reading comprehension of paragraphs which describe biological processes. This is, to the best of our knowledge, the first system to both predict a rich structured representation that includes entities, events and their relations, *and* utilize this structure for answering reading comprehension questions. We also created a new dataset, PROCESSBANK, which contains 200 paragraphs that are both fully-annotated with process structure, as well as accompanied by questions. We empirically demonstrated that modeling biological processes can substantially improve reading comprehension accuracy in this domain.

Acknowledgments

The authors would like to thank Luke Amuchastegui for authoring the multiple-choice questions, and also the anonymous reviewers for their constructive feedback. We thank the Allen Institute for Artificial Intelligence for assistance in funding this work.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of EMNLP*.

Neil Campbell and Jane Reece. 2005. *Biology*. Benjamin Cummings.

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

Nathanael Chambers and Daniel Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of EMNLP*.

Wen-Chi Chou, Richard Tzong-Han Tsai, Ying-Shan Su, Wei Ku, Ting-Yi Sung, and Wen-Lian Hsu. 2006. A semi-automatic method for annotating a biomedical proposition bank. In *Proceedings of the Workshop on Frontiers in Linguistically Annotated Corpora 2006*, July.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of CoNLL*.

James Clarke, Vivek Srikumar, Mark Sammons, and Dan Roth. 2012. An NLP Curator (or: How I

Learned to Stop Worrying and Love NLP Pipelines). In *Proceedings of LREC*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of ACL*.

Quang Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of EMNLP-CoNLL*.

George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie Strassel, and Ralph M. Weischedel. 2004. The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation. In *Proceedings of LREC*.

Oren Etzioni, Michele Banko, and Michael J. Cafarella. 2006. Machine reading. In *Proceedings of AAAI*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of EMNLP*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-Driven Learning for Open Question Answering. In *Proceedings of ACL*.

Christiane Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT Press.

Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep read: A reading comprehension system. In *Proceedings of ACL*.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Junichi Tsujii. 2009. Overview of BioNLP 09 shared task on event extraction. In *Proceedings of BioNLP*.

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, and Junichi Tsujii. 2011. Overview of BioNLP shared task 2011. In *Proceedings of BioNLP*.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *TACL*, 1:193–206.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of ACL*.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago Press.

Catherine Macleod, Ralph Grishman, Adam Meyers, Leslie Barrett, and Ruth Reeves. 1998. Nomlex: A lexicon of nominalizations. In *Proceedings of EURALEX*.

Thomas L. Magnanti and Laurence A. Wolsey. 1995. Optimal trees. *Handbooks in operations research and management science*, 7:503–615.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL: System Demonstrations*.

André L. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL/IJCNLP*.

David McClosky and Christopher D. Manning. 2012. Learning constraints for consistent timeline extraction. In *Proceedings of EMNLP-CoNLL*.

Dan Moldovan, Christine Clark, Sanda Harabagiu, and Steve Maiorano. 2003. Cogex: A logic prover for question answering. In *Proceedings of NAACL-HLT*.

Martha Palmer, Paul Kingsbury, and Daniel Gildea. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2).

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of EMNLP*.

Sebastian Riedel and Andrew McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *Proceedings of EMNLP*.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNLL*.

Josef Ruppenhofer, Michael Ellsworth, Miriam RL Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2006. FrameNet II: Extended theory and practice. *Berkeley FrameNet Release*, 1.

Aju Thalappillil Scaria, Jonathan Berant, Mengqiu Wang, Peter Clark, Justin Lewis, Brittany Harding, and Christopher D. Manning. 2013. Learning biological processes with global constraints. In *Proceedings of EMNLP*.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the demonstrations at EACL*.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of HLT-NAACL*.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with Markov logic. In *Proceedings of ACL/IJCNLP*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI*.

Sensicon: An Automatically Constructed Sensorial Lexicon

Serra Sinem Tekiroğlu
University of Trento
Fondazione Bruno Kessler
Trento, Italy
tekiroglu@fbk.eu

Gözde Özbal
Trento RISE
Trento, Italy
gozbalde@gmail.com

Carlo Strapparava
Fondazione Bruno Kessler
Trento, Italy
strappa@fbk.eu

Abstract

Connecting words with senses, namely, sight, hearing, taste, smell and touch, to comprehend the sensorial information in language is a straightforward task for humans by using commonsense knowledge. With this in mind, a lexicon associating words with senses would be crucial for the computational tasks aiming at interpretation of language. However, to the best of our knowledge, there is no systematic attempt in the literature to build such a resource. In this paper, we present a sensorial lexicon that associates English words with senses. To obtain this resource, we apply a computational method based on bootstrapping and corpus statistics. The quality of the resulting lexicon is evaluated with a gold standard created via crowdsourcing. The results show that a simple classifier relying on the lexicon outperforms two baselines on a sensory classification task, both at word and sentence level, and confirm the soundness of the proposed approach for the construction of the lexicon and the usefulness of the resource for computational applications.

1 Introduction

Sensorial information interpenetrates languages with various semantic roles in different levels since the main interaction instrument of humans with the outside world is the sensory organs. The transformation of the raw sensations that we receive through the sensory organs into our understanding of the world has been an important philosophical topic for centuries. According to a classification that dates back to Aristotle (Johansen, 1997), senses can be categorized into five modalities, namely, sight, hearing, taste, smell and touch. With the help of perception, we can process the data coming from our sensory receptors and become aware of our environment. While interpreting sensory data, we unconsciously use our existing knowledge and experience about the world to create a private experience (Bernstein, 2010).

Language has a significant role as our main communication device to convert our private experiences to shared representations of the environment that we perceive (Majid and Levinson, 2011). As a basic example, onomatopoeic words, such as *knock* or *woof*, are acquired by direct imitation of the sounds allowing us to share the experience of what we hear. As another example, where an imitation is not possible, is that giving a name to a color, such as *blue*, provides a tool to describe a visual feature of an object. In addition to the words that describe the direct sensorial features of objects, languages include many other lexical items that are connected to sensory modalities in various semantic roles. For instance, while some words can be used to describe a perception activity (e.g., *to sniff*, *to watch*, *to feel*), others can simply be physical phenomena that can be perceived by sensory receptors (e.g., *light*, *song*, *salt*, *smoke*).

Common usage of language, either written or spoken, can be very dense in terms of sensorial words. As an example, the sentence "*I felt the cold breeze.*" contains three sensorial words: *to feel* as a perception activity, *cold* as a perceived sensorial feature and *breeze* as a physical phenomenon. The connection to the sense modalities of the words might not be mutually exclusive, that is to say a word can be associated with more than one senses. For instance, the adjective *sweet* could be associated with both the senses of *taste* and *smell*. While we, as humans, have the ability to connect words with senses intuitively by using our commonsense knowledge, it is not straightforward for machines to interpret sensorial information.

Making use of a lexicon containing sensorial words could be beneficial for many computational scenarios. Rodriguez-Esteban and Rzhetsky

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1511–1521,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

(2008) report that using words related to senses in a text could clarify the meaning of an abstract concept by facilitating a more concrete imagination. To this respect, an existing text could be automatically modified with sensory words for various purposes such as attracting attention or biasing the audience towards a specific concept. Additionally, sensory words can be utilized to affect private psychology by inducing a positive or negative sentiment (Majid and Levinson, 2011). For instance, de Araujo et al. (2005) show that the pleasantness level of the same odor can be altered by labeling it as *body odor* or *cheddar cheese*. As another motivation, the readability and understandability of text could also be enhanced by using sensory words (Rodriguez-Esteban and Rzhetsky, 2008). A compelling use case of a sensorial lexicon is that automatic text modification to change the density of a specific sense could help people with sensory disabilities. For instance, while teaching a concept to a congenitally blind child, an application that eliminates color-related descriptions would be beneficial. A sensorial lexicon could also be exploited by search engines to personalize the results according to user needs.

Advertising is another broad area which would benefit from such a resource especially by using synaesthesia[1], as it strengthens creative thinking and it is commonly exploited as an imagination boosting tool in advertisement slogans (Pricken, 2008). As an example, we can consider the slogans "*The taste of a paradise*" where the sense of sight is combined with the sense of taste or "*Hear the big picture*" where *sight* and *hearing* are merged.

Various studies have been conducted both in computational linguistics and cognitive science that build resources associating words with several cognitive features such as abstractness-concreteness (Coltheart, 1981; Turney et al., 2011), emotions (Strapparava and Valitutti, 2004; Mohammad and Turney, 2010), colors (Özbal et al., 2011; Mohammad, 2011) and imageability (Coltheart, 1981). However, to the best of our knowledge, there is no attempt in the literature to build a resource that associates words with senses. In this paper, we propose a computational method to automatically generate a sensorial lexicon that associates words in English with senses. Our method consists of two main steps. First, we generate a set of seed words for each sense category with the help of a bootstrapping approach. In the second step, we exploit a corpus based probabilistic technique to create the final lexicon. We evaluate this lexicon with the help of a gold standard that we obtain by using the crowdsourcing service of CrowdFlower[2].

The sensorial lexicon, which we named Sensicon, embodies 22,684 English lemmas together with their part-of-speech (POS) information that have been linked to one or more of the five senses. Each entry in this lexicon consists of a lemma-POS pair and a score for each sensory modality that indicates the degree of association. For instance, the verb *stink* has the highest score for *smell* as expected while the scores for the other four senses are very low. The noun *tree*, which is a concrete object and might be perceived by multiple senses, has high scores for *sight*, *touch* and *smell*.

The rest of the paper is organized as follows. We first review previous work relevant to this task in Section 2. Then in Section 3, we describe the proposed approach in detail. In Section 4, we explain the annotation process that we conducted and the evaluation strategy that we employed. Finally, in Section 5, we draw our conclusions and outline possible future directions.

2 Related Work

Since to the best of our knowledge there is no attempt in the literature to automatically associate words with human senses, in this section we will summarize the most relevant studies that focused on linking words with various other cognitive features.

There are several studies focusing on word-emotion associations. WordNet Affect Lexicon (Strapparava and Valitutti, 2004) maps WordNet (Fellbaum, 1998) synsets to various cognitive features (e.g., emotion, mood, behaviour). This resource is created by using a small set of synsets as seeds and expanding them with the help of semantic and lexical relations among these synsets. Yang et al. (2007) propose a collocation model with emoticons instead of seed words while creating an emotion lexicon from a corpus. Perrie et al. (2013) build a word-emotion association lexicon by using subsets of a human-annotated lexicon as seed sets. The authors use frequencies, counts, or unique seed words extracted from an n-gram corpus to create lexicons in different sizes. They pro-

[1] American Heritage Dictionary (`http://ahdictionary.com/`) defines synaesthesia in linguistics as the description of one kind of sense impression by using words that normally describe another.

[2] `http://www.crowdflower.com/`

pose that larger lexicons with less accurate generation method perform better than the smaller human annotated lexicons. While a major drawback of manually generated lexicons is that they require a great deal of human labor, crowdsourcing services provide an easier procedure for manual annotations. Mohammad and Turney (2010) generate an emotion lexicon by using the crowdsourcing service provided by Amazon Mechanical Turk[3] and it covers 14,200 term-emotion associations.

Regarding the sentiment orientations and subjectivity levels of words, Sentiwordnet (Esuli and Sebastiani, 2006) is constructed as an extension to WordNet and it provides sentiments in synset level. Positive, negative and neutral values are assigned to synsets by using ternary classifiers and synset glosses. Another study that has been inspirational for the design of our approach is Banea et al. (2008). The authors generate a subjectivity lexicon starting with a set of seed words and then using a similarity measure among the seeds and the candidate words.

Another cognitive feature relevant to sensorial load of the words is the association between colors and words. Mohammad (2011) builds a color-word association lexicon by organizing a crowdsourcing task on Amazon Mechanical Turk. Instead, Özbal et al. (2011) aim to automate this process and propose three computational methods based on image analysis, language models and latent semantic analysis (LSA) (Landauer and Dumais, 1997). The authors compare these methods against a gold standard obtained by the crowdsourcing service of Amazon Mechanical Turk. The best performance is obtained by using image features while LSA performs slightly better than the baseline.

Finally, there have been efforts in the literature about the association of words with their abstractness-concreteness and imageability levels. MRC Psycholinguistic Database (Coltheart, 1981) includes abstractness-concreteness and imageability ratings of a small set of words determined according to psycholinguistic experiments. Turney et al. (2011) propose to use LSA similarities of words with a set of seed words to automatically calculate the abstractness and concreteness degrees of words.

[3] http://www.mturk.com/mturk

3 Automatic Association of Senses with Words

We adopt a two phased computational approach to construct a large sensorial lexicon. First, we employ a bootstrapping strategy to generate a sufficient number of sensory seed words from a small set of manually selected seed words. In the second phase, we perform a corpus based probabilistic method to estimate the association scores to build a larger lexicon.

3.1 Selecting Seed Words

The first phase of the lexicon construction process aims to collect *sensorial seed words*, which are directly related to senses (e.g., *sound*, *tasty* and *sightedness*). To achieve that, we utilized a lexical database called FrameNet (Baker et al., 1998), which is built upon *semantic frame*s of concepts in English and lexical units (i.e., words) that evoke these frames. The basic idea behind this resource is that meanings of words can be understood on the basis of a semantic frame. A semantic frame consists of semantic roles called frame elements, which are manually annotated in more than 170,000 sentences. We have considered FrameNet to be especially suitable for the collection of sensorial seed words since it includes semantic roles and syntactic features of sensational and perceptional concepts.

In order to determine the seed lemma-POS pairs in FrameNet, we first manually determined 31 frames that we found to be highly connected to senses such as *Hear*, *Color*, *Temperature* and *Perception_experience*. Then, we conducted an annotation task and asked 3 annotators to determine which senses the lemma-POS pairs evoking the collected frames are associated with. At the end of this task, we collected all the pairs (i.e. 277) with 100% agreement to constitute our initial seed set. This set contains 277 lemma-POS pairs associated with a specific sense such as the verb *click* with *hearing*, the noun *glitter* with *sight* and *aromatic* with *smell*.

3.2 Seed Expansion via Bootstrapping

In this step, we aim to extend the seed list that we obtained from FrameNet with the help of a bootstrapping approach. To achieve that, we adopt a similar approach to Dias et al. (2014), who propose a repetitive semantic expansion model to automatically build temporal associations of synsets in WordNet. Figure 1 provides an overview of the bootstrapping process. At each iteration, we

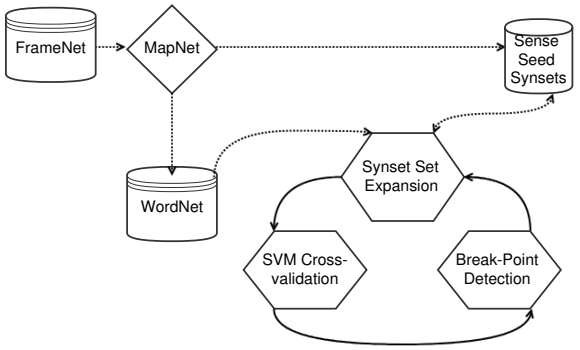


Figure 1: Bootstrapping procedure to expand the seed list.

first expand the seed list by using semantic relations provided by WordNet. We then evaluate the accuracy of the new seed list for sense classification by means of cross-validation against WordNet glosses. For each sense, we continue iterating until the cross-validation accuracy becomes stable or starts to decrease. The following sections explain the whole process in detail.

3.2.1 Extending the Seed List with WordNet

While the initial sensory seed list obtained from FrameNet contains only 277 lemma-POS pairs, we extend this list by utilizing the semantic relations provided by WordNet. To achieve that, we first map each lemma-POS pair in the seed list to WordNet synsets with the help of MapNet (Tonelli and Pighin, 2009), which is a resource providing direct mapping between WordNet synsets and FrameNet lexical units. Then, we add to the list the synsets that have WordNet relations *direct antonymy*, *similarity*, *derived-from*, *derivationally-related*, *pertains-to*, *attribute* and *also-see* with the already existing seeds. For instance, we add the synset containing the verb *laugh* for the synset of the verb *cry* with the relation *direct antonymy*, or the synset containing the adjective *chilly* for the synset of the adjective *cold* with the relation *similarity*. We prefer to use these relations as they might allow us to preserve the semantic information as much as possible during the extension process. It is worth mentioning that these relations were also found to be appropriate for preserving the affective connotation by Valitutti et al. (2004). Additionally, we use the relations *hyponym* and *hyponym-instance* to enrich the seed set with semantically more specific synsets. For instance, for the noun seed *smell*, we expand the list with the hyponyms of its synset such as the nouns *bouquet*, *fragrance*, *fragrancy*, *redolence* and *sweetness*.

3.2.2 Cross-validation of Sensorial Model

After obtaining new synsets with the help of WordNet relations in each bootstrapping cycle, we build a five-class sense classifier over the seed synsets defined by their glosses provided in WordNet. Similarly to Dias et al. (2014), we assume that the sense information of sensorial synsets is preserved in their definitions. Accordingly, we employ a support vector machine (SVM) (Boser et al., 1992; Vapnik, 1998) model with second degree polynomial kernel by representing the gloss of each synset as a vector of lemmas weighted by their counts. For each synset, its gloss is lemmatized by using Stanford Core NLP[4] and cleaned from the stop words. After each iteration cycle, we perform a 10-fold cross-validation in the updated seed list to detect the accuracy of the new sensorial model. For each sense class, we continue iterating and thereby expanding the seed list until the classifier accuracy steadily drops.

Table 1 lists the *precision* (*P*), *recall* (*R*) and *F1* values obtained for each sense after each iteration until the bootstrapping mechanism stops. While the iteration number is provided in the first column, the values under the last column group present the micro-average of the resulting multi-class classifier. The change in the performance values of each class in each iteration reveals that the number of iterations required to obtain the seed lists varies for each sense. For instance, the F1 value of *touch* continues to increase until the fourth cycle whereas *hearing* records a sharp decrease after the first iteration.

After the bootstrapping process, we create the final lexicon by repeating the expansion for each class until the optimal number of iterations is reached. The last row of Table 1, labeled as *Final*, demonstrates the accuracy of the classifier trained and tested on the final lexicon, i.e., using the seeds selected after iteration 2 for *Sight*, iteration 1 for *Hearing*, iteration 3 for *Taste* and *Smell* and iteration 4 for *Touch*. According to F1 measurements of each iteration, while *hearing* and *taste* have a lower value for the final model, *sight*, *smell* and *touch* have higher results. It should also be noted that the micro-average of the F1 values of the final model shows an increase when compared to the third iteration, which has the highest average F1 value among the iterations. At the end

[4] http://nlp.stanford.edu/software/corenlp.shtml

of this step we have a seed synset list consisting of 2572 synsets yielding the highest performance when used to learn a sensorial model.

3.3 Sensorial Lexicon Construction Using Corpus Statistics

After generating the seed lists consisting of synsets for each sense category with the help of a set of WordNet relations and a bootstrapping process, we use corpus statistics to create our final sensorial lexicon. More specifically, we exploit a probabilistic approach based on the co-occurrence of the seeds and the candidate lexical entries. Since working on the synset level would raise the data sparsity problem in synset tagged corpora such as SemCor (Miller et al., 1993) and we need a corpus that provides sufficient statistical information, we migrate from synset level to lexical level. Accordingly, we treat each POS role of the same lemmas as a distinct seed and extract 4287 lemma-POS pairs from 2572 synsets. In this section, we explain the steps to construct our final sensorial lexicon in detail.

3.3.1 Corpus and Candidate Words

As a corpus, we use a subset of English GigaWord 5th Edition released by Linguistic Data Consortium (LDC)[5]. This resource is a collection of almost 10 million English newswire documents collected in recent years, whose content sums up to nearly 5 billion words. The richly annotated GigaWord data comprises automatic parses obtained with the Stanford parser (Klein and Manning, 2003) so that we easily have access to the lemma and POS information of each word in the resource. For the scope of this study, we work on a randomly chosen subset that contains 79800 sentences and we define a co-occurrence event as the co-existence of a candidate word and a seed word within a window of 9 words(the candidate word, 4 words to its left and 4 words to its right). In this manner, we analyze the co-occurrence of each unique lemma-POS pair in the corpus with the sense seeds. We eliminate the candidates which have less than 5 co-occurrences with the sense categories.

3.3.2 Normalized Pointwise Mutual Information

For the co-occurrence analysis of the candidate words and seeds, we use pointwise mutual information (PMI), which is simply a measure of association between the probability of the co-occurrence of two events and their individual probabilities when they are assumed to be independent (Church and Hanks, 1990). PMI can be exploited as a semantic similarity measure (Han et al., 2013) and it is calculated as:

$$PMI(x,y) = \log\left[\frac{p(x,y)}{p(x)p(y)}\right] \quad (1)$$

To calculate the PMI value of a candidate word and a specific sense, we consider *p(x)* as the probability of the candidate word to occur in the corpus. Therefore, *p(x)* is calculated as $p(x) = c(x)/N$, where c(x) is the total count of the occurrences of the candidate word *x* in the corpus and N is the total co-occurrence count of all words in the corpus. Similarly, we calculate *p(y)* as the total occurrence count of all the seeds for the sense considered (*y*). *p(y)* can thus be formulated as $c(y)/N$. *p(x,y)* is the probability of the co-occurrence of a candidate word *x* with a sense event *y*.

A major shortcoming of PMI is its sensitivity for low frequency data (Bouma, 2009). As one possible solution, the author introduces Normalized Pointwise Mutual Information (NPMI), which normalizes the PMI values to the range (-1, +1) with the following formula:

$$NPMI(x,y) = \frac{PMI(x,y)}{-\log p(x,y)} \quad (2)$$

We adopt the proposed solution and calculate NPMI values for each candidate word and five sense events in the corpus. Sensicon covers 22,684 lemma-POS pairs and a score for each sense class that denotes their association degrees.

4 Evaluation

To evaluate the performance of the sensorial classification and the quality of Sensicon, we first created a gold standard with the help of a crowdsourcing task. Then, we compared the decisions coming from Sensicon against the gold standard. In this section, we explain the annotation process that we conducted and the evaluation technique that we adopted in detail. We also provide a brief discussion about the obtained results.

4.1 Crowdsourcing to Build a Gold Standard

The evaluation phase of Sensicon requires a gold standard data to be able to conduct a meaningful assessment. Since to our best knowledge there is no resource with sensory associations of words or

[5] http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2011T07

	Sight			Hearing			Taste			Smell			Touch			Micro-average		
It#	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	.873	.506	.640	.893	.607	**.723**	.716	.983	.828	.900	.273	.419	.759	.320	.451	.780	.754	.729
2	.666	.890	**.762**	.829	.414	.552	.869	.929	.898	.746	.473	.579	.714	.439	.543	.791	.787	.772
3	.643	.878	.742	.863	.390	.538	.891	.909	**.900**	.667	.525	**.588**	.720	.482	.578	.796	.786	**.776**
4	.641	.869	.738	.832	.400	.540	.866	.888	.877	.704	.500	.585	.736	.477	**.579**	.784	.774	.765
5	.640	.869	.737	.832	.400	.540	.866	.888	.877	.704	.500	.585	.738	.474	.578	.784	.774	.764
Final	.805	.827	.816	.840	.408	.549	.814	.942	.873	.685	.534	.600	.760	.582	.659	.800	.802	.790

Table 1: Bootstrapping cycles with validation results.

majority class	3	4	5	6	7	8	9	10
word	0	0.98	3.84	9.96	11.63	16.66	34.41	12.42
sentence	0.58	2.35	7.07	10.91	13.27	15.63	21.23	16.51

Table 2: Percentage of words and sentences in each majority class.

sentences, we designed our own annotation task using the crowdsourcing service of CrowdFlower. For the annotation task, we first compiled a collection of sentences to be annotated. Then, we designed two questions that the annotators were expected to answer for a given sentence. While the first question is related to the sense association of a whole sentence, the second asks the annotators to collect a fine-grained gold standard for word-sense associations.

We collected a dataset of 340 sentences consisting of 300 advertisement slogans from 11 advertisement categories (e.g., fashion, food, electronics) and 40 story sentences from a story corpus. We collected the slogans from various online resources such as `http://slogans.wikia.com/wiki` and `http://www.adslogans.co.uk/`. The story corpus is generated as part of a dissertation research (Alm, 2008) and it provides stories as a collection of sentences.

In both resources, we first determined the candidate sentences that had at least five tokens and contained at least one adjective, verb or noun. In addition, we replaced the brand names in the advertisement slogans with *X* to prevent any bias. For instance, the name of a well-known restaurant in a slogan might cause a bias towards *taste*. Finally, the slogans used in the annotation task were chosen randomly among the candidate sentences by considering a balanced number of slogans from each category. Similarly, 40 story sentences were selected randomly among the candidate story sentences. To give a more concrete idea, for our dataset we obtained an advertisement slogan such as "*X's Sugar Frosted Flakes They're Great!*" or a story sentence such as "*The ground is frozen, and besides the snow has covered everything.*"

In the crowdsourcing task we designed, the annotators were required to answer 2 questions for a given sentence. In the first question, they were asked to detect the human senses conveyed or directly described by a given sentence. To exemplify these cases, we provided two examples such as "*I saw the cat*" that directly mentions the action of seeing and "*The sun was shining on the blue water.*" that conveys the sense of sight by using visual descriptions or elements like "*blue*" or "*shine*" which are notable for their visual properties. The annotators were able to select more than one sense for each sentence and together with the five senses we provided another option as *None* which should be selected when an annotator could not associate a sentence with any sense. The second question was devoted do determining word-sense associations. Here, the annotators were expected to associate the words in each sentence with at least one sense. Again, annotators could choose *None* for every word that they could not confidently associate with a sense.

The reliability of the annotators was evaluated on the basis of 20 control sentences which were highly associated with a specific sense and which included at least one sensorial word. For instance, for the control sentence "*The skin you love to touch*", we only considered as reliable the annotators who associated the sentence with *touch* and the word *touch* with the sense *touch*[6]. Similarly, for the slogan "*The most colourful name in cosmetics.*", an annotator was expected to associate the sentence with at least the sense *sight* and the word *colorful* to at least the sense *sight*. The raters who scored at least 70% accuracy on average on

[6]If the annotators gave additional answers to the expected ones, we considered their answers as correct.

the control questions for the two tasks were considered to be reliable. Each unit was annotated by at least 10 reliable raters.

Similarly to Mohammad (2011) and Özbal et al. (2011), we calculated the majority class of each annotated item to measure the agreement among the annotators. Table 2 demonstrates the observed agreement at both word and sentence level. Since 10 annotators participated in the task, the annotations with a majority class greater than 5 can be considered as reliable (Özbal et al., 2011). Indeed, for 85.10% of the word annotations the absolute majority agreed on the same decision, while 77.58% of the annotations in the sentence level have majority class greater than 5. The high agreement observed among the annotators in both cases confirms the quality of the resulting gold standard data.

In Table 3, we present the results of the annotation task by providing the association percentage of each category with each sense, namely *sight (Si)*, *hear (He)*, *taste (Ta)*, *smell (Sm)* and *touch (To)*. As demonstrated in the table, while the sense of *sight* can be observed in almost every advertisement category and in *story*, *smell* and *taste* are very rare. We observe that the story sentences invoke all sensory modalities except *taste*, although the percentage of sentences annotated with *smell* is relatively low. Similarly, *personal care* category has an association with four of the senses while the other categories have either very low or no association with some of the sense classes. Indeed, the perceived sensorial effects in the sentences vary according to the category such that the slogans in the *travel* category are highly associated with *sight* whereas the *communication* category is highly associated with *hearing*. While the connection of the *food* and *beverages* categories with *taste* is very high as expected, they have no association with the sense of *smell*. This kind of analysis could be useful for copywriters to decide which sensory modalities to invoke while creating a slogan for a specific product category.

Category	Si	He	Ta	Sm	To
personal care	49.36	10.75	0.00	13.29	26.58
travel	58.18	0.00	29.09	0.00	12.72
fashion	43.47	0.00	0.00	26.08	30.43
beauty	84.56	0.00	0.00	0.00	15.43
computing	32.25	59.13	0.00	0.00	8.60
food	0.00	5.46	94.53	0.00	0.00
beverages	22.68	0.00	59.79	0.00	17.52
communications	25.00	67.50	0.00	0.00	0.075
electronics	45.94	54.05	0.00	0.00	0.00
education	28.57	42.85	0.00	0.00	28.57
transport	61.81	38.18	0.00	0.00	0.00
story	58.37	20.81	0.00	7.23	13.57

Table 3: The categories of the annotated data and their sense association percentages.

4.2 Evaluation Measures

Based on the annotation results of our crowdsourcing task, we propose an evaluation technique considering that a lemma-POS or a sentence might be associated with more than one sensory modalities. Similar to the evaluation framework defined by Özbal et al. (2011), we adapt the evaluation measures of SemEval-2007 English Lexical Substitution Task (McCarthy and Navigli, 2007), where a system generates one or more possible substitutions for a target word in a sentence preserving its meaning.

For a given lemma-POS or a sentence, which we will name as *item* in the rest of the section, we allow our system to provide as many sensorial associations as it determines by using a specific lexicon. While evaluating a sense-item association of a method, a *best* and an *oot* score are calculated by considering the number of the annotators who associate that sense with the given item, the number of the annotators who associate any sense with the given item and the number of the senses the system gives as an answer for that item. More specifically, *best* scoring provides a credit for the best answer for a given item by dividing it to the number of the answers of the system. *oot* scoring, on the other hand, considers only a certain number of system answers for a given item and does not divide the credit to the total number of the answers. Unlike the lexical substitution task, a limited set of labels (i.e., 5 sense labels and *none*) are allowed for the sensorial annotation of sentences or lemma-POS pairs. For this reason, we reformulate *out-of-ten (oot)* scoring used by McCarthy and Navigli (2007) as out-of-two.

In Equation 3, *best* score for a given item *i* from the set of items *I*, which consists of the items annotated with a specific sense by a majority of 5 annotators, is formulated where H_i is the multiset of gold standard sense associations for item *i* and S_i is the set of sense associations provided by the system. *oot* scoring, as formulated in Equation 4, accepts up to 2 sense associations s from the answers of system S_i for a given item *i* and the credit is not divided by the number of the answers of the

system.

$$best\,(i) = \frac{\sum_{s \in S_i} freq\,(s \in H_i)}{|H_i| \cdot |S_i|} \quad (3)$$

$$oot\,(i) = \frac{\sum_{s \in S_i} freq\,(s \in H_i)}{|H_i|} \quad (4)$$

As formulated in Equation 5, to calculate the precision of an item-sense association task with a specific method, the sum of the scores (i.e., *best* or *oot*) for each item is divided by the number of items *A*, for which the method can provide an answer. In recall, the denominator is the number of the items in the gold standard for which an answer is given by the annotators.

$$P = \frac{\sum_{i \in A} score_i}{|A|} \qquad R = \frac{\sum_{i \in I} score_i}{|I|} \quad (5)$$

4.3 Evaluation Method

For the evaluation, we compare the accuracy of a simple classifier based on Sensicon against two baselines on a sense classification task both at word and sentence level. To achieve that, we use the gold standard that we obtain from the crowdsourcing task and the evaluation measures *best* and *oot*. The lexicon-based classifier simply assigns to each word in a sentence the sense values found in the lexicon. The first baseline assigns the most frequently annotated sensory modality, which is *sight*, via crowdsourcing task with a float value of 1.0 to each lemma-POS pair in the sensorial lexicon. The second baseline instead builds the associations by using a Latent Semantic Analysis space generated from the same subset of LDC that we exploit for constructing Sensicon. More specifically, this baseline calculates the LSA similarities between each candidate lemma-POS pair and sense class by taking the cosine similarity between the vector of the target lemma-POS pair and the average of the vectors of the related sensory word (i.e., *see*, *hear*, *touch*, *taste*, and *smell*) for each possible POS tag. For instance, to get the association score of a lemma-POS pair with the sense sight, we first average the vectors of see (noun) and see (verb) before calculating its cosine similarity with the target lemma-POS pair.

For the first experiment, i.e., word-sense association, we automatically associate the lemma-POS pairs obtained from the annotated dataset with senses by using i) Sensicon, ii) the most-frequent-sense baseline (MFS), iii) the LSA baseline. To achieve that, we lemmatize and POS tag each sentence in the dataset by using Stanford Core NLP. In the end, for each method and target word, we obtain a list of senses sorted according to their sensorial association values in decreasing order. It is worth noting that we only consider the non-negative sensorial associations for Sensicon and both baselines. For instance, Sensicon associates the noun *wine* with [*smell, taste, sight*]. In this experiment, *best* scoring considers the associated senses as the best answer, *smell, taste, sight* according to the previous example, and calculates a score with respect to the best answer in the gold standard and the number of the senses in this answer. Instead, *oot* scoring takes the first two answers, *smell* and *taste* according to the previous example, and assigns the score accordingly.

To determine the senses associated with a sentence for the second experiment, we use a method similar to the one proposed by Turney (2002). For each sense, we simply calculate the average score of the lemma-POS pairs in a sentence. We set a threshold value of 0 to decide whether a sentence is associated with a given sense. In this manner, we obtain a sorted list of average sensory scores for each sentence according to the three methods. For instance, the classifier based on Sensicon associates the sentence *Smash it to pieces, love it to bits.* with [*touch*, *taste*]. For the *best* score, only *touch* would be considered, whereas *oot* would consider both *touch* and *taste*.

4.4 Evaluation Results

In Table 4, we list the F1 values that we obtained with the classifier using Sensicon and the two baselines (MFS and LSA) according to both *best* and *oot* measures. In addition, we provide the performance of Sensicon in two preliminary steps, before bootstrapping (BB) and after bootstrapping (AB) to observe the incremental progress of the lexicon construction method. As can be observed from the table, the best performance for both experiments is achieved by Sensicon when compared against the baselines.

While in the first experiment the lexicon generated after the bootstrapping step (AB) provides a very similar performance to the final lexicon according to the *best* measure, it can only build sense associations for 69 lemmas out of 153 appearing in the gold standard. Instead, the final lexicon attempts to resolve 129 lemma-sense associations and results in a better recall value. Additionally, AB yields a very high precision as expected,

since it is created by a controlled semantical expansion from manually annotated sensorial words. BB lexicon includes only 573 lemmas which are collected from 277 synsets and we can not obtain 2 sense association scores for *oot* in this lexicon since each lemma is associated with only one sense with a value of 1. The LSA baseline yields a very low performance in the *best* measure due to its tendency to derive positive values for all sensorial associations of a given lemma-POS tuple. Another observed shortcoming of LSA is its failure to correlate the names of the colors with *sight* while this association is explicit for the annotators. On the other hand, LSA baseline significantly improves the MFS baseline with a p-value of 0.0009 in *oot* measures. This result points out that even though LSA provides very similar positive association values for almost all the sensory modalities for a given item, the first two sensorial associations with the highest values yield a better performance on guessing the sensorial characteristics of a lemma-POS. Nevertheless, Sensicon significantly outperforms the LSA baseline in both *best* and *oot* measures with the p-values of 0.0009 and 0.0189 respectively. The statistical significance tests are conducted using one-sided bootstrap resampling (Efron and Tibshirani, 1994).

Concerning the sentence classification experiment, the classifier using Sensicon yields the highest performance in both measures. The very high F1 value obtained with the *oot* scoring indicates that the right answer for a sentence is included in the first two decisions in many cases. Sensicon significantly outperforms the LSA baseline on the *best* measure (p-value = 0.0069). On the other hand, when systems are allowed to provide two answers (*oot*), the performance of LSA comes close to Sensicon in terms of F1 measure.

After the manual analysis of Sensicon and gold standard data, we observe that the sensorial classification task could be nontrivial. For instance, a story sentence "*He went to sleep again and snored until the windows shook.*" has been most frequently annotated as *hearing*. While the sensorial-lexicon classifier associates this sentence with *touch* as the best answer, it can provide the correct association *hearing* as the second best answer. To find out the best sensorial association for a sentence, a classification method which exploits various aspects of sensorial elements in a sentence, such as the number of sensorial words or their dependencies, could be a better approach than using only the average sensorial values.

	Lemma		Sentence	
Model	*best*	*oot*	*best*	*oot*
Most-Frequent-Sense	33.33	33.33	38.90	38.90
LSA	18.80	70.38	53.44	76.51
Lexicon-BB	45.22	45.22	49.60	51.12
Lexicon-AB	55.85	55.85	59.89	63.21
Sensicon	55.86	80.13	69.76	80.73

Table 4: Evaluation results.

Based on our observations of the error cases, we believe that synaesthesia, which is one of the most common metaphoric transfers in language (Williams, 1976), should be further explored for sense classification. As an example observation, the advertisement slogan "*100% pure squeezed sunshine*" is associated with *touch* as the best answer by Sensicon and *taste* by LSA baseline while it is most frequently annotated as *sight* in the gold standard. This slogan is an example usage of synaesthesia and metaphors in advertising language. To clarify, a product from the category of *beverages*, which might be assumed to have a *taste* association, is described by a metaphorical substitution of a *taste*-related noun, most probably the name of a fruit, with a *sight*-related noun; *sunshine*. This metaphorical substitution, then used as the object of a *touch*-related verb, *to squeeze*, produces a synaesthetic expression with *touch* and *sight*.

5 Conclusion

In this paper we have presented the construction of Sensicon, a sensorial lexicon, which associates words with sensory modalities. This novel aspect of word semantics is captured by employing a two-step strategy. First, we collected seed words by using a bootstrapping approach based on a set of WordNet relations. Then, we performed a corpus based statistical analysis to produce the final lexicon. Sensicon consists of 22,684 lemma-POS pairs and their association degrees with five sensory modalities. To the best of our knowledge, this is the first systematic attempt to build a sensorial lexicon and we believe that our contribution constitutes a valid starting point for the community to consider sensorial information conveyed by text as a feature for various tasks and applications. The results that we obtain by comparing our lexicon against the gold standard and two baselines are promising even though not conclusive. The results confirm the soundness of the proposed approach for the construction of the lexicon and the useful-

ness of the resource for text classification and possibly other computational applications.

Sensicon is publicly available upon request to the authors so that the community can benefit from it for relevant tasks. From a resource point of view, we would like to explore the effect of using different kinds of WordNet relations during the bootstrapping phase. It would also be interesting to experiment with relations provided by other resources such as ConceptNet (Liu and Singh, 2004), which is a semantic network containing common sense, cultural and scientific knowledge. We would also like to use the sensorial lexicon for various applicative scenarios such as slanting existing text towards a specific sense with text modification. We believe that our resource could be extremely useful for automatic content personalization according to user profiles. As an example, one can imagine a system that automatically replaces hearing based expressions with sight based ones in pieces of texts for a hearing-impaired person. Automating the task of building sensorial associations could also be beneficial for various tasks that need linguistic creativity. For instance, copywriters can take advantage of a system detecting the sensorial load of a piece of text to generate more appropriate advertisement slogans for specific product categories. Finally, we plan to investigate the impact of using sensory information for metaphor detection and interpretation based on our observations during the evaluation. For instance, the synaesthetic metaphor *bittersweet symphony* could be detected by determining the sensorial characterizations of its components.

Acknowledgements

We would like to thank Daniele Pighin for his insightful comments and valuable suggestions.
This work was partially supported by the PerTe project (Trento RISE).

References

Ebba Cecilia Ovesdotter Alm. 2008. *Affect in Text and Speech*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. pages 86–90. Association for Computational Linguistics.

Carmen Banea, Rada Mihalcea, and Janyce Wiebe. 2008. A bootstrapping method for building subjectivity lexicons for languages with scarce resources. In *LREC*.

Douglas A. Bernstein. 2010. *Essentials of Psychology*. PSY 113 General Psychology Series. Cengage Learning.

Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. 1992. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the 5th Annual Workshop on Computational learning theory*.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16(1):22–29, March.

Max Coltheart. 1981. The mrc psycholinguistic database. *The Quarterly Journal of Experimental Psychology*, 33(4):497–505.

Ivan E. de Araujo, Edmund T. Rolls, Maria Inés Velazco, Christian Margot, and Isabelle Cayeux. 2005. Cognitive modulation of olfactory processing. *Neuron*, 46(4):671–679.

Gaël Harry Dias, Mohammed Hasanuzzaman, Stéphane Ferrari, and Yann Mathet. 2014. Tempowordnet for sentence time tagging. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, WWW Companion '14, pages 833–838, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Bradley Efron and Robert J. Tibshirani. 1994. *An introduction to the bootstrap*, volume 57. CRC press.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*, volume 6, pages 417–422.

Christiane Fellbaum, editor. 1998. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London.

Lushan Han, Tim Finin, Paul McNamee, Anupam Joshi, and Yelena Yesha. 2013. Improving word similarity by augmenting pmi with estimates of word polysemy. *Knowledge and Data Engineering, IEEE Transactions on*, 25(6):1307–1322.

Thomas Kjeller Johansen. 1997. *Aristotle on the Sense-organs*. Cambridge University Press.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Hugo Liu and Push Singh. 2004. Conceptnet - a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226, October.

Asifa Majid and Stephen C. Levinson. 2011. The senses in language and culture. *The Senses and Society*, 6(1):5–18.

Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 48–53. Association for Computational Linguistics.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proceedings of the workshop on Human Language Technology*, HLT '93, pages 303–308, Stroudsburg, PA, USA. Association for Computational Linguistics.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34. Association for Computational Linguistics.

Saif M. Mohammad. 2011. Colourful language: Measuring word-colour associations. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 97–106. Association for Computational Linguistics.

Gözde Özbal, Carlo Strapparava, Rada Mihalcea, and Daniele Pighin. 2011. A comparison of unsupervised methods to associate colors with words. In *Affective Computing and Intelligent Interaction*, pages 42–51. Springer.

Jessica Perrie, Aminul Islam, Evangelos Milios, and Vlado Keselj. 2013. Using google n-grams to expand word-emotion association lexicon. In *Computational Linguistics and Intelligent Text Processing*, pages 137–148. Springer.

Mario Pricken. 2008. *Creative Advertising Ideas and Techniques from the World's Best Campaigns*. Thames & Hudson, 2^{nd} edition.

Raul Rodriguez-Esteban and Andrey Rzhetsky. 2008. Six senses in the literature. The bleak sensory landscape of biomedical texts. *EMBO reports*, 9(3):212–215, March.

Carlo Strapparava and Alessandro Valitutti. 2004. WordNet-Affect: an affective extension of WordNet. In *Proceedings of LREC*, volume 4, pages 1083–1086.

Sara Tonelli and Daniele Pighin. 2009. New features for framenet - wordnet mapping. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL'09)*, Boulder, CO, USA.

Peter D. Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the 2011 Conference on the Empirical Methods in Natural Language Processing*, pages 680–690.

Peter D. Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics.

Alessandro Valitutti, Carlo Strapparava, and Oliviero Stock. 2004. Developing affective lexical resources. *PsychNology Journal*, 2(1):61–83.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.

Joseph M. Williams. 1976. Synaesthetic adjectives: A possible law of semantic change. *Language*, pages 461–478.

Changhua Yang, Kevin Hsin-Yih Lin, and Hsin-Hsi Chen. 2007. Building emotion lexicon from weblog corpora. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 133–136. Association for Computational Linguistics.

Word Semantic Representations using Bayesian Probabilistic Tensor Factorization

Jingwei Zhang and **Jeremy Salwen**
Columbia University
Computer Science
New York, NY 10027, USA
{jz2541,jas2312}@columbia.edu

Michael Glass and **Alfio Gliozzo**
IBM T.J. Waston Research
Yorktown Heights, NY 10598, USA
{mrglass,gliozzo}@us.ibm.com

Abstract

Many forms of word relatedness have been developed, providing different perspectives on word similarity. We introduce a Bayesian probabilistic tensor factorization model for synthesizing a single word vector representation and per-perspective linear transformations from any number of word similarity matrices. The resulting word vectors, when combined with the per-perspective linear transformation, approximately recreate while also regularizing and generalizing, each word similarity perspective.

Our method can combine manually created semantic resources with neural word embeddings to separate synonyms and antonyms, and is capable of generalizing to words outside the vocabulary of any particular perspective. We evaluated the word embeddings with GRE antonym questions, the result achieves the state-of-the-art performance.

1 Introduction

In recent years, vector space models (VSMs) have been proved successful in solving various NLP tasks including named entity recognition, part-of-speech tagging, parsing, semantic role-labeling and answering synonym or analogy questions (Turney et al., 2010; Collobert et al., 2011). Also, VSMs are reported performing well on tasks involving the measurement of word relatedness (Turney et al., 2010). Many existing works are distributional models, based on the *Distributional Hypothesis*, that words occurring in similar contexts tend to have similar meanings (Harris, 1954). The limitation is that word vectors developed from distributional models cannot reveal word relatedness if its information does not lie in word distributions. For instance, they are believed to have difficulty distinguishing antonyms from synonyms, because the distribution of antonymous words are close, since the context of antonymous words are always similar to each other (Mohammad et al., 2013). Although some research claims that in certain conditions there do exist differences between the contexts of different antonymous words (Scheible et al., 2013), the differences are subtle enough that it can hardly be detected by such language models, especially for rare words.

Another important class of lexical resource for word relatedness is a lexicon, such as WordNet (Miller, 1995) or Roget's Thesaurus (Kipfer, 2009). Manually producing or extending lexicons is much more labor intensive than generating VSM word vectors using a corpus. Thus, lexicons are sparse with missing words and multi-word terms as well as missing relationships between words. Considering the synonym / antonym perspective as an example, WordNet answers less than 40% percent of the the GRE antonym questions provided by Mohammad et al. (2008) directly. Moreover, binary entries in lexicons do not indicate the degree of relatedness, such as the degree of lexical contrast between *happy* and *sad* or *happy* and *depressed*. The lack of such information makes it less fruitful when adopted in NLP applications.

In this work, we propose a Bayesian tensor factorization model (BPTF) for synthesizing a composite word vector representation by combining multiple different sources of word relatedness. The input is a set of word by word matrices, which may be sparse, providing a number indicating the presence or degree of relatedness. We treat word relatedness matrices from different perspectives as slices, forming a word relatedness tensor. Then the composite word vectors can be efficiently obtained by performing BPTF. Furthermore, given any two words and any trained relatedness perspective, we

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1522–1531,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

can create or recreate the pair-wise word relatedness with regularization via per-perspective linear transformation.

This method allows one set of word vectors to represent word relatednesses from many different perspectives (e.g. LSA for topic relatedness / corpus occurrences, ISA relation and YAGO type) It is able to bring the advantages from both word relatedness calculated by distributional models, and manually created lexicons, since the former have much more vocabulary coverage and many variations, while the latter covers word relatedness that is hard to detect by distributional models. We can use information from distributional perspectives to create (if does not exist) or re-create (with regularization) word relatedness from the lexicon's perspective.

We evaluate our model on distinguishing synonyms and antonyms. There are a number of related works (Lin and Zhao, 2003; Turney, 2008; Mohammad et al., 2008; Mohammad et al., 2013; Yih et al., 2012; Chang et al., 2013). A number of sophisticated methods have been applied, producing competitive results using diverse approaches. We use the GRE antonym questions (Mohammad et al., 2008) as a benchmark, and answer these questions by finding the most contrasting choice according to the created or recreated synonym / antonym word relatedness. The result achieves state-of-the-art performance.

The rest of this paper is organized as follows. Section 2 describes the related work of word vector representations, the BPTF model and antonymy detection. Section 3 presents our BPTF model and the sampling method. Section 4 shows the experimental evaluation and results with Section 5 providing conclusion and future work.

2 Related Work

2.1 Word Vector Representations

Vector space models of semantics have a long history as part of NLP technologies. One widely-used method is deriving word vectors using latent semantic analysis (LSA) (Deerwester et al., 1990), for measuring word similarities. This provides a topic based perspective on word similarity. In recent years, neural word embeddings have proved very effective in improving various NLP tasks (e.g. part-of-speech tagging, chunking, named entity recognition and semantic role labeling) (Collobert et al., 2011). The proposed neural models have a large number of variations, such as feed-forward networks (Bengio et al., 2003), hierarchical models (Mnih and Hinton, 2008), recurrent neural networks (Mikolov, 2012), and recursive neural networks (Socher et al., 2011). Mikolov et al. (2013) reported their vector-space word representation is able to reveal linguistic regularities and composite semantics using simple vector addition and subtraction. For example, "King$-$Man$+$Woman" results in a vector very close to "Queen". Luong et al. (2013) proposed a recursive neural networks model incorporating morphological structure, and has better performance for rare words.

Some non-VSM models[1] also generate word vector representations. Yih et al. (2012) apply polarity inducing latent semantic analysis (PILSA) to a thesaurus to derive the embedding of words. They treat each entry of a thesaurus as a document giving synonyms positive term counts, and antonyms negative term counts, and preform LSA on the signed TF-IDF matrix In this way, synonyms will have cosine similarities close to one and antonyms close to minus one.

Chang et al. (2013) further introduced Multi-Relational LSA (MRLSA), as as extension of LSA, that performs Tucker decomposition over a three-way tensor consisting of multiple relations (document-term like matrix) between words as slices, to capture lexical semantics. The purposes of MRLSA and our model are similar, but the different factorization techniques offer different advantages. In MRLSA, the k-th slice of tensor $\mathcal{W}$ is approximated by

$$\mathcal{W}_{:,:,k} \approx \mathcal{X}_{:,:,k} = \mathbf{U} S_{:,:,k} \mathbf{V}^T,$$

where $\mathbf{U}$ and $\mathbf{V}$ are both for the same word list but are not guaranteed (or necessarily desired) to be the same. Thus, this model has the ability to capture asymmetric relations, but this flexibility is a detriment for symmetric relatedness. In order to expand word relatedness coverage, MRLSA needs to choose a pivot slice (e.g. the synonym slice), thus there always must existence such a slice, and the model performance depends on the quality of this pivot slice. Also, while non-completeness is a pervasive issue in manually created lexicons, MRLSA is not flexible enough to treat the unknown entries as missing. Instead it just sets them

[1] As defined by Turney et al. (2010), VSM must be derived from event frequencies.

to zero at the beginning and uses the pivot slice to re-calculate them. In contrast, our method of BPTF is well suited to symmetric relations with many unknown relatedness entries.

2.2 BPTF Model

Salakhutdinov and Mnih (2008) introduced a Bayesian Probabilistic Matrix Factorization (BPMF) model as a collaborative filtering algorithm. Xiong et al. (2010) proposed a Bayesian Probabilistic Tensor Factorization (BPTF) model which further extended the original model to incorporate temporal factors. They modeled latent feature vector for users and items, both can be trained efficiently using Markov chain Monte Carlo methods, and they obtained competitive results when applying their models on real-world recommendation data sets.

2.3 Antonomy Detection

There are a number of previous works in detecting antonymy. Lin and Zhao (2003) identifies antonyms by looking for pre-identified phrases in corpus datasets. Turney (2008) proposed a supervised classification method for handling analogies, then apply it to antonyms by transforming antonym pairs into analogy relations. Mohammad et al. (Mohammad et al., 2008; Mohammad et al., 2013) proposed empirical approaches considering corpus co-occurrence statistics and the structure of a published thesaurus. Based on the assumption that the strongly related words of two words in a contrasting pair are also often antonymous, they use affix patterns (e.g. "un-", "in-" and "im-") and a thesaurus as seed sets to add contrast links between word categories. Their best performance is achieved by further manually annotating contrasting adjacent categories. This approach relies on the *Contrast Hypothesis*, which will increase false positives even with a carefully designed methodology. Furthermore, while this approach can expand contrast relationships in a lexicon, out-of-vocabulary words still pose a substancial challenge.

Yih et al. (2012) and Chang et al. (2013) also applied their vectors on antonymy detection, and Yih et al. achieves the state-of-the-art performance in answering GRE antonym questions. In addition to the word vectors generated from PILSA, they use morphology and k-nearest neighbors from distributional word vector spaces to derive the embeddings for out-of-vocabulary words. The latter is problematic since both synonyms and antonyms are distributionally similar. Their approach is two stage: polarity inducing LSA from a manually created thesaurus, then falling back to morphology and distributional similarity when the lexicon lacks coverage. In contrast, we focus on fusing the information from thesauruses and automatically induced word relatedness measures during the word vector space creation. Then prediction is done in a single stage, from the latent vectors capturing all word relatedness perspectives and the appropriate per-perspective transformation vector.

3 Methods

3.1 The Bayesian Probabilistic Tensor Factorization Model

Our model is a variation of the BPMF model (Salakhutdinov and Mnih, 2008), and is similar to the temporal BPTF model (Xiong et al., 2010). To model word relatedness from multiple perspectives, we denote the relatedness between word i and word j from perspective k as R_{ij}^k. Then we can organize these similarities to form a three-way tensor $\mathbf{R} \in \mathbb{R}^{N \times N \times K}$.

Table 1 shows an example, the first slice of the tensor is a $N \times N$ matrix consists of 1/-1 corresponding to the synonym/antonym entries in the Roget's thesaurus, and the second slice is a $N \times N$ matrix consists of the cosine similarity from neural word embeddings created by Luong et al. (2013), where N is the number of words in the vocabulary. Note that in our model the entries missing in Table 1a do not necessarily need to be treated as zero. Here we use the indicator variable I_{ij}^k to denote if the entry R_{ij}^k exists ($I_{ij}^k = 1$) or not ($I_{ij}^k = 0$). If $K = 1$, the BPTF model becomes to BPMF. Hence the key difference between BPTF and BPMF is that the former combines multiple complementary word relatedness perspectives, while the later only smooths and generalizes over one.

We assume the relatedness R_{ij}^k to be Gaussian, and can be expressed as the inner-product of three D-dimensional latent vectors:

$$R_{ij}^k | V_i, V_j, P_k \sim \mathcal{N}(< V_i, V_j, P_k >, \alpha^{-1}),$$

where $< \cdot, \cdot, \cdot >$ is a generalization of dot product:

$$< V_i, V_j, P_k > \equiv \sum_{d=1}^{D} V_i^{(d)} V_j^{(d)} P_k^{(d)},$$

	happy	joyful	lucky	sad	depressed
happy		1	1	-1	-1
joyful	1			-1	
lucky	1			-1	
sad	-1	-1	-1		1
depressed	-1			1	

(a) The first slice: synonym & antonym relatedness

	happy	joyful	lucky	sad	depressed
happy		.03	.61	.65	.13
joyful	.03		.25	.18	.23
lucky	.61	.25		.56	.31
sad	.65	.18	.56		-.01
depressed	.13	.23	.31	-.01	

(b) The second slice: distributional similarity

Table 1: Word Relatedness Tensor

and α is the precision, the reciprocal of the variance. V_i and V_j are the latent vectors of word i and word j, and P_k is the latent vector for perspective k.

We follow a Bayesian approach, adding Gaussian priors to the variables:

$$V_i \sim \mathcal{N}(\mu_V, \Lambda_V^{-1}),$$

$$P_i \sim \mathcal{N}(\mu_P, \Lambda_P^{-1}),$$

where μ_V and μ_P are D dimensional vectors and Λ_V and Λ_P are D-by-D precision matrices.

Furthermore, we model the prior distribution of hyper-parameters as conjugate priors (following the model by (Xiong et al., 2010)):

$$p(\alpha) = \mathcal{W}(\alpha|\hat{W}_0, \nu_0),$$

$$p(\mu_V, \Lambda_V) = \mathcal{N}(\mu_V|\mu_0, (\beta_0\Lambda_V)^{-1})\mathcal{W}(\Lambda_V|W_0, \nu_0),$$

$$p(\mu_P, \Lambda_P) = \mathcal{N}(\mu_P|\mu_0, (\beta_0\Lambda_P)^{-1})\mathcal{W}(\Lambda_P|W_0, \nu_0),$$

where $\mathcal{W}(W_0, \nu_0)$ is the Wishart distribution of degree of freedom ν and a D-by-D scale matrix W, and $\hat{W}_0$ is a 1-by-1 scale matrix for α. The graphical model is shown in Figure 1 (with β_0 set to 1). After choosing the hyper-priors, the only remaining parameter to tune is the dimension of the latent vectors.

Due to the existence of prior distributions, our model can capture the correlation between different perspectives during the factorization stage, then create or re-create word relatedness using this correlation for regularization and generalization. This advantage is especially useful when such correlation is too subtle to be captured by other methods. On the other hand, if perspectives (let's say k and l) are actually unrelated, our model can handle it as well by making P_k and P_l orthogonal to each other.

3.2 Inference

To avoid calculating intractable distributions, we use a numerical method to approximate the results. Here we use the Gibbs sampling algorithm

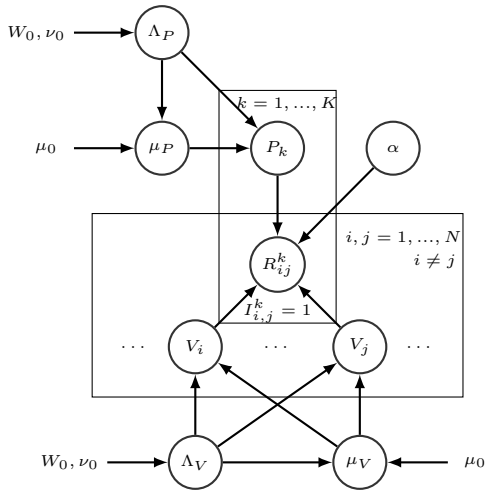


Figure 1: The graphical model for BPTF.

to perform the Markov chain Monte Carlo method. When sampling a block of parameters, all other parameters are fixed, and this procedure is repeated many times until convergence. The sampling algorithm is shown in Algorithm 1.

With conjugate priors, and assuming $I_{i,i}^k = 0, \forall i, k$ (we do not consider a word's relatedness to itself), the posterior distributions for each block of parameters are:

$$p(\alpha|\mathbf{R}, \mathbf{V}, \mathbf{P}) = \mathcal{W}(\hat{W_0}^*, \hat{\nu_0}^*) \quad (1)$$

Where:

$$\hat{\nu}_0^* = \hat{\nu}_0 + \sum_{k=1}^{2}\sum_{i,j=1}^{N} I_{ij}^k,$$

$$(\hat{W}_0^*)^{-1} = \hat{W}_0^{-1} + \sum_{k=1}^{2}\sum_{i,j=1}^{N} I_{ij}^k(R_{ij}^k - < V_i, V_j, P_k >)^2$$

$$p(\mu_V, \Lambda_V|\mathbf{V}) = \mathcal{N}(\mu_V|\mu_0^*, (\beta_0^*\Lambda_V)^{-1})\mathcal{W}(\Lambda_V|W_0^*, \nu_0^*) \tag{2}$$

Where:

$$\mu_0^* = \frac{\beta_0\mu_0 + N\bar{V}}{\beta_0 + N}, \beta_0^* = \beta_0 + N, \nu_0^* = \nu_0 + N,$$
$$(W_0^*)^{-1} = W_0^{-1} + N\bar{S} + \frac{\beta_0 N}{\beta_0 + N}(\mu_0 - \bar{V})(\mu_0 - \bar{V})^T,$$
$$\bar{V} = \frac{1}{N}\sum_{i=1}^{N} V_i, \bar{S} = \frac{1}{N}\sum_{i=1}^{N}(V_i - \bar{V})(V_i - \bar{V})^T$$

$$p(\mu_P, \Lambda_P|\mathbf{P}) = \mathcal{N}(\mu_P|\mu_0^*, (\beta_0^*\Lambda_P)^{-1})\mathcal{W}(\Lambda_P|W_0^*, \nu_0^*) \tag{3}$$

Which has the same form as $p(\mu_V, \Lambda_V|\mathbf{V})$.

$$p(V_i|\mathbf{R}, \mathbf{V}_{\neg i}, \mathbf{P}, \mu_V, \Lambda_V, \alpha) = \mathcal{N}(\mu_i^*, (\Lambda_i^*)^{-1}) \tag{4}$$

Where:

$$\mu_i^* = (\Lambda_i^*)^{-1}(\Lambda_V\mu_V + \alpha\sum_{k=1}^{2}\sum_{j=1}^{N} I_{ij}^k R_{ij}^k Q_{jk}),$$
$$\Lambda_i^* = \Lambda_V + \alpha\sum_{k=1}^{2}\sum_{j=1}^{N} I_{ij}^k Q_{jk}Q_{jk}^T,$$
$$Q_{jk} = V_j \odot P_k$$

$\odot$ is the element-wise product.

$$p(P_i|\mathbf{R}, \mathbf{V}, \mathbf{P}_{\neg i}, \mu_P, \Lambda_P, \alpha) = \mathcal{N}(\mu_i^*, (\Lambda_i^*)^{-1}) \tag{5}$$

Where:

$$\mu_k^* = (\Lambda_k^*)^{-1}(\Lambda_P\mu_P + \alpha\sum_{i,j=1}^{N} I_{ij}^k R_{ij}^k X_{ij}),$$
$$\Lambda_k^* = \Lambda_P + \alpha\sum_{i,j=1}^{N} I_{ij}^k X_{ij}X_{ij}^T,$$
$$X_{ij} = V_i \odot V_j$$

The influence each perspective k has on the latent word vectors is roughly propotional to the number of non-empty entries $n_k = \sum_{i,j} I_{i,j}^k$. If one wants to adjust the weight of each slices, this can easily achieved by adjusting (e.g. down sampling) the number of entries of each slice sampled at each iteration.

3.2.1 Out-of-Vocabulary words

It often occurs that some of the perspectives have greater word coverage than the others. For example, hand-labeled word relatedness usually has much less coverage than automatically acquired similarities. Of course, it is typically for the hand-labeled perspectives that the generalization is most desired. In this situation, our model can generalize word relatedness for the sparse perspective. For example, assume perspective k has larger vocabulary coverage N_k, while perspective l has a smaller coverage N_l.

There are two options for using the high vocabulary word relation matrix to generalize over the perspective with lower coverage. The most direct way simply considers the larger vocabulary in the BPTF $\mathbf{R} \in \mathbb{R}^{N_k \times N_k \times K}$ directly. A more efficient method trains on a tensor using the smaller vocabulary $\mathbf{R} \in \mathbb{R}^{N_l \times N_l \times K}$, then samples the $N_k - N_l$ word vectors using Equation 4.

Algorithm 1 Gibbs Sampling for BPTF

```
Initialize the parameters.
repeat
  Sample the hyper-parameters α, μ_V, Λ_V, μ_P,
  Λ_P (Equation 1, 2, 3)
  for i = 1 to N do
    Sample V_i (Equation 4)
  end for
  for k = 1 to 2 do
    Sample P_k (Equation 5)
  end for
until convergence
```

3.3 Predictions

With MCMC method, we can approximate the word relatedness distribution easily by averaging over a number of samples (instead of calculating intractable marginal distribution):

$$p(\hat{R}_{ij}^k|\mathbf{R}) \approx \frac{1}{M}\sum_{m=1}^{M} p(\hat{R}_{ij}^k|V_i^m, V_j^m, P_k^m, \alpha^m),$$

where m indicate parameters sampled from different sampling iterations.

3.4 Scalability

The time complexity of training our model is roughly $O(n \times D^2)$, where n is the number of observed entries in the tensor. If one is only interested in creating and re-creating word relatedness of one single slice rather than synthesizing word vectors, then entries in other slices can be down-sampled at every iteration to reduce the training time. In our model, the vector length D is not sensitive and does not necessarily need to be very long. Xiong et al. (2010) reported in their collaborative filtering experiment $D = 10$ usually gives satisfactory performance.

4 Experimental Evaluation

In this section, we evaluate our model by answering antonym questions. This task is especially suitable for evaluating our model since the performance of straight-forward look-up from the thesauruses we considered is poor. There are two major limitations:

1. The thesaurus usually only contains antonym information for word pairs with a strong contrast.

2. The vocabulary of the antonym entries in the thesaurus is limited, and does not contain many words in the antonym questions.

On the other hand, distributional similarities can be trained from large corpora and hence have a large coverage for words. This implies that we can treat the thesaurus data as the first slice, and the distributional similarities as the second slice, then use our model to create / recreate word relatedness on the first slice to answer antonym questions.

4.1 The GRE Antonym Questions

There are several publicly available test datasets to measure the correctness of our word embeddings. In order to be able to compare with previous works, we follow the widely-used GRE test dataset provided by (Mohammad et al., 2008), which has a development set (consisting of 162 questions) and a test set (consisting of 950 questions). The GRE test is a good benchmark because the words are relatively rare (19% of the words in Mohammad's test are not in the top 50,000 most frequent words from Google Books (Goldberg and Orwant, 2013)), thus it is hard to lookup answers from a thesaurus directly with high recall. Below is an example of the GRE antonym question:

adulterate: a. *renounce* b. *forbid*
c. *purify* d. *criticize* e. *correct*

The goal is to choose the most opposite word from the target, here the correct answer is *purify*.

4.2 Data Resources

In our tensor model, the first slice ($k = 1$) consists of synonyms and antonyms from public thesauruses, and the second slice ($k = 2$) consists of cosine similarities from neural word embeddings (example in Table 1)

4.2.1 Thesaurus

Two popular thesauruses used in other research are *the Macquarie Thesaurus* and *the Encarta Thesaurus*. Unfortunately, their electronic versions are not publicly available. In this work we use two alternatives:

WordNet Words in WordNet (version 3.0) are grouped into sense-disambiguated synonym sets (synsets), and synsets have links between each other to express conceptual relations. Previous works reported very different look-up performance using WordNet (Mohammad et al., 2008; Yih et al., 2012), we consider this difference as different understanding of the WordNet structure. By extending "indirect antonyms" defined in WordNet to nouns, verbs and adverbs that similar words share the antonyms,we achieve a look-up performance close to Yih et al. (2012). Using this interpretation of WordNet synonym and antonym structure we obtain a thesaurus containing 54,239 single-token words. Antonym entries are present for 21,319 of them with 16.5 words per entry on average, and 52,750 of them have synonym entries with 11.7 words per entry on average.

Roget's Only considering single-token words, *the Roget's Thesaurus* (Kipfer, 2009) contains 47,282 words. Antonym entries are present for 8,802 of them with 4.2 words per entry on average, and 22,575 of them have synonym entries with 20.7 words per entry on average. Although the Roget's Thesaurus has a less coverage on both vocabulary and antonym pairs, it has better look-up precision in the GRE antonym questions.

4.2.2 Distributional Similarities

We use cosine similarity of the *morphRNN* word representations[2] provided by Luong et al. (2013) as a distributional word relatedness perspective. They used morphological structure in training recursive neural networks and the learned models outperform previous works on word similarity tasks, especially a task focused on rare words. The vector space models were initialized from existing word embeddings trained on Wikipedia. We use word embeddings adapted from Collobert et al. (2011). This advantage complements the weakness of the thesaurus perspective – that it has less coverage on rare words. The word vector data contains 138,218 words, and it covers 86.9% of the words in the GRE antonym questions. Combining the two perspectives, we can cover 99.8% of the

	Dev. Set			Test Set		
	Prec.	Rec.	F_1	Prec.	Rec.	F_1
WordNet lookup	0.40	0.40	0.40	0.42	0.41	0.42
WordNet PILSA	0.63	0.62	0.62	0.60	0.60	0.60
WordNet MRLSA	0.66	0.65	0.65	0.61	0.59	0.60
Encarta lookup	0.65	0.61	0.63	0.61	0.56	0.59
Encarta PILSA	0.86	0.81	0.84	0.81	0.74	0.77
Encarta MRLSA	0.87	0.82	0.84	**0.82**	0.74	0.78
Encarta PILSA + S2Net + Emebed	**0.88**	**0.87**	**0.87**	0.81	**0.80**	**0.81**
W&E MRLSA	0.88	0.85	0.87	0.81	0.77	0.79
WordNet lookup*	0.93	0.32	0.48	0.95	0.33	0.49
WordNet lookup	0.48	0.44	0.46	0.46	0.43	0.44
WordNet BPTF	0.63	0.63	0.63	0.63	0.62	0.62
Roget lookup*	1.00	0.35	0.52	0.99	0.31	0.47
Roget lookup	0.61	0.44	0.51	0.55	0.39	0.45
Roget BPTF	0.80	0.80	0.80	0.76	0.75	0.76
W&R lookup*	1.00	0.48	0.64	0.98	0.45	0.62
W&R lookup	0.62	0.54	0.58	0.59	0.51	0.55
W&R BPMF	0.59	0.59	0.59	0.52	0.52	0.52
W&R BPTF	**0.88**	**0.88**	**0.88**	**0.82**	**0.82**	**0.82**

Table 2: Development and test results on the GRE antonym questions. *Note: to allow comparison, in look-up we follow the approach used by (Yih et al., 2012): randomly guess an answer if the target word is in the vocabulary while none of the choices are. Asterisk indicates the look-up results without random guessing.

GRE antonym question words. Further using morphology information from WordNet, the coverage achieves 99.9%.

4.3 Tests

To answer the GRE questions, we calculate R^1_{ij} for word pair (i, j), where i is the target word and j is one of the question's candidates. The candidate with the smallest similarity is then the predicted answer. If a target word is missing in the vocabulary, that question will not be answered, while if a choice is missing, that choice will be ignored.

We first train on a tensor from a subset consisting of words with antonym entries, then add all other words using the out-of-vocabulary method described in Section 3. During each iteration, zeros are randomly added into the first slice to keep the model from overfitting. In the meantime, the second slice entries is randomly downsampled to match the number of non-empty entries in the first slice. This ensures each perspective has approximately equal influence on the latent word vectors.

We sample the parameters iteratively, and choose the burn-in period and vector length D according to the development set. We choose the vector length $D = 40$, the burn-in period starting from the 30[th] iterations, then averaging the relatedness over 200 runs. The hyper-priors used are $\mu_0 = 0$, $\nu_0 = \hat{\nu}_0 = D$, $\beta_0 = 1$ and $W_0 = \hat{W}_0 = \mathbf{I}$ (not tuned). Note that Yih et al. (2012) use a vector length of 300, which means our embeddings save considerable storage space and running time. Our model usually takes less than 30 minutes to meet the convergence criteria (on a machine with an Intel Xeon E3-1230V2 @ 3.3GHz CPU). In contrast, the MRLSA requires about 3 hours for tensor decomposition (Chang et al., 2013).

4.4 Results

The results are summarized in Table 2. We list the results of previous works (Yih et al., 2012; Chang et al., 2013) at the top of the table, where the best performance is achieved by PILSA on Encarta with further discriminative training and embedding. For comparison, we adopt the standard first used by (Mohammad et al., 2008), where *precision* is the number of questions answered correctly

[2]http://www-nlp.stanford.edu/ lmthang/morphoNLM/

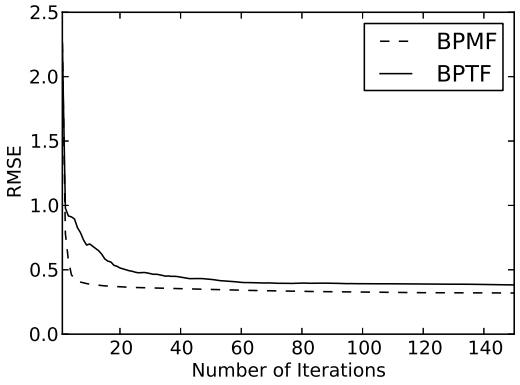


Figure 2: Convergence curves of BPMF and BPTF in training the W&R dataset. MAE is the mean absolute error over the synonym & antonym slice in the training tensor.

divided by the number of questions answered. *Recall* is the number of questions answered correctly divided by the total number of questions. BPMF (Bayesian Probabilistic Matrix Factorization) result is derived by only keeping the synonym & antonym slice in our BPTF model.

By using Roget's and WordNet together, our method increases the baseline look-up recall from 51% to 82% on the test set, while Yih's method increases the recall of Encarta from 56% to 80%. This state-of-the-art performance is achieved with the help of a neural network for fine tuning and multiple schemes of out-of-vocabulary embedding, while our method has inherent and straightforward "out-of-vocabulary embedding". While MRLSA, which has this character as well, only has a recall 77% when combining WordNet and Encarta together.

WordNet records less antonym relations for nouns, verbs and adverbs, while the GRE antonym questions has a large coverage of them. Although by extending these antonym relations using the "indirect antonym" concept achieves better look-up performance than Roget's, in contrast, the BPTF performance is actually much lower. This implies Roget's has better recording of antonym relations. Mohammad et al. (2008) reproted a 23% F-score look-up performance of WordNet which support this claim as well. Combining WordNet and Roget's together can improve the look-up performance further to 59% precision and 51% recall (still not as good as Encarta look-up).

Notably, if we strictly follow our BPTF approach but only use the synonym & antonym slice (i.e. a matrix factorization model instead of tensor factorization model), this single-slice model BPMF has performance that is only slightly better than look-up. Meanwhile Figure 1 shows the convergence curves of BPMF and BPTF. BPMF actually has lower MAE after convergence. Such behavior is caused by overfitting of BPMF on the training data. While known entries were recreated well, empty entries were not filled correctly. On the other hand, note that although our BPTF model has a higher MAE, it has much better performance in answering the GRE antonym questions. We interpret this as the regularization and generalization effect from other slice(s). Instead of focusing on one-slice training data, our model fills the missing entries with the help of inter-slice relations.

We also experimented with a linear metric learning method over the generated word vectors (to learn a metric matrix A to measure the word relatedness via $V_i^T A V_j$) using L-BFGS. By optimizing the mean square error on the synonym & antonym slice, we can reduce 8% of the mean square error on a held out test set, and improve the F-score by roughly 0.5% (of a single iteration). Although this method doesn't give a significant improvement, it is general and has the potential to boost the performance in other scenarios.

5 Conclusion

In this work, we propose a method to map words into a metric space automatically using thesaurus data, previous vector space models, or other word relatedness matrices as input, which is capable of handling out-of-vocabulary words of any particular perspective. This allows us to derive the relatedness of any given word pair and any perspective by the embedded word vectors with perperspective linear transformation. We evaluated the word embeddings with GRE antonym questions, and the result achieves the state-of-the-art performance.

For future works, we will extend the model and its applications in three main directions. First, in this model we only use a three-way tensor with two slices, while more relations may be able to add into it directly. Possible additional perspective slices include LSA for topic relatedness, and corpus occurrences in engineered or induced semantic patterns.

Second, we will apply the method to other tasks that require completing a word relatedness matrix. We evaluated the performance of our model on

creating / recreating one perspective of word relatedness: antonymy. Perhaps using vectors generated from many kinds of perspectives would improve the performance on other NLP tasks, such as term matching employed by textual entailment and machine translation metrics.

Third, if our model does learn the relation between semantic similarities and distributional similarities, there may be fruitful information contained in the vectors V_i and P_k that can be explored. One straight-forward idea is that the dot product of perspective vectors $P_k \cdot P_l$ should be a measurement of correlation between perspectives.

Also, a straightforward adaptation of our model has the potential ability to capture asymmetric word relatedness as well, by using a perperspective matrix instead of vector for the asymmetric slices (i.e. use $V_i^T A_k V_j$ instead of $\sum_{d=1}^{D} V_i^{(d)} P_k^{(d)} V_j^{(d)}$ for calculating word relatedness, where A_k is a square matrix).

Acknowledgments

We thank Christopher Kedzie for assisting the *Semantic Technologies in IBM Watson* seminar course in which this work has been carried out, and Kai-Wei Chang for giving detailed explanation of the evaluation method in his work.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In *EMNLP*.

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

Scott C. Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 1, pages 241–247.

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

Barbara Ann Kipfer. 2009. *Roget's 21st Century Thesaurus, Third Edition*. Philip Lief Group.

Dekang Lin and Shaojun Zhao. 2003. Identifying synonyms among distributionally similar words. In *In Proceedings of IJCAI-03*, page 14921493.

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

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751. The Association for Computational Linguistics.

Tom Mikolov. 2012. *Statistical language models based on neural networks*. Ph.D. thesis, Ph. D. thesis, Brno University of Technology.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.

Andriy Mnih and Geoffrey E. Hinton. 2008. A scalable hierarchical distributed language model. In *NIPS*, pages 1081–1088.

Saif Mohammad, Bonnie Dorr, and Graeme Hirst. 2008. Computing word-pair antonymy. In *EMNLP*, pages 982–991. Association for Computational Linguistics.

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

Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, pages 880–887. ACM.

Silke Scheible, Sabine Schulte im Walde, and Sylvia Springorum. 2013. Uncovering distributional differences between synonyms and antonyms in a word space model. *International Joint Conference on Natural Language Processing*, pages 489–497.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 129–136.

Peter D. Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Peter D. Turney. 2008. A uniform approach to analogies, synonyms, antonyms, and associations. *Coling*, pages 905–912, August.

Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G. Schneider, and Jaime G. Carbonell. 2010. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, volume 10, pages 211–222. SIAM.

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

GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning
Computer Science Department, Stanford University, Stanford, CA 94305
jpennin@stanford.edu, richard@socher.org, manning@stanford.edu

Abstract

Recent methods for learning vector space representations of words have succeeded in capturing fine-grained semantic and syntactic regularities using vector arithmetic, but the origin of these regularities has remained opaque. We analyze and make explicit the model properties needed for such regularities to emerge in word vectors. The result is a new global log-bilinear regression model that combines the advantages of the two major model families in the literature: global matrix factorization and local context window methods. Our model efficiently leverages statistical information by training only on the nonzero elements in a word-word co-occurrence matrix, rather than on the entire sparse matrix or on individual context windows in a large corpus. The model produces a vector space with meaningful substructure, as evidenced by its performance of 75% on a recent word analogy task. It also outperforms related models on similarity tasks and named entity recognition.

1 Introduction

Semantic vector space models of language represent each word with a real-valued vector. These vectors can be used as features in a variety of applications, such as information retrieval (Manning et al., 2008), document classification (Sebastiani, 2002), question answering (Tellex et al., 2003), named entity recognition (Turian et al., 2010), and parsing (Socher et al., 2013).

Most word vector methods rely on the distance or angle between pairs of word vectors as the primary method for evaluating the intrinsic quality of such a set of word representations. Recently, Mikolov et al. (2013c) introduced a new evaluation scheme based on word analogies that probes the finer structure of the word vector space by examining not the scalar distance between word vectors, but rather their various dimensions of difference. For example, the analogy "king is to queen as man is to woman" should be encoded in the vector space by the vector equation $king - queen = man - woman$. This evaluation scheme favors models that produce dimensions of meaning, thereby capturing the multi-clustering idea of distributed representations (Bengio, 2009).

The two main model families for learning word vectors are: 1) global matrix factorization methods, such as latent semantic analysis (LSA) (Deerwester et al., 1990) and 2) local context window methods, such as the skip-gram model of Mikolov et al. (2013c). Currently, both families suffer significant drawbacks. While methods like LSA efficiently leverage statistical information, they do relatively poorly on the word analogy task, indicating a sub-optimal vector space structure. Methods like skip-gram may do better on the analogy task, but they poorly utilize the statistics of the corpus since they train on separate local context windows instead of on global co-occurrence counts.

In this work, we analyze the model properties necessary to produce linear directions of meaning and argue that global log-bilinear regression models are appropriate for doing so. We propose a specific weighted least squares model that trains on global word-word co-occurrence counts and thus makes efficient use of statistics. The model produces a word vector space with meaningful substructure, as evidenced by its state-of-the-art performance of 75% accuracy on the word analogy dataset. We also demonstrate that our methods outperform other current methods on several word similarity tasks, and also on a common named entity recognition (NER) benchmark.

We provide the source code for the model as well as trained word vectors at `http://nlp.stanford.edu/projects/glove/`.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

2 Related Work

Matrix Factorization Methods. Matrix factorization methods for generating low-dimensional word representations have roots stretching as far back as LSA. These methods utilize low-rank approximations to decompose large matrices that capture statistical information about a corpus. The particular type of information captured by such matrices varies by application. In LSA, the matrices are of "term-document" type, i.e., the rows correspond to words or terms, and the columns correspond to different documents in the corpus. In contrast, the Hyperspace Analogue to Language (HAL) (Lund and Burgess, 1996), for example, utilizes matrices of "term-term" type, i.e., the rows and columns correspond to words and the entries correspond to the number of times a given word occurs in the context of another given word.

A main problem with HAL and related methods is that the most frequent words contribute a disproportionate amount to the similarity measure: the number of times two words co-occur with *the* or *and*, for example, will have a large effect on their similarity despite conveying relatively little about their semantic relatedness. A number of techniques exist that addresses this shortcoming of HAL, such as the COALS method (Rohde et al., 2006), in which the co-occurrence matrix is first transformed by an entropy- or correlation-based normalization. An advantage of this type of transformation is that the raw co-occurrence counts, which for a reasonably sized corpus might span 8 or 9 orders of magnitude, are compressed so as to be distributed more evenly in a smaller interval. A variety of newer models also pursue this approach, including a study (Bullinaria and Levy, 2007) that indicates that positive pointwise mutual information (PPMI) is a good transformation. More recently, a square root type transformation in the form of Hellinger PCA (HPCA) (Lebret and Collobert, 2014) has been suggested as an effective way of learning word representations.

Shallow Window-Based Methods. Another approach is to learn word representations that aid in making predictions within local context windows. For example, Bengio et al. (2003) introduced a model that learns word vector representations as part of a simple neural network architecture for language modeling. Collobert and Weston (2008) decoupled the word vector training from the downstream training objectives, which paved the way for Collobert et al. (2011) to use the full context of a word for learning the word representations, rather than just the preceding context as is the case with language models.

Recently, the importance of the full neural network structure for learning useful word representations has been called into question. The skip-gram and continuous bag-of-words (CBOW) models of Mikolov et al. (2013a) propose a simple single-layer architecture based on the inner product between two word vectors. Mnih and Kavukcuoglu (2013) also proposed closely-related vector log-bilinear models, vLBL and ivLBL, and Levy et al. (2014) proposed explicit word embeddings based on a PPMI metric.

In the skip-gram and ivLBL models, the objective is to predict a word's context given the word itself, whereas the objective in the CBOW and vLBL models is to predict a word given its context. Through evaluation on a word analogy task, these models demonstrated the capacity to learn linguistic patterns as linear relationships between the word vectors.

Unlike the matrix factorization methods, the shallow window-based methods suffer from the disadvantage that they do not operate directly on the co-occurrence statistics of the corpus. Instead, these models scan context windows across the entire corpus, which fails to take advantage of the vast amount of repetition in the data.

3 The GloVe Model

The statistics of word occurrences in a corpus is the primary source of information available to all unsupervised methods for learning word representations, and although many such methods now exist, the question still remains as to how meaning is generated from these statistics, and how the resulting word vectors might represent that meaning. In this section, we shed some light on this question. We use our insights to construct a new model for word representation which we call GloVe, for Global Vectors, because the global corpus statistics are captured directly by the model.

First we establish some notation. Let the matrix of word-word co-occurrence counts be denoted by X, whose entries X_{ij} tabulate the number of times word j occurs in the context of word i. Let $X_i = \sum_k X_{ik}$ be the number of times any word appears in the context of word i. Finally, let $P_{ij} = P(j|i) = X_{ij}/X_i$ be the probability that word j appear in the

Table 1: Co-occurrence probabilities for target words *ice* and *steam* with selected context words from a 6 billion token corpus. Only in the ratio does noise from non-discriminative words like *water* and *fashion* cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific of steam.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

context of word i.

We begin with a simple example that showcases how certain aspects of meaning can be extracted directly from co-occurrence probabilities. Consider two words i and j that exhibit a particular aspect of interest; for concreteness, suppose we are interested in the concept of thermodynamic phase, for which we might take $i = ice$ and $j = steam$. The relationship of these words can be examined by studying the ratio of their co-occurrence probabilities with various probe words, k. For words k related to ice but not steam, say $k = solid$, we expect the ratio P_{ik}/P_{jk} will be large. Similarly, for words k related to steam but not ice, say $k = gas$, the ratio should be small. For words k like *water* or *fashion*, that are either related to both ice and steam, or to neither, the ratio should be close to one. Table 1 shows these probabilities and their ratios for a large corpus, and the numbers confirm these expectations. Compared to the raw probabilities, the ratio is better able to distinguish relevant words (*solid* and *gas*) from irrelevant words (*water* and *fashion*) and it is also better able to discriminate between the two relevant words.

The above argument suggests that the appropriate starting point for word vector learning should be with ratios of co-occurrence probabilities rather than the probabilities themselves. Noting that the ratio P_{ik}/P_{jk} depends on three words i, j, and k, the most general model takes the form,

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}, \tag{1}$$

where $w \in \mathbb{R}^d$ are word vectors and $\tilde{w} \in \mathbb{R}^d$ are separate context word vectors whose role will be discussed in Section 4.2. In this equation, the right-hand side is extracted from the corpus, and F may depend on some as-of-yet unspecified parameters. The number of possibilities for F is vast, but by enforcing a few desiderata we can select a unique choice. First, we would like F to encode the information present the ratio P_{ik}/P_{jk} in the word vector space. Since vector spaces are inherently linear structures, the most natural way to do this is with vector differences. With this aim, we can restrict our consideration to those functions F that depend only on the difference of the two target words, modifying Eqn. (1) to,

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}. \tag{2}$$

Next, we note that the arguments of F in Eqn. (2) are vectors while the right-hand side is a scalar. While F could be taken to be a complicated function parameterized by, e.g., a neural network, doing so would obfuscate the linear structure we are trying to capture. To avoid this issue, we can first take the dot product of the arguments,

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}, \tag{3}$$

which prevents F from mixing the vector dimensions in undesirable ways. Next, note that for word-word co-occurrence matrices, the distinction between a word and a context word is arbitrary and that we are free to exchange the two roles. To do so consistently, we must not only exchange $w \leftrightarrow \tilde{w}$ but also $X \leftrightarrow X^T$. Our final model should be invariant under this relabeling, but Eqn. (3) is not. However, the symmetry can be restored in two steps. First, we require that F be a homomorphism between the groups $(\mathbb{R}, +)$ and $(\mathbb{R}_{>0}, \times)$, i.e.,

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)}, \tag{4}$$

which, by Eqn. (3), is solved by,

$$F(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i}. \tag{5}$$

The solution to Eqn. (4) is $F = \exp$, or,

$$w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i). \tag{6}$$

Next, we note that Eqn. (6) would exhibit the exchange symmetry if not for the $\log(X_i)$ on the right-hand side. However, this term is independent of k so it can be absorbed into a bias b_i for w_i. Finally, adding an additional bias $\tilde{b}_k$ for $\tilde{w}_k$ restores the symmetry,

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik}) \,. \tag{7}$$

Eqn. (7) is a drastic simplification over Eqn. (1), but it is actually ill-defined since the logarithm diverges whenever its argument is zero. One resolution to this issue is to include an additive shift in the logarithm, $\log(X_{ik}) \rightarrow \log(1 + X_{ik})$, which maintains the sparsity of X while avoiding the divergences. The idea of factorizing the log of the co-occurrence matrix is closely related to LSA and we will use the resulting model as a baseline in our experiments. A main drawback to this model is that it weighs all co-occurrences equally, even those that happen rarely or never. Such rare co-occurrences are noisy and carry less information than the more frequent ones — yet even just the zero entries account for 75–95% of the data in X, depending on the vocabulary size and corpus.

We propose a new weighted least squares regression model that addresses these problems. Casting Eqn. (7) as a least squares problem and introducing a weighting function $f(X_{ij})$ into the cost function gives us the model

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2 \,, \tag{8}$$

where V is the size of the vocabulary. The weighting function should obey the following properties:

1. $f(0) = 0$. If f is viewed as a continuous function, it should vanish as $x \rightarrow 0$ fast enough that the $\lim_{x \rightarrow 0} f(x) \log^2 x$ is finite.

2. $f(x)$ should be non-decreasing so that rare co-occurrences are not overweighted.

3. $f(x)$ should be relatively small for large values of x, so that frequent co-occurrences are not overweighted.

Of course a large number of functions satisfy these properties, but one class of functions that we found to work well can be parameterized as,

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} \,. \tag{9}$$

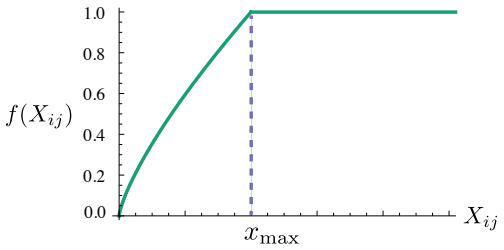


Figure 1: Weighting function f with $\alpha = 3/4$.

The performance of the model depends weakly on the cutoff, which we fix to $x_{\max} = 100$ for all our experiments. We found that $\alpha = 3/4$ gives a modest improvement over a linear version with $\alpha = 1$. Although we offer only empirical motivation for choosing the value 3/4, it is interesting that a similar fractional power scaling was found to give the best performance in (Mikolov et al., 2013a).

3.1 Relationship to Other Models

Because all unsupervised methods for learning word vectors are ultimately based on the occurrence statistics of a corpus, there should be commonalities between the models. Nevertheless, certain models remain somewhat opaque in this regard, particularly the recent window-based methods like skip-gram and ivLBL. Therefore, in this subsection we show how these models are related to our proposed model, as defined in Eqn. (8).

The starting point for the skip-gram or ivLBL methods is a model Q_{ij} for the probability that word j appears in the context of word i. For concreteness, let us assume that Q_{ij} is a softmax,

$$Q_{ij} = \frac{\exp(w_i^T \tilde{w}_j)}{\sum_{k=1}^{V} \exp(w_i^T \tilde{w}_k)} \,. \tag{10}$$

Most of the details of these models are irrelevant for our purposes, aside from the the fact that they attempt to maximize the log probability as a context window scans over the corpus. Training proceeds in an on-line, stochastic fashion, but the implied global objective function can be written as,

$$J = -\sum_{\substack{i \in \text{corpus} \\ j \in \text{context}(i)}} \log Q_{ij} \,. \tag{11}$$

Evaluating the normalization factor of the softmax for each term in this sum is costly. To allow for efficient training, the skip-gram and ivLBL models introduce approximations to Q_{ij}. However, the sum in Eqn. (11) can be evaluated much

more efficiently if we first group together those terms that have the same values for i and j,

$$J = -\sum_{i=1}^{V}\sum_{j=1}^{V} X_{ij} \log Q_{ij} \,, \tag{12}$$

where we have used the fact that the number of like terms is given by the co-occurrence matrix X.

Recalling our notation for $X_i = \sum_k X_{ik}$ and $P_{ij} = X_{ij}/X_i$, we can rewrite J as,

$$J = -\sum_{i=1}^{V} X_i \sum_{j=1}^{V} P_{ij} \log Q_{ij} = \sum_{i=1}^{V} X_i H(P_i, Q_i) \,, \tag{13}$$

where $H(P_i, Q_i)$ is the cross entropy of the distributions P_i and Q_i, which we define in analogy to X_i. As a weighted sum of cross-entropy error, this objective bears some formal resemblance to the weighted least squares objective of Eqn. (8). In fact, it is possible to optimize Eqn. (13) directly as opposed to the on-line training methods used in the skip-gram and ivLBL models. One could interpret this objective as a "global skip-gram" model, and it might be interesting to investigate further. On the other hand, Eqn. (13) exhibits a number of undesirable properties that ought to be addressed before adopting it as a model for learning word vectors.

To begin, cross entropy error is just one among many possible distance measures between probability distributions, and it has the unfortunate property that distributions with long tails are often modeled poorly with too much weight given to the unlikely events. Furthermore, for the measure to be bounded it requires that the model distribution Q be properly normalized. This presents a computational bottleneck owing to the sum over the whole vocabulary in Eqn. (10), and it would be desirable to consider a different distance measure that did not require this property of Q. A natural choice would be a least squares objective in which normalization factors in Q and P are discarded,

$$\hat{J} = \sum_{i,j} X_i \left(\hat{P}_{ij} - \hat{Q}_{ij}\right)^2 \tag{14}$$

where $\hat{P}_{ij} = X_{ij}$ and $\hat{Q}_{ij} = \exp(w_i^T \tilde{w}_j)$ are the unnormalized distributions. At this stage another problem emerges, namely that X_{ij} often takes very large values, which can complicate the optimization. An effective remedy is to minimize the squared error of the logarithms of $\hat{P}$ and $\hat{Q}$ instead,

$$\begin{aligned} \hat{J} &= \sum_{i,j} X_i \left(\log \hat{P}_{ij} - \log \hat{Q}_{ij}\right)^2 \\ &= \sum_{i,j} X_i \left(w_i^T \tilde{w}_j - \log X_{ij}\right)^2 . \end{aligned} \tag{15}$$

Finally, we observe that while the weighting factor X_i is preordained by the on-line training method inherent to the skip-gram and ivLBL models, it is by no means guaranteed to be optimal. In fact, Mikolov et al. (2013a) observe that performance can be increased by filtering the data so as to reduce the effective value of the weighting factor for frequent words. With this in mind, we introduce a more general weighting function, which we are free to take to depend on the context word as well. The result is,

$$\hat{J} = \sum_{i,j} f(X_{ij}) \left(w_i^T \tilde{w}_j - \log X_{ij}\right)^2 , \tag{16}$$

which is equivalent[1] to the cost function of Eqn. (8), which we derived previously.

3.2 Complexity of the model

As can be seen from Eqn. (8) and the explicit form of the weighting function $f(X)$, the computational complexity of the model depends on the number of nonzero elements in the matrix X. As this number is always less than the total number of entries of the matrix, the model scales no worse than $O(|V|^2)$. At first glance this might seem like a substantial improvement over the shallow window-based approaches, which scale with the corpus size, $|C|$. However, typical vocabularies have hundreds of thousands of words, so that $|V|^2$ can be in the hundreds of billions, which is actually much larger than most corpora. For this reason it is important to determine whether a tighter bound can be placed on the number of nonzero elements of X.

In order to make any concrete statements about the number of nonzero elements in X, it is necessary to make some assumptions about the distribution of word co-occurrences. In particular, we will assume that the number of co-occurrences of word i with word j, X_{ij}, can be modeled as a power-law function of the frequency rank of that word pair, r_{ij}:

$$X_{ij} = \frac{k}{(r_{ij})^\alpha} \,. \tag{17}$$

[1] We could also include bias terms in Eqn. (16).

The total number of words in the corpus is proportional to the sum over all elements of the co-occurrence matrix X,

$$|C| \sim \sum_{ij} X_{ij} = \sum_{r=1}^{|X|} \frac{k}{r^\alpha} = k H_{|X|,\alpha}\,, \qquad (18)$$

where we have rewritten the last sum in terms of the generalized harmonic number $H_{n,m}$. The upper limit of the sum, $|X|$, is the maximum frequency rank, which coincides with the number of nonzero elements in the matrix X. This number is also equal to the maximum value of r in Eqn. (17) such that $X_{ij} \geq 1$, i.e., $|X| = k^{1/\alpha}$. Therefore we can write Eqn. (18) as,

$$|C| \sim |X|^\alpha \, H_{|X|,\alpha}\,. \qquad (19)$$

We are interested in how $|X|$ is related to $|C|$ when both numbers are large; therefore we are free to expand the right hand side of the equation for large $|X|$. For this purpose we use the expansion of generalized harmonic numbers (Apostol, 1976),

$$H_{x,s} = \frac{x^{1-s}}{1-s} + \zeta(s) + O(x^{-s}) \quad \text{if} \quad s > 0,\, s \neq 1\,, \qquad (20)$$

giving,

$$|C| \sim \frac{|X|}{1-\alpha} + \zeta(\alpha)\,|X|^\alpha + O(1)\,, \qquad (21)$$

where $\zeta(s)$ is the Riemann zeta function. In the limit that X is large, only one of the two terms on the right hand side of Eqn. (21) will be relevant, and which term that is depends on whether $\alpha > 1$,

$$|X| = \begin{cases} O(|C|) & \text{if } \alpha < 1, \\ O(|C|^{1/\alpha}) & \text{if } \alpha > 1. \end{cases} \qquad (22)$$

For the corpora studied in this article, we observe that X_{ij} is well-modeled by Eqn. (17) with $\alpha = 1.25$. In this case we have that $|X| = O(|C|^{0.8})$. Therefore we conclude that the complexity of the model is much better than the worst case $O(V^2)$, and in fact it does somewhat better than the on-line window-based methods which scale like $O(|C|)$.

4 Experiments

4.1 Evaluation methods

We conduct experiments on the word analogy task of Mikolov et al. (2013a), a variety of word similarity tasks, as described in (Luong et al., 2013), and on the CoNLL-2003 shared benchmark dataset for NER (Tjong Kim Sang and De Meulder, 2003).

Table 2: Results on the word analogy task, given as percent accuracy. Underlined scores are best within groups of similarly-sized models; bold scores are best overall. HPCA vectors are publicly available[2]; (i)vLBL results are from (Mnih et al., 2013); skip-gram (SG) and CBOW results are from (Mikolov et al., 2013a,b); we trained SG† and CBOW† using the `word2vec` tool[3]. See text for details and a description of the SVD models.

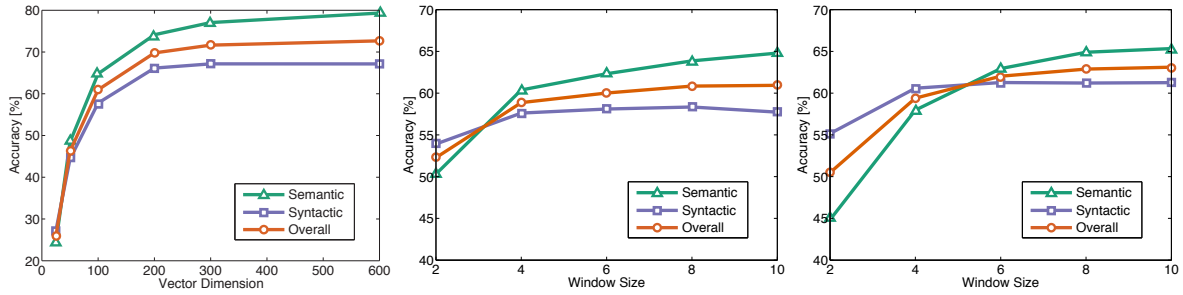
Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

Word analogies. The word analogy task consists of questions like, "a is to b as c is to __?" The dataset contains 19,544 such questions, divided into a semantic subset and a syntactic subset. The semantic questions are typically analogies about people or places, like "Athens is to Greece as Berlin is to __?". The syntactic questions are typically analogies about verb tenses or forms of adjectives, for example "dance is to dancing as fly is to __?". To correctly answer the question, the model should uniquely identify the missing term, with only an exact correspondence counted as a correct match. We answer the question "a is to b as c is to __?" by finding the word d whose representation w_d is closest to $w_b - w_a + w_c$ according to the cosine similarity.[4]

[2] http://lebret.ch/words/

[3] http://code.google.com/p/word2vec/

[4] Levy et al. (2014) introduce a multiplicative analogy evaluation, 3CosMul, and report an accuracy of 68.24% on



(a) Symmetric context (b) Symmetric context (c) Asymmetric context

Figure 2: Accuracy on the analogy task as function of vector size and window size/type. All models are trained on the 6 billion token corpus. In (a), the window size is 10. In (b) and (c), the vector size is 100.

Word similarity. While the analogy task is our primary focus since it tests for interesting vector space substructures, we also evaluate our model on a variety of word similarity tasks in Table 3. These include WordSim-353 (Finkelstein et al., 2001), MC (Miller and Charles, 1991), RG (Rubenstein and Goodenough, 1965), SCWS (Huang et al., 2012), and RW (Luong et al., 2013).

Named entity recognition. The CoNLL-2003 English benchmark dataset for NER is a collection of documents from Reuters newswire articles, annotated with four entity types: person, location, organization, and miscellaneous. We train models on CoNLL-03 training data on test on three datasets: 1) ConLL-03 testing data, 2) ACE Phase 2 (2001-02) and ACE-2003 data, and 3) MUC7 Formal Run test set. We adopt the BIO2 annotation standard, as well as all the preprocessing steps described in (Wang and Manning, 2013). We use a comprehensive set of discrete features that comes with the standard distribution of the Stanford NER model (Finkel et al., 2005). A total of 437,905 discrete features were generated for the CoNLL-2003 training dataset. In addition, 50-dimensional vectors for each word of a five-word context are added and used as continuous features. With these features as input, we trained a conditional random field (CRF) with exactly the same setup as the CRF_{join} model of (Wang and Manning, 2013).

4.2 Corpora and training details

We trained our model on five corpora of varying sizes: a 2010 Wikipedia dump with 1 billion tokens; a 2014 Wikipedia dump with 1.6 billion tokens; Gigaword 5 which has 4.3 billion tokens; the combination Gigaword5 + Wikipedia2014, which has 6 billion tokens; and on 42 billion tokens of web data, from Common Crawl[5]. We tokenize and lowercase each corpus with the Stanford tokenizer, build a vocabulary of the 400,000 most frequent words[6], and then construct a matrix of co-occurrence counts X. In constructing X, we must choose how large the context window should be and whether to distinguish left context from right context. We explore the effect of these choices below. In all cases we use a decreasing weighting function, so that word pairs that are d words apart contribute $1/d$ to the total count. This is one way to account for the fact that very distant word pairs are expected to contain less relevant information about the words' relationship to one another.

For all our experiments, we set $x_{\text{max}} = 100$, $\alpha = 3/4$, and train the model using AdaGrad (Duchi et al., 2011), stochastically sampling nonzero elements from X, with initial learning rate of 0.05. We run 50 iterations for vectors smaller than 300 dimensions, and 100 iterations otherwise (see Section 4.6 for more details about the convergence rate). Unless otherwise noted, we use a context of ten words to the left and ten words to the right.

The model generates two sets of word vectors, W and $\tilde{W}$. When X is symmetric, W and $\tilde{W}$ are equivalent and differ only as a result of their random initializations; the two sets of vectors should perform equivalently. On the other hand, there is evidence that for certain types of neural networks, training multiple instances of the network and then combining the results can help reduce overfitting and noise and generally improve results (Ciresan et al., 2012). With this in mind, we choose to use

the analogy task. This number is evaluated on a subset of the dataset so it is not included in Table 2. 3CosMul performed worse than cosine similarity in almost all of our experiments.

[5]To demonstrate the scalability of the model, we also trained it on a much larger sixth corpus, containing 840 billion tokens of web data, but in this case we did not lowercase the vocabulary, so the results are not directly comparable.

[6]For the model trained on Common Crawl data, we use a larger vocabulary of about 2 million words.

the sum $W + \tilde{W}$ as our word vectors. Doing so typically gives a small boost in performance, with the biggest increase in the semantic analogy task.

We compare with the published results of a variety of state-of-the-art models, as well as with our own results produced using the `word2vec` tool and with several baselines using SVDs. With `word2vec`, we train the skip-gram (SG†) and continuous bag-of-words (CBOW†) models on the 6 billion token corpus (Wikipedia 2014 + Gigaword 5) with a vocabulary of the top 400,000 most frequent words and a context window size of 10. We used 10 negative samples, which we show in Section 4.6 to be a good choice for this corpus.

For the SVD baselines, we generate a truncated matrix X_{trunc} which retains the information of how frequently each word occurs with only the top 10,000 most frequent words. This step is typical of many matrix-factorization-based methods as the extra columns can contribute a disproportionate number of zero entries and the methods are otherwise computationally expensive.

The singular vectors of this matrix constitute the baseline "SVD". We also evaluate two related baselines: "SVD-S" in which we take the SVD of $\sqrt{X_{\text{trunc}}}$, and "SVD-L" in which we take the SVD of $\log(1+X_{\text{trunc}})$. Both methods help compress the otherwise large range of values in X.[7]

4.3 Results

We present results on the word analogy task in Table 2. The GloVe model performs significantly better than the other baselines, often with smaller vector sizes and smaller corpora. Our results using the `word2vec` tool are somewhat better than most of the previously published results. This is due to a number of factors, including our choice to use negative sampling (which typically works better than the hierarchical softmax), the number of negative samples, and the choice of the corpus.

We demonstrate that the model can easily be trained on a large 42 billion token corpus, with a substantial corresponding performance boost. We note that increasing the corpus size does not guarantee improved results for other models, as can be seen by the decreased performance of the SVD-L model on this larger corpus. The fact that this basic SVD model does not scale well to large corpora lends further evidence to the necessity of the type of weighting scheme proposed in our model.

Table 3: Spearman rank correlation on word similarity tasks. All vectors are 300-dimensional. The CBOW* vectors are from the `word2vec` website and differ in that they contain phrase vectors.

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

Table 3 shows results on five different word similarity datasets. A similarity score is obtained from the word vectors by first normalizing each feature across the vocabulary and then calculating the cosine similarity. We compute Spearman's rank correlation coefficient between this score and the human judgments. CBOW* denotes the vectors available on the `word2vec` website that are trained with word and phrase vectors on 100B words of news data. GloVe outperforms it while using a corpus less than half the size.

Table 4 shows results on the NER task with the CRF-based model. The L-BFGS training terminates when no improvement has been achieved on the dev set for 25 iterations. Otherwise all configurations are identical to those used by Wang and Manning (2013). The model labeled *Discrete* is the baseline using a comprehensive set of discrete features that comes with the standard distribution of the Stanford NER model, but with no word vector features. In addition to the HPCA and SVD models discussed previously, we also compare to the models of Huang et al. (2012) (HSMN) and Collobert and Weston (2008) (CW). We trained the CBOW model using the `word2vec` tool[8]. The GloVe model outperforms all other methods on all evaluation metrics, except for the CoNLL test set, on which the HPCA method does slightly better. We conclude that the GloVe vectors are useful in downstream NLP tasks, as was first

[7] We also investigated several other weighting schemes for transforming X; what we report here performed best. Many weighting schemes like PPMI destroy the sparsity of X and therefore cannot feasibly be used with large vocabularies. With smaller vocabularies, these information-theoretic transformations do indeed work well on word similarity measures, but they perform very poorly on the word analogy task.

[8] We use the same parameters as above, except in this case we found 5 negative samples to work slightly better than 10.

Table 4: F1 score on NER task with 50d vectors. *Discrete* is the baseline without word vectors. We use publicly-available vectors for HPCA, HSMN, and CW. See text for details.

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

shown for neural vectors in (Turian et al., 2010).

4.4 Model Analysis: Vector Length and Context Size

In Fig. 2, we show the results of experiments that vary vector length and context window. A context window that extends to the left and right of a target word will be called symmetric, and one which extends only to the left will be called asymmetric. In (a), we observe diminishing returns for vectors larger than about 200 dimensions. In (b) and (c), we examine the effect of varying the window size for symmetric and asymmetric context windows. Performance is better on the syntactic subtask for small and asymmetric context windows, which aligns with the intuition that syntactic information is mostly drawn from the immediate context and can depend strongly on word order. Semantic information, on the other hand, is more frequently non-local, and more of it is captured with larger window sizes.

4.5 Model Analysis: Corpus Size

In Fig. 3, we show performance on the word analogy task for 300-dimensional vectors trained on different corpora. On the syntactic subtask, there is a monotonic increase in performance as the corpus size increases. This is to be expected since larger corpora typically produce better statistics. Interestingly, the same trend is not true for the semantic subtask, where the models trained on the smaller Wikipedia corpora do better than those trained on the larger Gigaword corpus. This is likely due to the large number of city- and country-based analogies in the analogy dataset and the fact that Wikipedia has fairly comprehensive articles for most such locations. Moreover, Wikipedia's

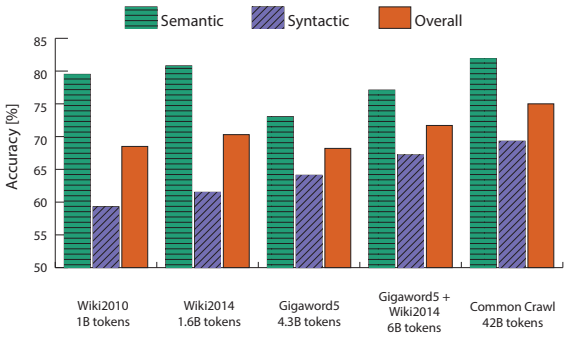


Figure 3: Accuracy on the analogy task for 300-dimensional vectors trained on different corpora.

entries are updated to assimilate new knowledge, whereas Gigaword is a fixed news repository with outdated and possibly incorrect information.

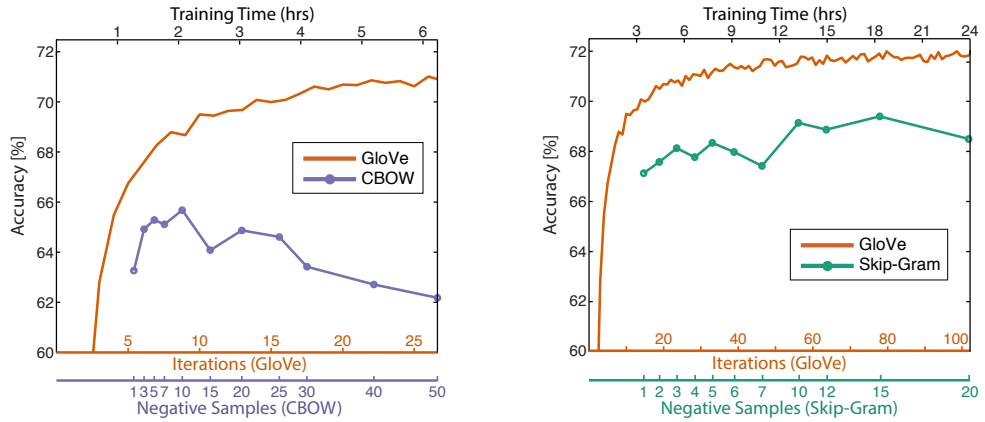
4.6 Model Analysis: Run-time

The total run-time is split between populating X and training the model. The former depends on many factors, including window size, vocabulary size, and corpus size. Though we did not do so, this step could easily be parallelized across multiple machines (see, e.g., Lebret and Collobert (2014) for some benchmarks). Using a single thread of a dual 2.1GHz Intel Xeon E5-2658 machine, populating X with a 10 word symmetric context window, a 400,000 word vocabulary, and a 6 billion token corpus takes about 85 minutes. Given X, the time it takes to train the model depends on the vector size and the number of iterations. For 300-dimensional vectors with the above settings (and using all 32 cores of the above machine), a single iteration takes 14 minutes. See Fig. 4 for a plot of the learning curve.

4.7 Model Analysis: Comparison with `word2vec`

A rigorous quantitative comparison of GloVe with `word2vec` is complicated by the existence of many parameters that have a strong effect on performance. We control for the main sources of variation that we identified in Sections 4.4 and 4.5 by setting the vector length, context window size, corpus, and vocabulary size to the configuration mentioned in the previous subsection.

The most important remaining variable to control for is training time. For GloVe, the relevant parameter is the number of training iterations. For `word2vec`, the obvious choice would be the number of training epochs. Unfortunately, the code is currently designed for only a single epoch:



(a) GloVe vs CBOW (b) GloVe vs Skip-Gram

Figure 4: Overall accuracy on the word analogy task as a function of training time, which is governed by the number of iterations for GloVe and by the number of negative samples for CBOW (a) and skip-gram (b). In all cases, we train 300-dimensional vectors on the same 6B token corpus (Wikipedia 2014 + Gigaword 5) with the same 400,000 word vocabulary, and use a symmetric context window of size 10.

it specifies a learning schedule specific to a single pass through the data, making a modification for multiple passes a non-trivial task. Another choice is to vary the number of negative samples. Adding negative samples effectively increases the number of training words seen by the model, so in some ways it is analogous to extra epochs.

We set any unspecified parameters to their default values, assuming that they are close to optimal, though we acknowledge that this simplification should be relaxed in a more thorough analysis.

In Fig. 4, we plot the overall performance on the analogy task as a function of training time. The two x-axes at the bottom indicate the corresponding number of training iterations for GloVe and negative samples for `word2vec`. We note that `word2vec`'s performance actually decreases if the number of negative samples increases beyond about 10. Presumably this is because the negative sampling method does not approximate the target probability distribution well.[9]

For the same corpus, vocabulary, window size, and training time, GloVe consistently outperforms `word2vec`. It achieves better results faster, and also obtains the best results irrespective of speed.

5 Conclusion

Recently, considerable attention has been focused on the question of whether distributional word representations are best learned from count-based methods or from prediction-based methods. Currently, prediction-based models garner substantial support; for example, Baroni et al. (2014) argue that these models perform better across a range of tasks. In this work we argue that the two classes of methods are not dramatically different at a fundamental level since they both probe the underlying co-occurrence statistics of the corpus, but the efficiency with which the count-based methods capture global statistics can be advantageous. We construct a model that utilizes this main benefit of count data while simultaneously capturing the meaningful linear substructures prevalent in recent log-bilinear prediction-based methods like `word2vec`. The result, GloVe, is a new global log-bilinear regression model for the unsupervised learning of word representations that outperforms other models on word analogy, word similarity, and named entity recognition tasks.

[9] In contrast, noise-contrastive estimation is an approximation which improves with more negative samples. In Table 1 of (Mnih et al., 2013), accuracy on the analogy task is a non-decreasing function of the number of negative samples.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. Stanford University gratefully acknowledges the support of the Defense Threat Reduction Agency (DTRA) under Air Force Research Laboratory (AFRL) contract no. FA8650-10-C-7020 and the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under AFRL contract no. FA8750-13-2-0040. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DTRA, AFRL, DEFT, or the US government.

References

Tom M. Apostol. 1976. *Introduction to Analytic Number Theory*. Introduction to Analytic Number Theory.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*.

Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *JMLR*, 3:1137–1155.

John A. Bullinaria and Joseph P. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39(3):510–526.

Dan C. Ciresan, Alessandro Giusti, Luca M. Gambardella, and Jürgen Schmidhuber. 2012. Deep neural networks segment neuronal membranes in electron microscopy images. In *NIPS*, pages 2852–2860.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *JMLR*, 12:2493–2537.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12.

Lev Finkelstein, Evgenly Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *ACL*.

Rémi Lebret and Ronan Collobert. 2014. Word embeddings through Hellinger PCA. In *EACL*.

Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. 2014. Linguistic regularities in sparse and explicit word representations. *CoNLL-2014*.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instrumentation, and Computers*, 28:203–208.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *ICLR Workshop Papers*.

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

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *HLT-NAACL*.

George A. Miller and Walter G. Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *NIPS*.

Douglas L. T. Rohde, Laura M. Gonnerman, and David C. Plaut. 2006. An improved model of semantic similarity based on lexical co-occurence. *Communications of the ACM*, 8:627–633.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34:1–47.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing With Compositional Vector Grammars. In *ACL*.

Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the SIGIR Conference on Research and Development in Informaion Retrieval*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL-2003*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

Mengqiu Wang and Christopher D. Manning. 2013. Effect of non-linear deep architecture in sequence labeling. In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP)*.

Jointly Learning Word Representations and Composition Functions Using Predicate-Argument Structures

Kazuma Hashimoto[†], **Pontus Stenetorp**[†], **Makoto Miwa**[‡], **and Yoshimasa Tsuruoka**[†]
†The University of Tokyo, 3-7-1 Hongo, Bunkyo-ku, Tokyo, Japan
{hassy,pontus,tsuruoka}@logos.t.u-tokyo.ac.jp
‡Toyota Technological Institute, 2-12-1 Hisakata, Tempaku-ku, Nagoya, Japan
makoto-miwa@toyota-ti.ac.jp

Abstract

We introduce a novel compositional language model that works on Predicate-Argument Structures (PASs). Our model jointly learns word representations and their composition functions using bag-of-words and dependency-based contexts. Unlike previous word-sequence-based models, our PAS-based model composes arguments into predicates by using the category information from the PAS. This enables our model to capture long-range dependencies between words and to better handle constructs such as verb-object and subject-verb-object relations. We verify this experimentally using two phrase similarity datasets and achieve results comparable to or higher than the previous best results. Our system achieves these results without the need for pre-trained word vectors and using a much smaller training corpus; despite this, for the subject-verb-object dataset our model improves upon the state of the art by as much as $\sim$10% in relative performance.

1 Introduction

Studies on embedding single words in a vector space have made notable successes in capturing their syntactic and semantic properties (Turney and Pantel, 2010). These embeddings have also been found to be a useful component for Natural Language Processing (NLP) systems; for example, Turian et al. (2010) and Collobert et al. (2011) demonstrated how low-dimensional word vectors learned by Neural Network Language Models (NNLMs) are beneficial for a wide range of NLP tasks.

Recently, the main focus of research on vector space representation is shifting from word representations to phrase representations (Baroni and Zamparelli, 2010; Grefenstette and Sadrzadeh, 2011; Mitchell and Lapata, 2010; Socher et al., 2012). Combining the ideas of NNLMs and semantic composition, Tsubaki et al. (2013) introduced a novel NNLM incorporating verb-object dependencies. More recently, Levy and Goldberg (2014) presented a NNLM that integrated syntactic dependencies. However, to the best of our knowledge, there is no previous work on integrating a variety of syntactic and semantic dependencies into NNLMs in order to learn *composition functions* as well as word representations. The following question thus arises naturally:

> Can a variety of dependencies be used to jointly learn both stand-alone word vectors and their compositions, embedding them in the same vector space?

In this work, we bridge the gap between purely context-based (Levy and Goldberg, 2014; Mikolov et al., 2013b; Mnih and Kavukcuoglu, 2013) and compositional (Tsubaki et al., 2013) NNLMs by using the flexible set of categories from Predicate-Argument-Structures (PASs). More specifically, we propose a Compositional Log-Bilinear Language Model using PASs (PAS-CLBLM), an overview of which is shown in Figure 1. The model is trained by maximizing the accuracy of predicting target words from their bag-of-words *and* dependency-based context, which provides information about selectional preference. As shown in Figure 1 (b), one of the advantages of the PAS-CLBLM is that the model can treat not only word vectors but also composed vectors as contexts. Since the composed vectors

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1544–1555,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

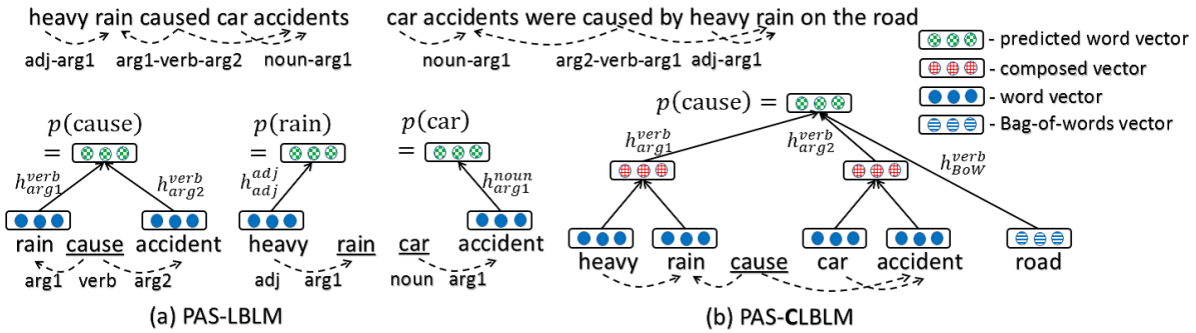


Figure 1: An overview of the proposed model: PAS-CLBLM. (a) The PAS-LBLM predicts target words from their bag-of-words and dependency-based context words. (b) The PAS-CLBLM predicts target words using not only context words but also composed vector representations derived from another level of predicate-argument structures. Underlined words are target words and we only depict the bag-of-words vector for the PAS-CLBLM.

are treated as input to the language model in the same way as word vectors, these composed vectors are expected to become similar to word vectors for words with similar meanings.

Our empirical results demonstrate that the proposed model has the ability to learn meaningful representations for adjective-noun, noun-noun, and (subject-) verb-object dependencies. On three tasks of measuring the semantic similarity between short phrases (adjective-noun, noun-noun, and verb-object), the learned composed vectors achieve scores (Spearman's rank correlation ρ) comparable to or higher than those of previous models. On a task involving more complex phrases (subject-verb-object), our learned composed vectors achieve state-of-the-art performance ($\rho = 0.50$) with a training corpus that is an order of magnitude smaller than that used by previous work (Tsubaki et al., 2013; Van de Cruys et al., 2013). Moreover, the proposed model does not require any pre-trained word vectors produced by external models, but rather induces word vectors jointly while training.

2 Related Work

There is a large body of work on how to represent the meaning of a word in a vector space. Distributional approaches assume that the meaning of a word is determined by the contexts in which it appears (Firth, 1957). The context of a word is often defined as the words appearing in a window of fixed-length (bag-of-words) and a simple approach is to treat the co-occurrence statistics of a word w as a vector representation for w (Mitchell and Lapata, 2008; Mitchell and Lapata, 2010); alternatively, dependencies between words can be used to define contexts (Goyal et al., 2013; Erk and Padó, 2008; Thater et al., 2010).

In contrast to distributional representations, NNLMs represent words in a low-dimensional vector space (Bengio et al., 2003; Collobert et al., 2011). Recently, Mikolov et al. (2013b) and Mnih and Kavukcuoglu (2013) proposed highly scalable models to learn high-dimensional word vectors. Levy and Goldberg (2014) extended the model of Mikolov et al. (2013b) by treating syntactic dependencies as contexts.

Mitchell and Lapata (2008) investigated a variety of compositional operators to combine word vectors into phrasal representations. Among these operators, simple element-wise addition and multiplication are now widely used to represent short phrases (Mitchell and Lapata, 2010; Blacoe and Lapata, 2012). The obvious limitation with these simple approaches is that information about word order and syntactic relations is lost.

To incorporate syntactic information into composition functions, a variety of compositional models have been proposed. These include recursive neural networks using phrase-structure trees (Socher et al., 2012; Socher et al., 2013b) and models in which words have a specific form of parameters according to their syntactic roles and composition functions are syntactically dependent on the relations of input words (Baroni and Zamparelli, 2010; Grefenstette and Sadrzadeh, 2011; Hashimoto et al., 2013; Hermann and Blunsom, 2013; Socher et al., 2013a).

More recently, syntactic dependency-based

compositional models have been proposed (Paperno et al., 2014; Socher et al., 2014; Tsubaki et al., 2013). One of the advantages of these models is that they are less restricted by word order. Among these, Tsubaki et al. (2013) introduced a novel compositional NNLM mainly focusing on verb-object dependencies and achieved state-of-the-art performance for the task of measuring the semantic similarity between subject-verb-object phrases.

3 PAS-CLBLM: A Compositional Log-Bilinear Language Model Using Predicate-Argument Structures

In some recent studies on representing words as vectors, word vectors are learned by solving word prediction tasks (Mikolov et al., 2013a; Mnih and Kavukcuoglu, 2013). More specifically, given target words and their context words, the basic idea is to train classifiers to discriminate between each target word and artificially induced negative target words. The feature vector of the classifiers are calculated using the context word vectors whose values are optimized during training. As a result, vectors of words in similar contexts become similar to each other.

Following these studies, we propose a novel model to jointly learn representations for words and their compositions by training word prediction classifiers using PASs. In this section, we first describe the predicate-argument structures as they serve as the basis of our model. We then introduce a Log-Bilinear Language Model using Predicate-Argument Structures (PAS-LBLM) to learn word representations using both bag-of-words and dependency-based contexts. Finally, we propose integrating compositions of words into the model. Figure 1 (b) shows the overview of the proposed model.

3.1 Predicate-Argument Structures

Due to advances in deep parsing technologies, syntactic parsers that can produce predicate-argument structures are becoming accurate and fast enough to be used for practical applications. In this work, we use the probabilistic HPSG parser *Enju* (Miyao and Tsujii, 2008) to obtain the predicate-argument structures of individual sentences. In its grammar, each word in a sentence is treated as a predicate of a certain category with zero or more arguments. Table 1 shows some examples of predicates of different categories.[1] For example, a predicate of the category *verb_arg12* expresses a verb with two arguments. A graph can be constructed by connecting words in predicate-argument structures in a sentence; in general, these graphs are acyclic.

Category	predicate	arg1	arg2
adj_arg1	heavy	rain	
noun_arg1	car	accident	
verb_arg12	cause	rain	accident
prep_arg12	at	eat	restaurant

Table 1: Examples of predicates of different categories from the grammar of the Enju parser. *arg1* and *arg2* denote the first and second arguments.

One of the merits of using predicate-argument structures is that they can capture dependencies between more than two words, while standard syntactic dependency structures are limited to dependencies between two words. For example, one of the predicates in the phrase "heavy rain caused car accidents" is the verb "cause", and it has two arguments ("rain" and "accident"). Furthermore, exactly the same predicate-argument structure (predicate: cause, first argument: rain, second argument: accident) is extracted from the passive form of the above phrase: "car accidents were caused by heavy rain". This is helpful when capturing semantic dependencies between predicates and arguments, and in extracting facts or relations described in a sentence, such as *who did what to whom*.

3.2 A Log-Bilinear Language Model Using Predicate-Argument Structures

3.2.1 PAS-based Word Prediction

The PAS-LBLM predicts a target word given its PAS-based context. We assume that each word w in the vocabulary $\mathbb{V}$ is represented with a d-dimensional vector $v(w)$. When a predicate of category c is extracted from a sentence, the PAS-LBLM computes the predicted d-dimensional vector $p(w_t)$ for the target word w_t from its context words $w_1, w_2, \ldots, w_m$:

$$p(w_t) = f\left(\sum_{i=1}^{m} h_i^c \odot v(w_i)\right), \qquad (1)$$

[1] The categories of the predicates in the Enju parser are summarized at `http://kmcs.nii.ac.jp/~yusuke/enju/enju-manual/enju-output-spec.html`.

where $h_i^c \in \mathbb{R}^{d \times 1}$ are category-specific weight vectors and $\odot$ denotes element-wise multiplication. f is a non-linearity function; in this work we define f as $\tanh$.

As an example following Figure 1 (a), when the predicate "cause" is extracted with its first and second arguments "rain" and "accident", the PAS-LBLM computes $p(\text{cause}) \in \mathbb{R}^d$ following Eq. (1):

$$p(\text{cause}) = f(h_{\text{arg1}}^{\text{verb\_arg12}} \odot v(\text{rain}) + h_{\text{arg2}}^{\text{verb\_arg12}} \odot v(\text{accident})). \quad (2)$$

In Eq. (2), the predicate is treated as the target word, and its arguments are treated as the context words. In the same way, an argument can be treated as a target word:

$$p(\text{rain}) = f(h_{\text{verb}}^{\text{verb\_arg12}} \odot v(\text{cause}) + h_{\text{arg2}}^{\text{verb\_arg12}} \odot v(\text{accident})). \quad (3)$$

Relationship to previous work. If we omit the the category-specific weight vectors h_i^c in Eq. (1), our model is similar to the CBOW model in Mikolov et al. (2013a). CBOW predicts a target word given its surrounding bag-of-words context, while our model uses its PAS-based context. To incorporate the PAS information in our model more efficiently, we use category-specific weight vectors. Similarly, the vLBL model of Mnih and Kavukcuoglu (2013) uses different weight vectors depending on the position relative to the target word. As with previous neural network language models (Collobert et al., 2011; Huang et al., 2012), our model and vLBL can use weight matrices rather than weight vectors. However, as discussed by Mnih and Teh (2012), using weight vectors makes the training significantly faster than using weight matrices. Despite the simple formulation of the element-wise operations, the category-specific weight vectors efficiently propagate PAS-based context information as explained next.

3.2.2 Training Word Vectors

To train the PAS-LBLM, we use a scoring function to evaluate how well the target word w_t fits the given context:

$$s(w_t, p(w_t)) = \tilde{v}(w_t)^{\mathrm{T}} p(w_t), \quad (4)$$

where $\tilde{v}(w_t) \in \mathbb{R}^{d \times 1}$ is the scoring weight vector for w_t. Thus, the model parameters in the PAS-LBLM are $(V, \tilde{V}, H)$. V is the set of word vectors $v(w)$, and $\tilde{V}$ is the set of scoring weight vectors $\tilde{v}(w)$. H is the set of the predicate-category-specific weight vectors h_i^c.

Based on the objective in the model of Collobert et al. (2011), the model parameters are learned by minimizing the following hinge loss:

$$\sum_{n=1}^{N} \max(1 - s(w_t, p(w_t)) + s(w_n, p(w_t)), 0), \quad (5)$$

where the negative sample w_n is a randomly sampled word other than w_t, and N is the number of negative samples. In our experiments we set $N = 1$. Following Mikolov et al. (2013b), negative samples were drawn from the distribution over unigrams that we raise to the power 0.75 and then normalize to once again attain a probability distribution. We minimize the loss function in Eq. (5) using AdaGrad (Duchi et al., 2011). For further training details, see Section 4.5.

Relationship to softmax regression models. The model parameters can be learned by maximizing the log probability of the target word w_t based on the softmax function:

$$p(w_t|context) = \frac{\exp(s(w_t, p(w_t)))}{\sum_{i=1}^{|\mathbb{V}|} \exp(s(w_i, p(w_t)))}. \quad (6)$$

This is equivalent to a softmax regression model. However, when the vocabulary $\mathbb{V}$ is large, computing the softmax function in Eq. (6) is computationally expensive. If we do not need probability distributions over words, we are not necessarily restricted to using the probabilistic expressions. Recently, several methods have been proposed to efficiently learn word representations rather than accurate language models (Collobert et al., 2011; Mikolov et al., 2013b; Mnih and Kavukcuoglu, 2013), and our objective follows the work of Collobert et al. (2011). Mikolov et al. (2013b) and Mnih and Kavukcuoglu (2013) trained their models using word-dependent scoring weight vectors which are the arguments of our scoring function in Eq. (4). During development we also trained our model using the negative sampling technique of Mikolov et al. (2013b); however, we did not observe any significant performance difference.

Intuition behind the PAS-LBLM. Here we briefly explain how each class of the model parameters of the PAS-LBLM contributes to learning word representations at each stochastic gradient

decent step. The category-specific weight vectors provide the PAS information for context word vectors which we would like to learn. During training, context word vectors having the same PAS-based syntactic roles are updated similarly. The word-dependent scoring weight vectors propagate the information on which words should, or should not, be predicted. In effect, context word vectors making similar contributions to word predictions are updated similarly. The non-linear function f provides context words with information on the other context words in the same PAS. In this way, word vectors are expected to be learned efficiently by the PAS-LBLM.

3.3 Learning Composition Functions

As explained in Section 3.1, predicate-argument structures inherently form graphs whose nodes are words in a sentence. Using the graphs, we can integrate relationships between multiple predicate-argument structures into our model.

When the context word w_i in Eq. (1), excluding predicate words, has another predicate-argument of category c' as a dependency, we replace $v(w_i)$ with the vector produced by the composition function for the predicate category c'. For example, as shown in Figure 1 (b), when the first argument "rain" of the predicate "cause" is also the argument of the predicate "heavy", we first compute the d-dimensional composed vector representation for "heavy" and "rain":

$$g_{c'}(v(\mathrm{heavy}), v(\mathrm{rain})), \tag{7}$$

where c' is the category *adj_arg1*, and $g_{c'}$ is a function to combine input vectors for the predicate-category c'. We can use any composition function that produces a representation of the same dimensionality as its inputs, such as element-wise addition/multiplication (Mitchell and Lapata, 2008) or neural networks (Socher et al., 2012). We then replace $v(\mathrm{rain})$ in Eq. (2) with $g_{c'}(v(\mathrm{heavy}), v(\mathrm{rain}))$. When the second argument "accident" in Eq. (2) is also the argument of the predicate "car", $v(\mathrm{accident})$ is replaced with $g_{c''}(v(\mathrm{car}), v(\mathrm{accident}))$. c'' is the predicate category *noun_arg1*. These multiple relationships of predicate-argument structures should provide richer context information. We refer to the PAS-LBLM with composition functions as PAS-**C**LBLM.

3.4 Bag-of-Words Sensitive PAS-CLBLM

Both the PAS-LBLM and PAS-CLBLM can take meaningful relationships between words into account. However, at times, the number of context words can be limited and the ability of other models to take ten or more words from a fixed context in a bag-of-words (BoW) fashion could compensate for this sparseness. Huang et al. (2012) combined local and global contexts in their neural network language models, and motivated by their work, we integrate bag-of-words vectors into our models. Concretely, we add an additional input term to Eq. (1):

$$p(w_t) = f\left(\sum_{i=1}^{m} h_i^c \odot v(w_i) + h_{\mathrm{BoW}}^c \odot v(\mathrm{BoW})\right), \tag{8}$$

where $h_{\mathrm{BoW}}^c \in \mathbb{R}^{d\times 1}$ are additional weight vectors, and $v(\mathrm{BoW}) \in \mathbb{R}^{d\times 1}$ is the average of the word vectors in the same sentence. To construct the $v(\mathrm{BoW})$ for each sentence, we average the word vectors of nouns and verbs in the same sentence, excluding the target and context words.

4 Experimental Settings

4.1 Training Corpus

We used the British National Corpus (BNC) as our training corpus, extracted 6 million sentences from the original BNC files, and parsed them using the Enju parser described in Section 3.1.

4.2 Word Sense Disambiguation Using Part-of-Speech Tags

In general, words can have multiple syntactic usages. For example, the word *cause* can be a noun or a verb depending on its context. Most of the previous work on learning word vectors ignores this ambiguity since word sense disambiguation could potentially be performed after the word vectors have been trained (Huang et al., 2012; Kartsaklis and Sadrzadeh, 2013). Some recent work explicitly assigns an independent vector for each word usage according to its part-of-speech (POS) tag (Hashimoto et al., 2013; Kartsaklis and Sadrzadeh, 2013). Alternatively, Baroni and Zamparelli (2010) assigned different forms of parameters to adjectives and nouns.

In our experiments, we combined each word with its corresponding POS tags. We used the base-forms provided by the Enju parser rather than

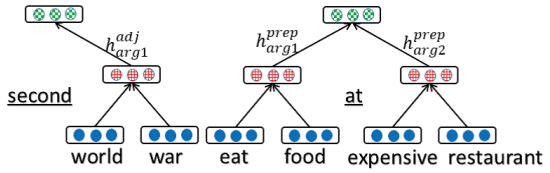


Figure 2: Two PAS-CLBLM training samples.

the surface-forms, and used the first two characters of the POS tags. For example, *VB*, *VBP*, *VBZ*, *VBG*, *VBD*, *VBN* were all mapped to *VB*. This resulted in two kinds of *cause*: *cause_NN* and *cause_VB* and we used the 100,000 most frequent lowercased word-POS pairs in the BNC.

4.3 Selection of Training Samples Based on Categories of Predicates

To train the PAS-LBLM and PAS-CLBLM, we could use all predicate categories. However, our preliminary experiments showed that these categories covered many training samples which are not directly relevant to our experimental setting, such as determiner-noun dependencies. We thus manually selected the categories used in our experiments. The selected predicates are listed in Table 1: *adj_arg1*, *noun_arg1*, *prep_arg12*, and *verb_arg12*. These categories should provide meaningful information on selectional preference. For example, the *prep_arg12* denotes prepositions with two arguments, such as "eat at restaurant" which means that the verb "eat" is related to the noun "restaurant" by the preposition "at". Prepositions are one of the predicates whose arguments can be verbs, and thus prepositions are important in training the composition functions for (subject-) verb-object dependencies as described in the next paragraph.

Another point we had to consider was how to construct the training samples for the PAS-CLBLM. We constructed compositional training samples as explained in Section 3.3 when c' was *adj_arg1*, *noun_arg1*, or *verb_arg12*. Figure 2 shows two examples in addition to the example in Figure 1 (b). Using such training samples, the PAS-CLBLM could, for example, recognize from the two predicate-argument structures, "eat food" and "eat at restaurant", that eating foods is an action that occurs at restaurants.

Model	Composition Function
Add_l	$v(w_1) + v(w_2)$
Add_nl	$\tanh(v(w_1) + v(w_2))$
Wadd_l	$m^c_\text{adj} \odot v(w_1) + m^c_\text{arg1} \odot v(w_2)$
Wadd_nl	$\tanh(m^c_\text{adj} \odot v(w_1) + m^c_\text{arg1} \odot v(w_2))$

Table 2: Composition functions used in this work. The examples are shown as the *adjective-noun* dependency between w_1 ="heavy" and w_2 ="rain".

4.4 Selection of Composition Functions

As described in Section 3.3, we are free to select any composition functions in Eq. (7). To maintain the fast training speed of the PAS-LBLM, we avoid dense matrix-vector multiplication in our composition functions. In Table 2, we list the composition functions used for the PAS-CLBLM. $\textbf{Add}_\text{l}$ is element-wise addition and $\textbf{Add}_\text{nl}$ is element-wise addition with the non-linear function $\tanh$. The subscripts l and nl denote the words *linear* and *non-linear*. Similarly, $\textbf{Wadd}_\text{l}$ is element-wise weighted addition and $\textbf{Wadd}_\text{nl}$ is element-wise weighted addition with the non-linear function $\tanh$. The weight vectors $m^c_i \in \mathbb{R}^{d \times 1}$ in Table 2 are predicate-category-specific parameters which are learned during training. We investigate the effects of the non-linear function $\tanh$ for these composition functions. In the formulations of the backpropagation algorithm, non-linear functions allow the input vectors to weakly interact with each other.

4.5 Initialization and Optimization of Model Parameters

We assigned a 50-dimensional vector for each word-POS pair described in Section 4.2 and initialized the vectors and the scoring weight vectors using small random values. In part inspired by the initialization method of the weight matrices in Socher et al. (2013a), we initialized all values in the compositional weight vectors of the Wadd_l and Wadd_nl as 1.0. The context weight vectors were initialized using small random values.

We minimized the loss function in Eq. (5) using mini-batch SGD and AdaGrad (Duchi et al., 2011). Using AdaGrad, the SGD's learning rate is adapted independently for each model parameter. This is helpful in training the PAS-LBLM and PAS-CLBLM, as they have conditionally dependent model parameters with varying frequencies.

The mini-batch size was 32 and the learning rate was 0.05 for each experiment, and no regularization was used. To verify the semantics captured by the proposed models during training and to tune the hyperparameters, we used the *WordSim-353*[2] word similarity data set (Finkelstein et al., 2001).

5 Evaluation on Phrase Similarity Tasks

5.1 Evaluation Settings

The learned models were evaluated on four tasks of measuring the semantic similarity between short phrases. We performed evaluation using the three tasks (AN, NN, and VO) in the dataset[3] provided by Mitchell and Lapata (2010), and the SVO task in the dataset[4] provided by Grefenstette and Sadrzadeh (2011).

The datasets include pairs of short phrases extracted from the BNC. AN, NN, and VO contain 108 phrase pairs of adjective-noun, noun-noun, and verb-object. SVO contains 200 pairs of subject-verb-object phrases. Each phrase pair has multiple human-ratings: the higher the rating is, the more semantically similar the phrases. For example, the subject-verb-object phrase pair of "student write name" and "student spell name" has a high rating. The pair "people try door" and "people judge door" has a low rating.

For evaluation we used the Spearman's rank correlation ρ between the human-ratings and the cosine similarity between the composed vector pairs. We mainly used *non-averaged* human-ratings for each pair, and as described in Section 5.3, we also used *averaged* human-ratings for the SVO task. Each phrase pair in the datasets was annotated by more than two annotators. In the case of averaged human ratings, we averaged multiple human-ratings for each phrase pair, and in the case of non-averaged human-ratings, we treated each human-rating as a separate annotation.

With the PAS-CLBLM, we represented each phrase using the composition functions listed in Table 2. When there was no composition present, we represented the phrase using element-wise addition. For example, when we trained the PAS-CLBLM with the composition function Wadd_{nl}, the composed vector for each phrase was computed using the Wadd_{nl} function, and when we trained the PAS-LBLM, we used the element-wise addition function. To compute the composed vectors using the Wadd_{l} and Wadd_{nl} functions, we used the categories of the predicates *adj_arg1*, *noun_arg1*, and *verb_arg12* listed in Table 1.

Model	AN	NN	VO
PAS-CLBLM (Add_{l})	**0.52**	0.44	0.35
PAS-CLBLM (Add_{nl})	**0.52**	0.46	**0.45**
PAS-CLBLM (Wadd_{l})	0.48	0.39	0.34
PAS-CLBLM (Wadd_{nl})	0.48	0.40	0.39
PAS-LBLM	0.41	0.44	0.39
word2vec	**0.52**	0.48	0.42
BL w/ BNC	0.48	**0.50**	0.35
HB w/ BNC	0.41	0.44	0.34
KS w/ ukWaC	n/a	n/a	**0.45**
K w/ BNC	n/a	n/a	0.41
Human agreement	0.52	0.49	0.55

Table 3: Spearman's rank correlation scores ρ for the three tasks: AN, NN, and VO.

As a strong baseline, we trained the *Skip-gram* model of Mikolov et al. (2013b) using the publicly available *word2vec*[5] software. We fed the POS-tagged BNC into word2vec since our models utilize POS tags and trained 50-dimensional word vectors using word2vec. For each phrase we then computed the representation using vector addition.

5.2 AN, NN, and VO Tasks

Table 3 shows the correlation scores ρ for the AN, NN, and VO tasks. *Human agreement* denotes the inter-annotator agreement. The word2vec baseline achieves unexpectedly high scores for these three tasks. Previously these kinds of models (Mikolov et al., 2013b; Mnih and Kavukcuoglu, 2013) have mainly been evaluated for word analogy tasks and, to date, there has been no work using these word vectors for the task of measuring the semantic similarity between phrases. However, this experimental result suggests that word2vec can serve as a strong baseline for these kinds of tasks, in addition to word analogy tasks.

In Table 3, **BL**, **HB**, **KS**, and **K** denote the work of Blacoe and Lapata (2012), Hermann and Blunsom (2013), Kartsaklis and Sadrzadeh (2013), and Kartsaklis et al. (2013) respectively. Among these,

[2] http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

[3] http://homepages.inf.ed.ac.uk/s0453356/share

[4] http://www.cs.ox.ac.uk/activities/compdistmeaning/GS2011data.txt

[5] https://code.google.com/p/word2vec/

Model	Corpus	Averaged		Non-averaged	
		SVO-SVO	SVO-V	SVO-SVO	SVO-V
PAS-CLBLM (Add_l)	BNC	0.29	0.34	0.24	0.28
PAS-CLBLM (Add_nl)		0.27	0.32	0.24	0.28
PAS-CLBLM (Wadd_l)		0.25	0.26	0.21	0.23
PAS-CLBLM (Wadd_nl)		**0.42**	**0.50**	**0.34**	**0.41**
PAS-LBLM		0.21	0.06	0.18	0.08
word2vec	BNC	0.12	0.32	0.12	0.28
Grefenstette and Sadrzadeh (2011)	BNC	n/a	n/a	0.21	n/a
Tsubaki et al. (2013)	ukWaC	n/a	0.47	n/a	n/a
Van de Cruys et al. (2013)	ukWaC	n/a	n/a	0.32	0.37
Human agreement		0.75		0.62	

Table 4: Spearman's rank correlation scores ρ for the SVO task. *Averaged* denotes the ρ calculated by averaged human ratings, and *Non-averaged* denotes the ρ calculated by non-averaged human ratings.

only Kartsaklis and Sadrzadeh (2013) used the ukWaC corpus (Baroni et al., 2009) which is an order of magnitude larger than the BNC. As we can see in Table 3, the PAS-CLBLM (Add_nl) achieves scores comparable to and higher than those of the baseline and the previous state-of-the-art results. In relation to these results, the Wadd_l and Wadd_nl variants of the PAS-CLBLM do not achieve great improvements in performance. This indicates that simple word vector addition can be sufficient to compose representations for phrases consisting of word pairs.

5.3 SVO Task

Table 4 shows the correlation scores ρ for the SVO task. The scores ρ for this task are reported for both *averaged* and *non-averaged* human ratings. This is due to a disagreement in previous work regarding which metric to use when reporting results. Hence, we report the scores for both settings in Table 4. Another point we should consider is that some previous work reported scores based on the similarity between composed representations (Grefenstette and Sadrzadeh, 2011; Van de Cruys et al., 2013), and others reported scores based on the similarity between composed representations and word representations of landmark verbs from the dataset (Tsubaki et al., 2013; Van de Cruys et al., 2013). For completeness, we report the scores for both settings: *SVO-SVO* and *SVO-V* in Table 4.

The results show that the weighted addition model with the non-linear function $\tanh$ (PAS-CLBLM (Wadd_nl)) is effective for the more complex phrase task. While simple vector addition is sufficient for phrases consisting of word pairs, it is clear from our experimental results that they fall short for more complex structures such as those involved in the SVO task.

Our PAS-CLBLM (Wadd_nl) model outperforms the previous state-of-the-art scores for the SVO task as reported by Tsubaki et al. (2013) and Van de Cruys et al. (2013). As such, there are three key points that we would like to emphasize:

(1) the difference of the training corpus size,

(2) the necessity of the pre-trained word vectors,

(3) the modularity of deep learning models.

Tsubaki et al. (2013) and Van de Cruys et al. (2013) used the ukWaC corpus. This means our model works better, despite using a considerably smaller corpora. It should also be noted that, like us, Grefenstette and Sadrzadeh (2011) used the BNC corpus.

The model of Tsubaki et al. (2013) is based on neural network language models which use syntactic dependencies between verbs and their objects. While their novel model, which incorporates the idea of *co-compositionality*, works well with pre-trained word vectors produced by external models, it is not clear whether the pre-trained vectors are required to achieve high scores. In contrast, we have achieved state-of-the-art results without the use of pre-trained word vectors.

Despite our model's scalability, we trained 50-dimensional vector representations for words and their composition functions and achieved high scores using this low dimensional vector space.

model	d	AN	NN	VO	SVO
Add_l	50	0.52	0.44	0.35	0.24
	1000	0.51	**0.51**	0.43	0.31
Add_nl	50	0.52	0.46	0.45	0.24
	1000	0.51	0.50	0.45	0.31
Wadd_l	50	0.48	0.39	0.34	0.21
	1000	0.50	0.49	0.43	0.32
Wadd_nl	50	0.48	0.40	0.39	0.34
	1000	0.51	0.48	**0.48**	0.34

Table 5: Comparison of the PAS-CLBLM between $d = 50$ and $d = 1000$.

model	BoW	AN	NN	VO	SVO
Add_l	w/	0.52	0.44	0.35	0.24
	w/o	0.48	0.46	0.38	0.23
Add_nl	w/	0.52	0.46	**0.45**	0.24
	w/o	0.50	0.47	0.41	0.15
Wadd_l	w/	0.48	0.39	0.34	0.21
	w/o	0.47	0.39	0.38	0.21
Wadd_nl	w/	0.48	0.40	0.39	**0.34**
	w/o	0.52	0.42	0.33	0.26

Table 6: Scores of the PAS-CLBLM with and without BoW contexts.

This maintains the possibility to incorporate recently developed deep learning composition functions into our models, such as recursive neural tensor networks (Socher et al., 2013b) and co-compositional neural networks (Tsubaki et al., 2013). While such complex composition functions slow down the training of compositional models, richer information could be captured during training.

5.4 Effects of the Dimensionality

To see how the dimensionality of the word vectors affects the scores, we trained the PAS-CLBLM for each setting using 1,000-dimensional word vectors and set the learning rate to 0.01. Table 5 shows the scores for all four tasks. Note that we only report the scores for the setting *non-averaged SVO-SVO* here. As shown in Table 5, the scores consistently improved with a few exceptions. The scores $\rho = 0.51$ for the NN task and $\rho = 0.48$ for the VO task are the best results to date. However, the score $\rho = 0.34$ for the SVO task did not improve by increasing the dimensionality. This means that simply increasing the dimensionality of the word vectors does not necessarily lead to better results for complex phrases.

5.5 Effects of Bag-of-Words Contexts

Lastly, we trained the PAS-CLBLM without the bag-of-words contexts described in Section 3.4 and used 50-dimensional word vectors. As can be seen in Table 6, large score improvements were observed only for the VO and SVO tasks by including the bag-of-words contexts and the non-linearity function. It is likely that the results depend on how the bag-of-words contexts are constructed. However, we leave this line of analysis as future work. Both adjective-noun and noun-noun phrase are noun phrases, and (subject-) verb-object phrases can be regarded as complete sentences. Therefore, different kinds of context information might be required for both groups.

6 Qualitative Analysis on Composed Vectors

An open question that remains is to what extent composition affects the representations produced by our PAS-CLBLM model. To evaluate this we assigned a vector for each composed representation. For example, the adjective-noun dependency "heavy rain" would be assigned an independent vector. We added the most frequent 100,000 adjective-noun, noun-noun, and (subject-) verb-object tuples to the vocabulary and the resulting vocabulary contained 400,000 tokens (100,000+3$\times$100,000). A similar method for treating frequent neighboring words as single words was introduced by Mikolov et al. (2013b). However, some dependencies, such as (subject-) verb-object phrases, are not always captured when considering only neighboring words.

Table 7 (*No composition*) shows some examples of predicate-argument dependencies with their closest neighbors in the vector space according to the cosine similarity. The table shows that the learned vectors of multiple words capture semantic similarity. For example, the vector of "heavy rain" is close to the vectors of words which express the phenomena *heavily raining*. The vector of "new york" captures the concept of a *major city*. The vectors of (subject-) verb-object dependencies also capture the semantic similarity, which is the main difference to previous approaches, such as that of Mikolov et al. (2013b), which only consider neighboring words. These results suggest that the PAS-CLBLM can learn meaningful composition

Query	No composition	Composition
(AN) heavy rain	rain	rain
	thunderstorm	sunshine
	downpour	storm
	blizzard	drizzle
	much rain	chill
(AN) chief executive	general manager	executive
	vice president	director
	executive director	representative
	project manager	officer
	managing director	administrator
(NN) world war	second war	war
	plane crash	world
	riot	race
	last war	holocaust
	great war	warfare
(NN) new york	oslo	york
	paris	toronto
	birmingham	paris
	moscow	edinburgh
	madrid	glasgow
(VO) make payment	make order	make
	carry survey	allow
	pay tax	demand
	pay	produce
	impose tax	bring
(VO) solve problem	achieve objective	solve
	bridge gap	alleviate
	improve quality	overcome
	deliver information	resolve
	encourage development	circumvent
(SVO) meeting take place	hold meeting	take
	event take place	get
	end season	win
	discussion take place	put
	do work	gain

Table 7: Nearest neighbor vectors for multiple words. POS-tags are not shown for simplicity.

category	predicate	arg1	arg2
adj_arg1	2.38	**6.55**	-
noun_arg1	3.37	**5.60**	-
verb_arg12	**6.78**	2.57	2.18

Table 8: L2-norms of the 50-dimensional weight vectors of the composition function Wadd_{nl}.

functions since the composition layers receive the same error signal via backpropagation.

We then trained the PAS-CLBLM using Wadd_{nl} to learn composition functions. Table 7 (*Composition*) shows the nearest neighbor words for each composed vector, and as we can see, the learned composition function emphasizes the head words and captures some sort of semantic similarity. We then inspected the L2-norms of the weight vectors of the composition function. As shown in Table 8, head words are strongly emphasized. Emphasizing head words is helpful in representing composed meanings, but in the case of verbs it may not always be sufficient. This can be observed in Table 3 and Table 4, which demonstrates that verb-related tasks are more difficult than noun-phrase tasks.

While *No composition* captures the semantic similarity well using independent parameters, there is the issue of data sparseness. As the size of the vocabulary increases, the number of tuples of word dependencies increases rapidly. In this experiment, we used only the 300,000 most frequent tuples. In contrast to this, the learned composition functions can capture similar information using only word vectors and a small set of predicate categories.

7 Conclusion and Future Work

We have presented a compositional log-bilinear language model using predicate-argument structures that incorporates both bag-of-words and dependency-based contexts. In our experiments the learned composed vectors achieve state-of-the-art scores for the task of measuring the semantic similarity between short phrases. For the subject-verb-object phrase task, the result is achieved without any pre-trained word vectors using a corpus an order of magnitude smaller than that of the previous state of the art. For future work, we will investigate how our models and the resulting vector representations can be helpful for other unsupervised and/or supervised tasks.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by JSPS KAKENHI Grant Number 13F03041.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora. *Language Resources and Evaluation*, 43(3):209–226.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Lan-

guage Model. *Journal of Machine Learning Research*, 3:1137–1155.

William Blacoe and Mirella Lapata. 2012. A Comparison of Vector-based Representations for Semantic Composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Katrin Erk and Sebastian Padó. 2008. A Structured Vector Space Model for Word Meaning in Context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 897–906.

Lev Finkelstein, Gabrilovich Evgenly, Matias Yossi, Rivlin Ehud, Solan Zach, Wolfman Gadi, and Ruppin Eytan. 2001. Placing Search in Context: The Concept Revisited. In *Proceedings of the Tenth International World Wide Web Conference*.

John Rupert Firth. 1957. A synopsis of linguistic theory 1930-55. In *Studies in Linguistic Analysis*, pages 1–32.

Kartik Goyal, Sujay Kumar Jauhar, Huiying Li, Mrinmaya Sachan, Shashank Srivastava, and Eduard Hovy. 2013. A Structured Distributional Semantic Model : Integrating Structure with Semantics. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 20–29.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental Support for a Categorical Compositional Distributional Model of Meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple Customization of Recursive Neural Networks for Semantic Relation Classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376.

Karl Moritz Hermann and Phil Blunsom. 2013. The Role of Syntax in Vector Space Models of Compositional Semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 894–904.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2013. Prior Disambiguation of Word Tensors for Constructing Sentence Vectors. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1590–1601.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2013. Separating Disambiguation from Composition in Distributional Semantics. In *Proceedings of 17th Conference on Natural Language Learning (CoNLL)*, pages 114–123.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at the International Conference on Learning Representations*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 236–244.

Jeff Mitchell and Mirella Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388–1439.

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

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems 26*, pages 2265–2273.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 1751–1758.

Denis Paperno, Nghia The Pham, and Marco Baroni. 2014. A practical and linguistically-motivated approach to compositional distributional semantics. In

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 90–99.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013a. Parsing with Compositional Vector Grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013b. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Richard Socher, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded Compositional Semantics for Finding and Describing Images with Sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing Semantic Representations Using Syntactically Enriched Vector Models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 948–957.

Masashi Tsubaki, Kevin Duh, Masashi Shimbo, and Yuji Matsumoto. 2013. Modeling and Learning Semantic Co-Compositionality through Prototype Projections and Neural Networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 130–140.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word Representations: A Simple and General Method for Semi-Supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394.

Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. 2013. A Tensor-based Factorization Model of Semantic Compositionality. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1142–1151.

Combining Distant and Partial Supervision for Relation Extraction

Gabor Angeli, Julie Tibshirani, Jean Y. Wu, Christopher D. Manning
Stanford University
Stanford, CA 94305
{angeli, jtibs, jeaneis, manning}@stanford.edu

Abstract

Broad-coverage relation extraction either requires expensive supervised training data, or suffers from drawbacks inherent to distant supervision. We present an approach for providing partial supervision to a distantly supervised relation extractor using a small number of carefully selected examples. We compare against established active learning criteria and propose a novel criterion to sample examples which are both uncertain and representative. In this way, we combine the benefits of fine-grained supervision for difficult examples with the coverage of a large distantly supervised corpus. Our approach gives a substantial increase of 3.9% end-to-end F_1 on the 2013 KBP Slot Filling evaluation, yielding a net F_1 of 37.7%.

1 Introduction

Fully supervised relation extractors are limited to relatively small training sets. While able to make use of much more data, *distantly supervised* approaches either make dubious assumptions in order to simulate fully supervised data, or make use of latent-variable methods which get stuck in local optima easily. We hope to combine the benefits of supervised and distantly supervised methods by annotating a small subset of the available data using selection criteria inspired by active learning.

To illustrate, our training corpus contains 1 208 524 relation mentions; annotating all of these mentions for a fully supervised classifier, at an average of \$0.13 per annotation, would cost approximately \$160 000. Distant supervision allows us to make use of this large corpus without requiring costly annotation. The traditional approach is based on the assumption that every mention of an entity pair (e.g., *Obama* and *USA*) participates in the known relation between the two (i.e., *born in*). However, this introduces noise, as not every mention expresses the relation we are assigning to it.

We show that by providing annotations for only 10 000 informative examples, combined with a large corpus of distantly labeled data, we can yield notable improvements in performance over the distantly supervised data alone. We report results on three criteria for selecting examples to annotate: a baseline of sampling examples uniformly at random, an established active learning criterion, and a new metric incorporating both the *uncertainty* and the *representativeness* of an example. We show that the choice of metric is important – yielding as much as a 3% F_1 difference – and that our new proposed criterion outperforms the standard method in many cases. Lastly, we train a supervised classifier on these collected examples, and report performance comparable to distantly supervised methods. Furthermore, we notice that initializing the distantly supervised model using this supervised classifier is critical for obtaining performance improvements.

This work makes a number of concrete contributions. We propose a novel application of active learning techniques to distantly supervised relation extraction. To the best of the authors knowledge, we are the first to apply active learning to the class of latent-variable distantly supervised models presented in this paper. We show that annotating a proportionally small number of examples yields improvements in end-to-end accuracy. We compare various selection criteria, and show that this decision has a notable impact on the gain in performance. In many ways this reconciles our results with the negative results of Zhang et al. (2012), who show limited gains from naïvely annotating examples. Lastly, we make our annotations available to the research community.[1]

[1] http://nlp.stanford.edu/software/mimlre.shtml

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1556–1567,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

2 Background

2.1 Relation Extraction

We are interested in extracting a set of relations $y_1 \dots y_k$ from a fixed set of possible relations $\mathcal{R}$, given two entities e_1 and e_2. For example, we would like to extract that Barack Obama was born in Hawaii. The task is decomposed into two steps: First, sentences containing mentions of both e_1 and e_2 are collected. The set of these sentences x, marked with the *entity mentions* for e_1 and e_2, becomes the input to the relation extractor, which then produces a set of relations which hold between the mentions. We are predominantly interested in the second step – classifying a set of pairs of entity mentions into the relations they express. Figure 1 gives the general setting for relation extraction, with entity pairs *Barack Obama* and *Hawaii*, and *Barack Obama* and *president*.

Traditionally, relation extraction has fallen into one of four broad approaches: supervised classification, as in the ACE task (Doddington et al., 2004; GuoDong et al., 2005; Surdeanu and Ciaramita, 2007), distant supervision (Craven and Kumlien, 1999; Wu and Weld, 2007; Mintz et al., 2009; Sun et al., 2011; Roth and Klakow, 2013) deterministic rule-based systems (Soderland, 1997; Grishman and Min, 2010; Chen et al., 2010), and translation from open domain information extraction schema (Riedel et al., 2013). We focus on the first two of these approaches.

2.2 Supervised Relation Extraction

Relation extraction can be naturally cast as a supervised classification problem. A corpus of relation mentions is collected, and each mention x is annotated with the relation y, if any, it expresses. The classifier's output is then aggregated to decide the relations between the two entities.

However, annotating supervised training data is generally expensive to perform at large scale. Although resources such as Freebase or the TAC KBP knowledge base have on the order of millions of training tuples over entities it is not feasible to manually annotate the corresponding mentions in the text. This has led to the rise of *distantly supervised* methods, which make use of this indirect supervision, but do not necessitate mention-level supervision.

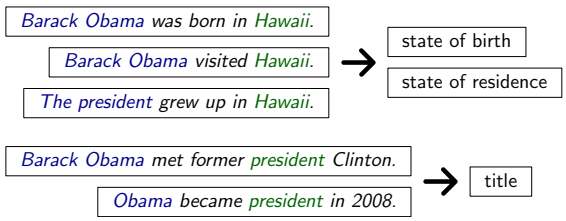


Figure 1: The relation extraction setup. For a pair of entities, we collect sentences which mention both entities. These sentences are then used to predict one or more relations between those entities. For instance, the sentences containing both *Barack Obama* and *Hawaii* should support the *state of birth* and *state of residence* relation.

2.3 Distant Supervision

Traditional distant supervision makes the assumption that for every triple (e_1, y, e_2) in a knowledge base, every sentence containing mentions for e_1 and e_2 express the relation y. For instance, taking Figure 1, we would create a datum for each of the three sentences containing BARACK OBAMA and HAWAII labeled with *state of birth*, and likewise with *state of residence*, creating 6 training examples overall. Similarly, both sentences involving *Barack Obama* and *president* would be marked as expressing the *title* relation.

While this allows us to leverage a large database effectively, it nonetheless makes a number of naïve assumptions. First – explicit in the formulation of the approach – it assumes that every mention expresses some relation, and furthermore expresses the known relation(s). For instance, the sentence *Obama visited Hawaii* would be erroneously treated as a positive example of the *born in* relation. Second, it implicitly assumes that our knowledge base is complete: entity mentions with no known relation are treated as negative examples.

The first of these assumptions is addressed by multi-instance multi-label (MIML) learning, described in Section 2.4. Min et al. (2013) address the second assumption by extending the MIML model with additional latent variables, while Xu et al. (2013) allow feedback from a coarse relation extractor to augment labels from the knowledge base. These latter two approaches are compatible with but are not implemented in this work.

2.4 Multi-Instance Multi-Label Learning

The multi-instance multi-label (MIML-RE) model of Surdeanu et al. (2012), which builds upon work

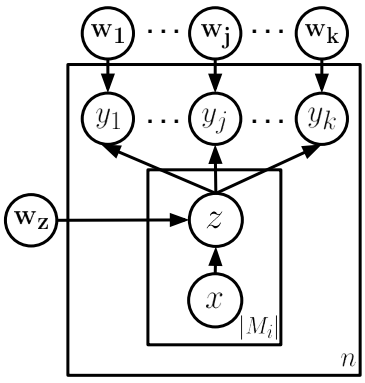


Figure 2: The MIML-RE model, as shown in Surdeanu et al. (2012). The outer plate corresponds to each of the n entity pairs in our knowledge base. Each entity pair has a set of mention pairs M_i, and a corresponding plate in the diagram for each mention pair in M_i. The variable x represents the input mention pair, whereas y represents the positive and negative relations for the given pair of entities. The latent variable z denotes a mention-level prediction for each input. The weight vector for the multinomial z classifier is given by $\mathbf{w}_z$, and there is a weight vector $\mathbf{w}_j$ for each binary y classifier.

by Hoffmann et al. (2011) and Riedel et al. (2010), addresses the assumptions of distantly supervised relations extractors in a principled way by positing a latent mention-level annotation.

The model groups mentions according to their entity pair – for instance, every mention pair with *Obama* and *Hawaii* would be grouped together. A latent variable z_i is created for every mention i, where $z_i \in \mathcal{R} \cup \{\text{None}\}$ takes a single relation label, or a *no relation* marker. We create $|\mathcal{R}|$ binary variables y representing the known positive and negative relations for the entity pair. A set of binary classifiers (log-linear factors in the graphical model) links the latent predictions $z_1 \ldots z_{|M_i|}$ and each y_j. These classifiers include two classes of features: first, a binary feature which fires if *at least one* of the mentions expresses a known relation between the entity pair, and second, a feature for each co-occurrence of relations for a given entity pair. Figure 2 describes the model.

2.5 Background on Active Learning

We describe preliminaries and prior work on active learning; we use this framework to propose two sampling schemes in Section 3 which we use to annotate mention-level labels for MIML-RE.

One way of expressing the generalization error of a hypothesis $\hat{h}$ is through its mean-squared error with the true hypothesis h:

$$\begin{aligned} &E[(h(x) - \hat{h}(x))^2] \\ &= E[E[(h(x) - \hat{h}(x))^2|x]] \\ &= \int_x E[(h(x) - \hat{h}(x))^2|x]p(x)\mathrm{d}x. \end{aligned}$$

The integrand can be further broken into bias and variance terms:

$$\begin{aligned} E[(h(x) - \hat{h}(x))^2] = (E[\hat{h}(x)] - h(x))^2 \\ + E[(\hat{h}(x) - E[\hat{h}(x)])^2] \end{aligned}$$

where for simplicity we've dropped the conditioning on x.

Many traditional sampling strategies, such as query-by-committee (QBC) (Freund et al., 1992; Freund et al., 1997) and uncertainty sampling (Lewis and Gale, 1994), work by decreasing the variance of the learned model. In QBC, we first create a 'committee' of classifiers by randomly sampling their parameters from a distribution based on the training data. These classifiers then make predictions on the unlabeled examples, and the examples on which there is the most disagreement are selected for labeling. This strategy can be seen as an attempt to decrease the *version space* – the set of classifiers that are consistent with the labeled data. Decreasing the version space should lower variance, since variance is inversely related to the size of the hypothesis space.

In most scenarios, active learning does not concern itself with the bias term. If a model is fundamentally misspecified, then no amount of additional training data can lower its bias. However, our paradigm differs from the traditional setting, in that we are annotating latent variables in a model with a non-convex objective. These annotations may help increase the convexity of our objective, leading us to a more accurate optimum and thereby lowering bias.

The other component to consider is $\int_x \cdots p(x)\mathrm{d}x$. This suggests that it is important to choose examples that are representative of the underlying distribution $p(x)$, as we want to label points that will improve the classifier's predictions on as many and as high-probability examples as possible. Incorporating a representativeness metric has been shown to provide a significant improvement over plain QBC or

uncertainty sampling (McCallum and Nigam, 1998; Settles, 2010).

2.6 Active Learning for Relation Extraction

Several papers have explored active learning for relation extraction. Fu and Grishman (2013) employ active learning to create a classifier quickly for new relations, simulated from the ACE corpus. Finn and Kushmerick (2003) compare a number of selection criteria – including QBC – for a supervised classifier. To the best of our knowledge, we are the first to apply active learning to distantly supervised relation extraction. Furthermore, we evaluate our selection criteria live in a real-world setting, collecting new sentences and evaluating on an end-to-end task.

For latent variable models, McCallum and Nigam (1998) apply active learning to semi-supervised document classification. We take inspiration from their use of QBC and the choice of metric for classifier disagreement. However their model assumes a fully Bayesian set-up, whereas ours does not require strong assumptions about the parameter distributions.

Settles et al. (2008) use active learning to improve a multiple-instance classifier. Their model is simpler in that it does not allow for unobserved variables or multiple labels, and the authors only evaluate on image retrieval and synthetic text classification datasets.

3 Example Selection

We describe three criteria for selection examples to annotate. The first – sampling uniformly – is a baseline for our hypothesis that intelligently selecting examples is important. For this criterion, we select mentions uniformly at random from the training set to annotate. This is the approach used in Zhang et al. (2012). The other two criteria rely on a metric for disagreement provided by QBC; we describe our adaptation of QBC for MIML-RE as a preliminary to introducing these criteria.

3.1 QBC For MIML-RE

We use a version of QBC based on bootstrapping (Saar-Tsechansky and Provost, 2004). To create the committee of classifiers, we re-sample the training set with replacement 7 times and train a model over each sampled dataset. We measure disagreement on z-labels among the classifiers using a generalized Jensen-Shannon divergence (McCallum and Nigam, 1998), taking the average KL divergence of all classifier judgments.

We first calculate the mention-level confidences. Note that $z_i^{(m)} \in M_i$ denotes the latent variable in entity pair i with index m; $\mathbf{z}_i^{(-m)}$ denotes the set of all latent variables except $z_i^{(m)}$:

$$p(z_i^{(m)}|\mathbf{y_i}, \mathbf{x_i}) = \frac{p(\mathbf{y_i}, z_i^{(m)}|\mathbf{x_i})}{p(\mathbf{y_i}|\mathbf{x_i})} = \frac{\sum_{\mathbf{z_i^{(-m)}}} p(\mathbf{y_i}, \mathbf{z_i}|\mathbf{x_i})}{\sum_{z_i^{(m)}} p(\mathbf{y_i}, z_i^{(m)}|\mathbf{x_i})}.$$

Notice that the denominator just serves to normalize the probability within a sentence group. We can rewrite the numerator as follows:

$$\begin{aligned}&\sum_{\mathbf{z_i^{(-m)}}} p(\mathbf{y_i}, \mathbf{z_i}|\mathbf{x_i}) \\ &= \sum_{\mathbf{z_i^{(-m)}}} p(\mathbf{y_i}|\mathbf{z_i}) p(\mathbf{z_i}|\mathbf{x_i}) \\ &= p(z_i^{(m)}|\mathbf{x_i}) \sum_{\mathbf{z_i^{(-m)}}} p(\mathbf{y_i}|\mathbf{z_i}) p(\mathbf{z_i^{(-m)}}|\mathbf{x_i}).\end{aligned}$$

For computational efficiency, we approximate $p(\mathbf{z_i^{(-m)}}|\mathbf{x_i})$ with a point mass at its maximum. Next, we calculate the Jensen-Shannon (JS) divergence from the k bootstrapped classifiers:

$$\frac{1}{k} \sum_{c=1}^{k} \mathrm{KL}(p_c(z_i^{(m)}|\mathbf{y_i}, \mathbf{x_i}) || p_{\text{mean}}(z_i^{(m)}|\mathbf{y_i}, \mathbf{x_i})) \quad (1)$$

where p_c is the probability assigned by each of the k classifiers to the latent $z_i^{(m)}$, and p_{mean} is the average of these probabilities. We use this metric to capture the *disagreement* of our model with respect to a particular latent variable. This is then used to inform our selection criteria.

We note that QBC may be especially useful in our situation as our objective is highly nonconvex. If two committee members disagree on a latent variable, it is likely because they converged to different local optima; annotating that example could help bring the classifiers into agreement.

The second selection criterion we consider is the most straightforward application of QBC – selecting the examples with the highest JS disagreement. This allows us to compare our criterion, described next, against an established criterion from the active learning literature.

3.2 Sample by JS Disagreement

We propose a novel active learning sampling criterion that incorporates not only disagreement but also *representativeness* in selecting examples to annotate. Prior work has taken a weighted combination of an example's disagreement and a score corresponding to whether the example is drawn from a dense portion of the feature space (e.g., McCallum and Nigam (1998)). However, this requires both selecting a criterion for defining density (e.g., distance metric in feature space), and tuning a parameter for the relative weight of disagreement versus representativeness.

Instead, we account for choosing representative examples by sampling without replacement proportional to the example's disagreement. Formally, we define the probability of selecting an example $z_i^{(m)}$ to be proportional to the Jensen-Shannon divergence in (1). Since the training set is an approximation to the prior distribution over examples, sampling uniformly over the training set is an approximation to sampling from the prior probability of seeing an input x. We can view our criterion as an approximation to sampling proportional to the product of two densities: a prior over examples x, and the JS divergence mentioned above.

4 Incorporating Sentence-Level Annotations

Following Surdeanu et al. (2012), MIML-RE is trained through hard discriminative Expectation Maximization, inferring the latent z values in the E-step and updating the weights for both the z and y classifiers in the M-step. During the E-step, we constrain the latent z to match our sentence-level annotations when available.

It is worth noting that even in the hard-EM regime, we can in principle incorporate annotator uncertainty elegantly into the model. At each E step, each z_i is set according to

$$z_i{}^{(m)*} \approx \arg\max_{z \in \mathcal{R}} \left[p(z \mid x_i^{(m)}, \mathbf{w}_z) \times \prod_r p(y_i^{(r)} \mid \mathbf{z}_i', \mathbf{w}_y^{(r)}) \right]$$

where $\mathbf{z}_i'$ contains the inferred labels from the previous iteration, but with its mth component replaced by $z_i^{(m)}$.

By setting the distribution $p(z \mid x_i^{(m)}, \mathbf{w}_z)$ to reflect uncertainty among annotators, we can leave open the possibility for the model to choose a relation which annotators deemed unlikely, but the model nonetheless prefers. For simplicity, however, we treat our annotations as a hard assignment.

In addition to incorporating annotations during training, we can also use this data to intelligently initialize the model. Since the MIML-RE objective is non-convex, the initialization of the classifier weights w_y and w_z is important. The y classifiers are initialized with the "at-least-once" assumption of Hoffmann et al. (2011); w_z can be initialized either using traditional distant supervision or from a supervised classifier trained on the annotated sentences. If initialized with a supervised classifier, the model can be viewed as augmenting this supervised model with a large distantly labeled corpus, providing both additional entity pairs to train from, and additional mentions for an annotated entity pair.

5 Crowdsourced Example Annotation

Most prior work on active learning is done by simulation on a fully labeled dataset; such a dataset doesn't exist for our case. Furthermore, a key aim of this paper is to practically improve state-of-the-art performance in relation extraction in addition to evaluating active learning criteria. Therefore, we develop and execute an annotation task for collecting labels for our selected examples.

We utilize Amazon Mechanical Turk to crowdsource annotations. For each task, the annotator (Turker) is presented with the task description, followed by 15 questions, 2 of which are randomly placed controls. For each question, we present Turkers with a relation mention and the top 5 relation predictions from our classifier. The Turker also has an option to freely specify a relation not presented in the first five options, or mark that there is no relation. We attempt to heuristically match common free-form answers to official relations.

To maintain the quality of the results, we discard all submissions in which both controls were answered incorrectly, and additionally discard all submissions from Turkers who failed the controls on more than $\frac{1}{3}$ of their submissions. Rejected tasks were republished for other workers to complete. We collect 5 annotations for each example, and use the most commonly agreed answer as the ground truth. Ties are broken arbitrarily, except in

Figure 3: The task shown to Amazon Mechanical Turk workers. A sentence along with the top 5 relation predictions from our classifier are shown to Turkers, as well as an option to specify a custom relation or manually enter "no relation." The correct response for this example should be either no relation or a custom relation.

the case of deciding between a relation and no relation, in which case the relation was always chosen.

A total of 23 725 examples were annotated, covering 10 000 examples for each of the three selection criteria. Note that there is overlap between the examples selected for the three criteria. In addition, 10 023 examples were annotated during development; these are included in the set of all annotated examples, but excluded from any of the three criteria. The compensation per task was 23 cents; the total cost of annotating examples was $3156, in addition to $204 spent on developing the task. Informally, Turkers achieved an accuracy of around 75%, as evaluated by a paper author, performing disproportionately well on identifying the *no relation* label.

6 Experiments

We evaluate the three high-level research contributions of this work: we show that we improve the accuracy of MIML-RE, we validate the effectiveness of our selection criteria, and we provide a corpus of annotated examples, evaluating a supervised classifier trained on this corpus. The training and testing methodology for evaluating these contributions is given in Sections 6.1 and 6.2; experiments are given in Section 6.3.

6.1 Training Setup

We adopt the setup of Surdeanu et al. (2012) for training the MIML-RE model, with minor modifications. We use both the 2010 and 2013 KBP official document collections, as well as a July 2013 dump of Wikipedia as our text corpus. We subsample negatives such that $\frac{1}{3}$ of our dataset consists of entity pairs with no known relations. In all experiments, MIML-RE is trained for 7 iterations of EM; for efficiency, the z classifier is optimized using stochastic gradient descent;[2] the y classifiers are optimized using L-BFGS.

Similarly to Surdeanu et al. (2011), we assign negative relations which are either incompatible with the known positive relations (e.g., relations whose co-occurrence would violate type constraints); or, are actually functional relations in which another entity already participates. For example, if we know that Obama was born in the United States, we could add *born in* as a negative relation to the pair Obama and Kenya.

Our dataset consists of 325 891 entity pairs with at least one positive relation, and 158 091 entity pairs with no positive relations. Pairs with at least one known relation have an average of 4.56 mentions per group; groups with no known relations have an average of 1.55 mentions per group. In total, 1 208 524 distinct mentions are considered; the annotated examples are selected from this pool.

6.2 Testing Methodology

We compare against the original MIML-RE model using the same dataset and evaluation methodology as Surdeanu et al. (2012). This allows for an evaluation where the only free variable between this and prior work is the predictions of the relation extractor.

Additionally, we evaluate the relation extractors in the context of Stanford's end-to-end KBP system (Angeli et al., 2014) using the NIST TAC-KBP 2013 English Slotfilling evaluation. In the end-to-end framework, the input to the system is a query entity and a set of articles, and the output is a set of *slot fills* – each slot fill is a candidate triple in the knowledge base, the first element of which is the query entity. This amounts to roughly populating a data structure like Wikipedia infoboxes automatically from a large corpus of text.

Importantly, an end-to-end evaluation in a top-performing full system gives a more accurate idea of the expected real-world gain from each model. Both the information retrieval component providing candidates to the relation extractor, as well as

[2]For the sake of consistency, the supervised classifiers and those in Mintz++ are trained identically to the z classifiers in MIML-RE.

Method	Init	Active Learning Criterion														
		Not Used			Uniform			High JS			Sample JS			All Available		
		P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Mintz++	—	41.3	28.2	33.5		—			—			—			—	
MIML-RE	Dist	38.0	30.5	33.8	39.2	30.4	34.2	41.7	28.9	34.1	36.6	31.1	33.6	37.5	30.6	33.7
	Sup	35.1	35.6	35.4	34.4	35.0	34.7	**46.2**	30.8	37.0	39.4	36.2	**37.7**	36.0	**37.1**	36.5
Supervised	—		—		**35.5**	28.9	31.9	31.3	33.2	32.2	33.5	**35.0**	**34.2**	32.9	33.4	33.2

Table 1: A summary of results on the end-to-end KBP 2013 evaluation for various experiments. The first column denotes the algorithm used: either traditional distant supervision (Mintz++), MIML-RE, or a supervised classifier. In the case of MIML-RE, the model may be initialized either using Mintz++, or the corresponding supervised classifier (the "Not Used" column is initialized with the "All" supervised classifier). One of five active learning scenarios are evaluated: no annotated examples provided, the three active learning criteria, and all available examples used. The entry in blue denotes the basic MIML-RE model; entries in gray perform worse than this model. The bold items denote the best performance among selection criteria.

the consistency and inference performed on the classifier output introduce bias in this evaluation's sensitivity to particular types of errors. Mistakes which are easy to filter, or are difficult to retrieve using IR are less important in this evaluation; in contrast, factors such as providing good confidence scores for consistency become more important.

For the end-to-end evaluation, we use the official evaluation script with two changes: First, all systems are evaluated with provenance ignored, so as not to penalize any system for finding a new provenance not validated in the official evaluation key. Second, each system reports its optimal F_1 along its P/R curve, yielding results which are optimistic when compared against other systems entered into the competition. However, this also yields results which are invariant to threshold tuning, and is therefore more appropriate for comparing between systems in this paper.

Development was done on the KBP 2010–2012 queries, and results are reported using the 2013 queries as a simulated test set. Our best system achieves an F_1 of 37.7; the top two teams at KBP 2013 (of 18 entered) achieved F_1 scores of 40.2 and 37.1 respectively, ignoring provenance.

6.3 Results

Table 1 summarizes all results for the end-to-end task; relevant features of the table are copied in subsequent sections to illustrate key trends. Models which perform worse than the original MIML-RE model (MIML-RE, initialized with "Dist," under "Not Used") are denoted in gray. The best performing model improves on the base model by 3.9 F_1 points on the end-to-end task.

System	P	R	F_1
Mintz++	41.3	28.2	33.5
MIML + Dist	38.0	30.5	33.8
MIML + Sup	35.1	35.6	35.4
MIML + Dist + SampleJS	36.6	31.1	33.6
MIML + Sup + SampleJS	**39.4**	**36.2**	**37.7**

Table 2: A summary of improvements to MIML-RE on the end-to-end slotfilling task, copied from Table 1. Mintz++ is the traditional distantly supervised model. The second row corresponds to the unmodified MIML-RE model. The third row corresponds to MIML-RE initialized with a supervised classifier (trained on all examples). The fourth row is MIML-RE with annotated examples incorporated during training (but not initialization). The last row shows the best results obtained by our model.

We evaluate each of the individual contributions of the paper: improving the accuracy of the MIML-RE relation extractor, evaluating our example selection criteria, and demonstrating the annotated examples' effectiveness for a fully-supervised relation extractor.

Improve MIML-RE Accuracy A key goal of this work is to improve the accuracy of the MIML-RE model; we show that we improve the model both on the end-to-end slotfilling task (Table 2) as well as on a standard evaluation (Figure 5). Similar to our work, recent work by Pershina et al.

System	P	R	F_1
MIML + Sup	35.1	35.6	35.4
MIML + Sup + Uniform	34.4	35.0	34.7
MIML + Sup + HighJS	**46.2**	30.8	37.0
MIML + Sup + SampleJS	39.4	36.2	**37.7**
MIML + Sup + All	36.0	**37.1**	36.5

Table 3: A summary of the performance of each example selection criterion. In each case, the model was initialized with a supervised classifier. The first row corresponds to the MIML-RE model initialized with a supervised classifier. The middle three rows show performance for the three selection criteria, used both for initialization and during training. The last row shows results if all available annotations are used, independent of their source.

System	P	R	F_1
Mintz++	**41.3**	28.2	33.5
MIML + Dist	38.0	30.5	33.8
Supervised + SampleJS	33.5	**35.0**	**34.2**
MIML + Sup	35.1	35.6	35.5
MIML + Sup + SampleJS	39.4	36.2	37.7

Table 4: A comparison of the best performing supervised classifier with other systems. The top section compares the supervised classifier with prior work. The lower section highlights the improvements gained from initializing MIML-RE with a supervised classifier.

(2014) incorporates labeled data to guide MIML-RE during training. They make use of labeled data to extract *training guidelines*, which are intended to generalize across many examples. We show that we can match or outperform their improvements with our best criterion.

A few interesting trends emerge from the end-to-end results in Table 2. Using annotated sentences during training alone did not improve performance consistently, even hurting performance when the SampleJS criterion was used. This supports an intuition that the initialization of the model is important, and that it is relatively difficult to coax the model out of a local optimum if it is initialized poorly. This is further supported by the improvement in performance when the model is initialized with a supervised classifier, even when no examples are used during training. Similar trends are reported in prior work, e.g., Smith and Eisner (2007) Section 4.4.6.

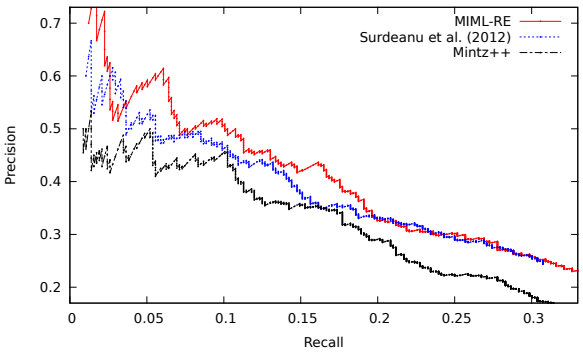


Figure 4: MIML-RE and Mintz++ evaluated according to Surdeanu et al. (2012). The original model from the paper is plotted for comparison, as our training methodology is somewhat different.

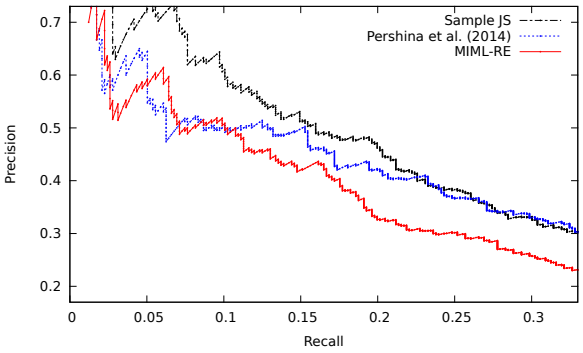


Figure 5: Our best active learning criterion evaluated against our version of MIML-RE, alongside the best system of Pershina et al. (2014).

Also interesting is the relatively small gain MIML-RE provides over traditional distant supervision (Mintz++) in this setting. We conjecture that the mistakes made by Mintz++ are often relatively easily filtered by the downstream consistency component. This is supported by Figure 4; we evaluate our trained MIML-RE model against Mintz++ and the results reported in Surdeanu et al. (2012). We show that our model performs as well or better than the original implementation, and consistently outperforms Mintz++.

Evaluate Selection Criteria A key objective of this work is to evaluate how much of an impact careful selection of annotated examples has on the overall performance of the system. We evaluate the three selection criteria from Section 3.2, showing the results for MIML-RE in Table 3; results for the supervised classifier are given in Table 1. In both cases, we show that the sampled JS cri-

terion performs comparably to or better than the other criteria.

At least two interesting trends can be noted from these results: First, the uniformly sampled criterion performed worse than MIML-RE initialized with a supervised classifier. This may be due to noise in the annotation: a small number of annotation errors on entity pairs with only a single corresponding mention could introduce dangerous noise into training. These singleton mentions will rarely have disagreement between the committee of classifiers, and therefore will generally only be selected in the uniform criterion.

Second, adding in the full set of examples did not improve performance – in fact, performance generally dropped in this scenario. We conjecture that this is due to the inclusion of the uniformly sampled examples, with performance dropping for the same reasons as above.

Both of these results can be reconciled with the results of Zhang et al. (2012); like this work, they annotated examples to analyze the trade-off between adding more data to a distantly supervised system, and adding more direct supervision. They conclude that annotations provide only a relatively small improvement in performance. However, their examples were uniformly selected from the training corpus, and did not make use of the structure provided by MIML-RE. Our results agree in that neither the uniform selection criterion nor the supervised classifier significantly outperformed the unmodified MIML-RE model; nonetheless, we show that if care is taken in selecting these labeled examples we can achieve noticeable improvements in accuracy.

We also evaluate our selection criteria on the evaluation of Surdeanu et al. (2012), both initialized with Mintz++ (Figure 7) and with the supervised classifier (Figure 6). These results mirror those in the end-to-end evaluation; when initialized with the supervised classifier the high disagreement (High JS) and sampling proportional to disagreement (Sample JS) criteria clearly outperform both the base MIML-RE model as well as the uniformly sampling criterion. Using the annotated examples only during training yielded no perceivable benefit over the base model (Figure 7).

Supervised Relation Extractor The examples collected can be used to directly train a supervised classifier, with results summarized in Table 4. The most salient insight is that the performance of the best supervised classifier is similar to that of the MIML-RE model, despite being trained on nearly two orders of magnitude less training data.

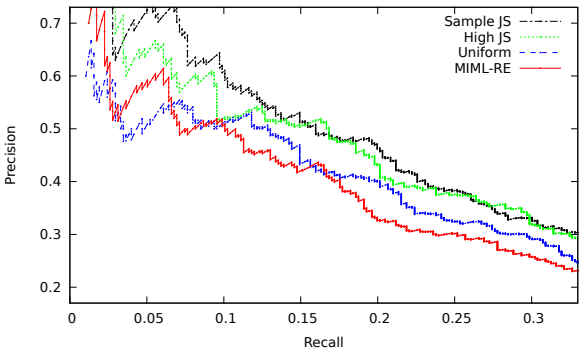


Figure 6: A comparison of models trained with various selection criteria on the evaluation of Surdeanu et al. (2012), all initialized with the corresponding supervised classifier.

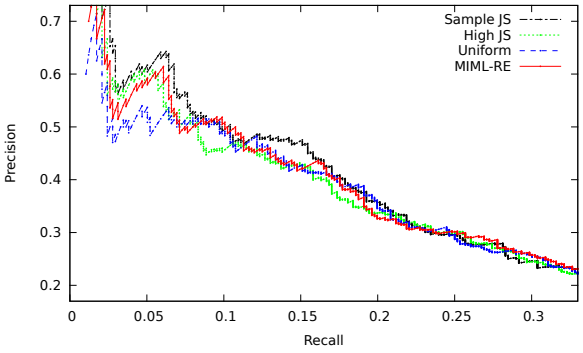


Figure 7: A comparison of models trained with various selection criteria on the evaluation of Surdeanu et al. (2012), all initialized with Mintz++.

More interestingly, however, the supervised classifier provides a noticeably better initialization for MIML-RE than Mintz++, yielding better results even without enforcing the labels during EM. These results suggest that the power gained from the the more sophisticated MIML-RE model is best used in conjunction with a small amount of training data. That is, using MIML-RE as a principled model for combining a large distantly labeled corpus and a small number of careful annotations yields significant improvement over using either of the two data sources alone.

Relation	#	P		R		F_1	
no_relation	3073						
employee_of	1978	29	**32**	33	**46**	31	**38**
countries_of_res.	1061	30	**42**	7	**40**	11	**41**
states_of_residence	427	57	**33**	14	**7**	23	**12**
cities_of_residence	356	31	**52**	9	**30**	14	**38**
(org:)member_of	290	0	0	0	0	0	0
country_of_hq	280	63	**62**	65	**62**	64	**62**
top_members	221	36	**26**	50	**60**	42	**36**
country_of_birth	205	22	**0**	40	**0**	29	**0**
parents	196	10	**26**	31	**54**	15	**35**
city_of_hq	194	46	**52**	57	**61**	51	**56**
(org:)alt_names	184	52	**48**	39	39	45	**43**
founded_by	180	100	**89**	29	**38**	44	**53**
city_of_birth	145	17	**50**	8	**17**	11	**25**
state_of_hq	132	50	**64**	30	**35**	38	**45**
title	121	20	**26**	28	**35**	23	**30**
subsidiaries	105	33	**25**	6	**3**	10	**5**
founded	90	62	**82**	62	**69**	62	**75**
spouse	88	37	**54**	85	85	51	**66**
origin	86	42	**43**	68	**70**	51	**53**
state_of_birth	83	0	**50**	0	**10**	0	**17**
charges	69	54	54	16	16	24	24
cause_of_death	69	93	93	39	39	55	55
(per:)alt_names	69	9	**20**	2	**3**	3	**6**
country_of_death	65	100	100	10	10	18	18
members	54	0	0	0	0	0	0
children	52	53	**62**	14	**18**	22	**27**
parents	50	64	64	28	28	39	39
city_of_death	38	42	**75**	16	**19**	23	**30**
dissolved	38	0	0	0	0	0	0
date_of_death	33	64	64	44	**39**	52	**48**
political_affiliation	23	7	**25**	100	100	13	**40**
state_of_death	19	0	0	0	0	0	0
shareholders	19	0	0	0	0	0	0
siblings	16	50	50	33	33	40	40
schools_attended	14	80	**78**	41	**48**	54	**60**
date_of_birth	11	100	100	85	85	92	92
other_family	9	0	0	0	0	0	0
age	4	94	**97**	94	**90**	94	**93**
#_of_employees	3	0	0	0	0	0	0
religion	2	100	100	29	29	44	44
website	0	25	**0**	3	**0**	6	**0**

Table 5: A summary of relations annotated, and end-to-end slotfilling performance by relation. The first column gives the relation; the second shows the number of examples annotated. The subsequent columns show the performance of the unmodified MIML-RE model and our best performing model (SampleJS). Changes in values are bolded; positive changes are shown in green and negative changes in red. The most frequent 10 relations in the evaluation are likewise bolded.

6.4 Analysis By Relation

In this section, we explore which of the KBP relations were shown to Turkers, and whether the improvements in accuracy correspond to these relations. We compare only the unmodified MIML-RE model, and our best model (MIML-RE initialized with the supervised classifier, under the SampleJS criterion). Results are shown in Table 5.

A few interesting trends emerge from this analysis. We note that annotating even 80+ examples for a relation seems to provide a consistent boost in accuracy, whereas relations with fewer annotated examples tended to show little or no change. However, the gains of our model are not universal across relation types, even dropping noticeably on some – for instance, F_1 drops on both *state of residence* and *country of birth*. This could suggest systematic noise from Turker judgments; e.g., for foreign geography (*state of residence*) or ambiguous relations (*top members*).

An additional insight from the table is the mismatch between examples chosen to be annotated, and the most popular relations in the KBP evaluation. For instance, by far the most popular KBP relation (*title*) had only 121 examples annotated.

7 Conclusion

We have shown that providing a relatively small number of mention-level annotations can improve the accuracy of MIML-RE, yielding an end-to-end improvement of 3.9 F_1 on the KBP task. Furthermore, we have introduced a new active learning criterion, and shown both that the choice of criterion is important, and that our new criterion performs well. Lastly, we make available a dataset of mention-level annotations for constructing a traditional supervised relation extractor.

Acknowledgements

We thank the anonymous reviewers for their thoughtful comments, and Dan Weld for his feedback on additional experiments and analysis. We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Gabor Angeli, Arun Chaganty, Angel Chang, Kevin Reschke, Julie Tibshirani, Jean Y. Wu, Osbert Bastani, Keith Siilats, and Christopher D. Manning. 2014. Stanford's 2013 KBP system. In *TAC-KBP*.

Zheng Chen, Suzanne Tamang, Adam Lee, Xiang Li, Wen-Pin Lin, Matthew Snover, Javier Artiles, Marissa Passantino, and Heng Ji. 2010. CUNY-BLENDER. In *TAC-KBP*.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *AAAI*.

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

Aidan Finn and Nicolas Kushmerick. 2003. Active learning selection strategies for information extraction. In *International Workshop on Adaptive Text Extraction and Mining*.

Yoav Freund, H Sebastian Seung, Eli Shamir, and Naftali Tishby. 1992. Information, prediction, and query by committee. In *NIPS*.

Yoav Freund, H Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2-3):133–168.

Lisheng Fu and Ralph Grishman. 2013. An efficient active learning framework for new relation types. In *IJCNLP*.

Ralph Grishman and Bonan Min. 2010. New York University KBP 2010 slot-filling system. In *Proc. TAC 2010 Workshop*.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *ACL*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL-HLT*.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR*.

Andrew McCallum and Kamal Nigam. 1998. Employing EM and pool-based active learning for text classification. In *ICML*.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *NAACL-HLT*.

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

Maria Pershina, Bonan Min, Wei Xu, and Ralph Grishman. 2014. Infusion of labeled data into distant supervision for relation extraction. In *ACL*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*. Springer.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL-HLT*.

Benjamin Roth and Dietrich Klakow. 2013. Feature-based models for improving the quality of noisy training data for relation extraction. In *CIKM*.

Maytal Saar-Tsechansky and Foster Provost. 2004. Active sampling for class probability estimation and ranking. *Machine Learning*, 54(2):153–178.

Burr Settles, Mark Craven, and Soumya Ray. 2008. Multiple-instance active learning. In *Advances in neural information processing systems*, pages 1289–1296.

Burr Settles. 2010. Active learning literature survey. *University of Wisconsin Madison Technical Report 1648*.

Noah Smith and Jason Eisner. 2007. *Novel estimation methods for unsupervised discovery of latent structure in natural language text*. Ph.D. thesis, Johns Hopkins.

Stephen G Soderland. 1997. *Learning text analysis rules for domain-specific natural language processing*. Ph.D. thesis, University of Massachusetts.

Ang Sun, Ralph Grishman, Wei Xu, and Bonan Min. 2011. New York University 2011 system for KBP slot filling. In *Proceedings of the Text Analytics Conference*.

Mihai Surdeanu and Massimiliano Ciaramita. 2007. Robust information extraction with perceptrons. In *ACE07 Proceedings*.

Mihai Surdeanu, Sonal Gupta, John Bauer, David McClosky, Angel X Chang, Valentin I Spitkovsky, and Christopher D Manning. 2011. Stanfords distantly-supervised slot-filling system. In *Proceedings of the Text Analytics Conference*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP*.

Fei Wu and Daniel S Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on information and knowledge management*. ACM.

Wei Xu, Le Zhao, Raphael Hoffman, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *ACL*.

Ce Zhang, Feng Niu, Christopher Ré, and Jude Shavlik. 2012. Big data versus the crowd: Looking for relationships in all the right places. In *ACL*.

Typed Tensor Decomposition of Knowledge Bases for Relation Extraction

Kai-Wei Chang[†*] **Wen-tau Yih**[♮] **Bishan Yang**[‡*] **Christopher Meek**[♮]

[†]University of Illinois, Urbana, IL 61801, USA
[‡]Cornell University, Ithaca, NY 14850, USA
[♮]Microsoft Research, Redmond, WA 98052, USA

Abstract

While relation extraction has traditionally been viewed as a task relying solely on textual data, recent work has shown that by taking as input existing facts in the form of entity-relation triples from both knowledge bases and textual data, the performance of relation extraction can be improved significantly. Following this new paradigm, we propose a tensor decomposition approach for knowledge base embedding that is highly scalable, and is especially suitable for relation extraction. By leveraging relational domain knowledge about entity type information, our learning algorithm is significantly faster than previous approaches and is better able to discover new relations missing from the database. In addition, when applied to a relation extraction task, our approach alone is comparable to several existing systems, and improves the weighted mean average precision of a state-of-the-art method by 10 points when used as a subcomponent.

1 Introduction

Identifying the relationship between entities from free text, *relation extraction* is a key task for acquiring new facts to increase the coverage of a structured knowledge base. Given a pre-defined database schema, traditional relation extraction approaches focus on learning a classifier using textual data alone, such as patterns between the occurrences of two entities in documents, to determine whether the entities have a particular relation. Other than using the existing known facts to label the text corpora in a distant supervision setting (Bunescu and Mooney, 2007; Mintz et al., 2009; Riedel et al., 2010; Ritter et al., 2013), an existing knowledge base is typically not involved in the process of relation extraction.

However, this paradigm has started to shift recently, as researchers showed that by taking existing facts of a knowledge base as an integral part of relation extraction, the model can leverage richer information and thus yields better performance. For instance, Riedel et al. (2013) borrowed the idea of *collective filtering* and constructed a matrix where each row is a pair of entities and each column is a particular relation. For a true entity-relation triple (e_1, r, e_2), either from the text corpus or from the knowledge base, the corresponding entry in the matrix is 1. A previously unknown fact (i.e., triple) can be discovered through matrix decomposition. This approach can be viewed as creating vector representations of each relation and candidate pair of entities. Because each entity does not have its own representation, relationships of any unpaired entities cannot be discovered. Alternatively, Weston et al. (2013) created two types of embedding – one based on textual similarity and the other based on knowledge base, where the latter maps each entity and relation to the same d-dimensional vector space using a model proposed by Bordes et al. (2013a). They also showed that combining these two models results in a significant improvement over the model trained using only textual data.

To make such an integrated strategy work, it is important to capture all existing entities and relations, as well as the known facts, from both textual data and large databases. In this paper, we propose a new knowledge base embedding model, TRESCAL, that is highly efficient and scalable, with relation extraction as our target application. Our work is built on top of RESCAL (Nickel et al., 2011), which is a tensor decomposition method that has proven its scalability by factoring YAGO (Biega et al., 2013) with 3 million entities

[*]Work conducted while interning at Microsoft Research.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1568–1579,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

and 41 million triples (Nickel et al., 2012). We improve the tensor decomposition model with two technical innovations. First, we exclude the triples that do not satisfy the relational constraints (e.g., both arguments of the relation *spouse-of* need to be *person* entities) from the loss, which is done by selecting sub-matrices of each slice of the tensor during training. Second, we introduce a mathematical technique that significantly reduces the computational complexity in both time and space when the loss function contains a regularization term. As a consequence, our method is more than four times faster than RESCAL, and is also more accurate in discovering unseen triples.

Our contributions are twofold. First, compared to other knowledge base embedding methods developed more recently, it is much more efficient to train our model. As will be seen in Sec. 5, when applied to a large knowledge base created using NELL (Carlson et al., 2010) that has 1.8M entity-relation triples, our method finishes training in 4 to 5 hours, while an alternative method (Bordes et al., 2013a) needs almost 3 days. Moreover, the prediction accuracy of our model is competitive to others, if not higher. Second, to validate its value to relation extraction, we apply TRESCAL to extracting relations from a free text corpus along with a knowledge base, using the data provided in (Riedel et al., 2013). We show that TRESCAL is complementary to existing systems and significantly improves their performance when using it as a subcomponent. For instance, this strategy improves the weighted mean average precision of the best approach in (Riedel et al., 2013) by 10 points (47% to 57%).

The remainder of this paper is organized as follows. We survey most related work in Sec. 2 and provide the technical background of our approach in Sec. 3. Our approach is detailed in Sec. 4, followed by the experimental validation in Sec. 5. Finally, Sec. 6 concludes the paper.

2 Related Work

Our approach of creating knowledge base embedding is based on tensor decomposition, which is a well-developed mathematical tool for data analysis. Existing tensor decomposition models can be categorized into two main families: the CP and Tucker decompositions. The CP (CANDECOMP/PARAFAC) decomposition (Kruskal, 1977; Kiers, 2000) approximates a tensor by a sum of rank-one tensors, while the Tucker decomposition (Tucker, 1966), also known as high-order SVD (De Lathauwer et al., 2000), factorizes a tensor into a core tensor multiplied by a matrix along each dimension. A highly scalable distributional algorithm using the Map-Reduce architecture has been proposed recently for computing CP (Kang et al., 2012), but not for the Tucker decomposition, probably due to its inherently more complicated model form.

Matrix and tensor decomposition methods have been applied to modeling multi-relational data. For instance, Speer et al. (2008) aimed to create vectors of latent components for representing concepts in a common sense knowledge base using SVD. Franz et al. (2009) proposed TripleRank to model the `subject-predicate-object` RDF triples in a tensor, and then applied the CP decomposition to identify hidden triples. Following the same tensor encoding, Nickel et al. (2011) proposed RESCAL, a restricted form of Tucker decomposition for discovering previously unknown triples in a knowledge base, and later demonstrated its scalability by applying it to YAGO, which was encoded in a 3M $\times$ 3M $\times$ 38 tensor with 41M triples (Nickel et al., 2012).

Methods that revise the objective function based on additional domain information have been proposed, such as MrWTD, a multi-relational weighted tensor decomposition method (London et al., 2013), coupled matrix and tensor factorization (Papalexakis et al., 2014), and collective matrix factorization (Singh and Gordon, 2008). Alternatively, instead of optimizing for the least-squares reconduction loss, a non-parametric Bayesian approach for 3-way tensor decomposition for modeling relational data has also been proposed (Sutskever et al., 2009). Despite the existence of a wide variety of tensor decomposition models, most methods do not scale well and have only been tested on datasets that are much smaller than the size of real-world knowledge bases.

Multi-relational data can be modeled by neural-network methods as well. For instance, Bordes et al. (2013b) proposed the Semantic Matching Energy model (SME), which aims to have the same d-dimensional vector representations for both entities and relations. Given the vectors of entities e_1, e_2 and relation r. They first learn the latent representations of (e_1, r) and (e_2, r). The score of (e_1, r, e_2) is determined by the inner product

of the vectors of (e_1, r) and (e_2, r). Later, they proposed a more scalable method called translating embeddings (TransE) (Bordes et al., 2013a). While both entities and relations are still represented by vectors, the score of (e_1, r, e_2) becomes the negative dissimilarity measure of the corresponding vectors $-\|\mathbf{e_i} + \mathbf{r_k} - \mathbf{e_j}\|$, motivated by the work in (Mikolov et al., 2013b; Mikolov et al., 2013a). Alternatively, Socher et al. (2013) proposed a Neural Tensor Network (NTN) that represents entities in d-dimensional vectors created separately by averaging pre-trained word vectors, and then learns a $d \times d \times m$ tensor describing the interactions between these latent components in each of the m relations. All these methods optimize for loss functions that are more directly related to the true objective – the prediction accuracy of correct entity-relation triples, compared to the mean-squared reconstruction error in our method. Nevertheless, they typically require much longer training time.

3 Background

In this section, we first describe how entity-relation triples are encoded in a tensor. We then introduce the recently proposed tensor decomposition method, RESCAL (Nickel et al., 2011) and explain how it adopts an alternating least-squares method, ASALSAN (Bader et al., 2007), to compute the factorization.

3.1 Encoding Binary Relations in a Tensor

Suppose we are given a knowledge base with n entities and m relation types, and the facts in the knowledge base are denoted as a set of entity-relation triples $\mathcal{T} = \{(e_i, r_k, e_j)\}$, where $i, j \in \{1, 2, \cdots n\}$ and $k \in \{1, 2, \cdots m\}$. A triple (e_i, r_k, e_j) simply means that the i-th entity and the j-th entity have the k-th relation. Following (Franz et al., 2009), these triples can naturally be encoded in a 3-way tensor $\mathcal{X} \in \{0, 1\}^{n \times n \times m}$, such that $\mathcal{X}_{i,j,k} = 1$ if and only if the triple $(e_i, r_k, e_j) \in \mathcal{T}$[1]. The tensor can be viewed as consisting of m slices, where each slice is an $n \times n$ square matrix, denoting the interactions of the entities of a particular relation type. In the remainder of this paper, we will use $\mathcal{X}_k$ to refer to the k-th slice of the tensor $\mathcal{X}$. Fig. 1 illustrates this representation.

[1]This representation can easily be extended for a probabilistic knowledge base by allowing nonnegative real values.

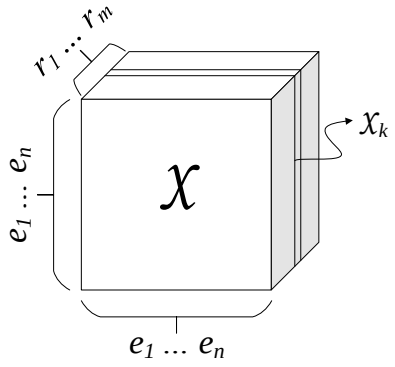


Figure 1: A tensor encoding of m binary relation types and n entities. A slice $\mathcal{X}_k$ denotes the entities having the k-th relation.

3.2 RESCAL

In order to identify latent components in a tensor for collective learning, Nickel et al. (2011) proposed RESCAL, which is a tensor decomposition approach specifically designed for the multi-relational data described in Sec. 3.1. Given a tensor $\mathcal{X}_{n \times n \times m}$, RESCAL aims to have a rank-r approximation, where each slice $\mathcal{X}_k$ is factorized as

$$\mathcal{X}_k \approx \mathbf{A}\mathcal{R}_k\mathbf{A}^T. \tag{1}$$

$\mathbf{A}$ is an $n \times r$ matrix, where the i-th row denotes the r latent components of the i-th entity. $\mathcal{R}_k$ is an asymmetric $r \times r$ matrix that describes the interactions of the latent components according to the k-th relation. Notice that while $\mathcal{R}_k$ differs in each slice, $\mathbf{A}$ remains the same.

$\mathbf{A}$ and $\mathcal{R}_k$ are derived by minimizing the loss function below.

$$\min_{\mathbf{A}, \mathcal{R}_k} f(\mathbf{A}, \mathcal{R}_k) + \lambda \cdot g(\mathbf{A}, \mathcal{R}_k), \tag{2}$$

where $f(\mathbf{A}, \mathcal{R}_k) = \frac{1}{2}\left(\sum_k \|\mathcal{X}_k - \mathbf{A}\mathcal{R}_k\mathbf{A}^T\|_F^2\right)$ is the mean-squared reconstruction error and $g(\mathbf{A}, \mathcal{R}_k) = \frac{1}{2}\left(\|\mathbf{A}\|_F^2 + \sum_k \|\mathcal{R}_k\|_F^2\right)$ is the regularization term.

RESCAL is a special form of Tucker decomposition (Tucker, 1966) operating on a 3-way tensor. Its model form (Eq. (1)) can also be regarded as a relaxed form of DEDICOM (Bader et al., 2007), which derives the low-rank approximation as: $\mathcal{X}_k \approx \mathbf{A}\mathcal{D}_k\mathbf{R}\mathcal{D}_k\mathbf{A}^T$. To compare RESCAL to other tensor decomposition methods, interested readers can refer to (Kolda and Bader, 2009).

The optimization problem in Eq. (2) can be solved using the efficient alternating least-squares (ALS) method. This approach alternatively fixes $\mathcal{R}_k$ to solve for $\mathbf{A}$ and then fixes $\mathbf{A}$ to solve $\mathcal{R}_k$. The whole procedure stops until $\frac{f(\mathbf{A},\mathcal{R}_k)}{\|\mathcal{X}\|_F^2}$ converges to some small threshold ϵ or the maximum number of iterations has been reached.

By finding the solutions where the gradients are 0, we can derive the update rules of $\mathbf{A}$ and $\mathcal{R}_k$ as below.

$$\mathbf{A} \leftarrow \left[\sum_k \mathcal{X}_k \mathbf{A}\mathcal{R}_k^T + \mathcal{X}_k^T \mathbf{A}\mathcal{R}_k\right]\left[\sum_k \mathcal{B}_k + \mathcal{C}_k + \lambda \mathbf{I}\right]^{-1},$$

where $\mathcal{B}_k = \mathcal{R}_k \mathbf{A}^T \mathbf{A} \mathcal{R}_k^T$ and $\mathcal{C}_k = \mathcal{R}_k^T \mathbf{A}^T \mathbf{A} \mathcal{R}_k$.

$$\mathbf{vec}(\mathcal{R}_k) \leftarrow \left(\mathbf{Z}^T\mathbf{Z} + \lambda \mathbf{I}\right)^{-1} \mathbf{Z}^T \mathbf{vec}(\mathcal{X}_k), \quad (3)$$

where $\mathbf{vec}(\mathcal{R}_k)$ is the vectorization of $\mathcal{R}_k$, $\mathbf{Z} = \mathbf{A} \otimes \mathbf{A}$ and the operator $\otimes$ is the Kronecker product.

Complexity Analysis Following the analysis in (Nickel et al., 2012), we assume that each $\mathcal{X}_k$ is a sparse matrix, and let p be the number of non-zero entries[2]. The complexity of computing $\mathcal{X}_k \mathbf{A} \mathcal{R}_k^T$ and $\mathcal{X}_k^T \mathbf{A} \mathcal{R}_k$ is $O(pr + nr^2)$. Evaluating $\mathcal{B}_k$ and $\mathcal{C}_k$ requires $O(nr^2)$ and the matrix inversion requires $O(r^3)$. Therefore, the complexity of updating $\mathbf{A}$ is $O(pr+nr^2)$ assuming $n \gg r$. The updating rule of R_k involves inverting an $r^2 \times r^2$ matrix. Therefore, directly computing the inversion requires time complexity $O(r^6)$ and space complexity $O(r^4)$. Although Nickel et al. (2012) considered using QR decomposition to simplify the updates, it is still time consuming with the time complexity $O(r^6 + pr^2)$. Therefore, the total time complexity is $O(r^6+pr^2)$ and the step of updating $\mathcal{R}_k$ is the bottleneck in the optimization process. We will describe how to reduce the time complexity of this step to $O(nr^2 + pr)$ in Section 4.2.

4 Approach

We describe how we leverage the relational domain knowledge in this section. By removing the incompatible entity-relation triples from the loss function, training can be done much more efficiently and results in a model with higher prediction accuracy. In addition, we also introduce a mathematical technique to reduce the computational complexity of the tensor decomposition methods when taking into account the regularization term.

4.1 Applying Relational Domain Knowledge

In the domain of knowledge bases, the notion of entity *types* is the side information that commonly exists and dictates whether some entities can be legitimate arguments of a given predicate. For instance, suppose the relation of interest is *born-in*, which denotes the birth *location* of a *person*. When asked whether an incompatible pair of entities, such as two *person* entities like `Abraham Lincoln` and `John Henry`, having this relation, we can immediately reject the possibility. Although the type information and the constraints are readily available, it is overlooked in the previous work on matrix and tensor decomposition models for knowledge bases (Riedel et al., 2013; Nickel et al., 2012). Ignoring the type information has two implications. Incompatible entity-relation triples still participate in the loss function of the optimization problem, which incurs unnecessary computation. Moreover, by choosing values for these incompatible entries we introduce errors in training the model that can reduce the quality of the model.

Based on this observation, we propose *Typed*-RESCAL, or TRESCAL, which leverages the entity type information to improve both the efficiency of model training and the quality of the model in term of prediction accuracy. We employ a direct and simple approach by excluding the triples of the incompatible entity types from the loss in Eq. (2). For each relation, let $\mathbb{L}_k$ and $\mathbb{R}_k$ be the set of entities with a compatible type to the k-th relation. That is, (e_i, r_k, e_j) is a feasible triple if and only if $e_i \in \mathbb{L}_k$ and $e_j \in \mathbb{R}_k$. For notational convenience, we use $\mathbf{A}_{k_l}, \mathbf{A}_{k_r}$ to denote the sub-matrices of $\mathbf{A}$ that consists of rows associated with $\mathbb{L}_k$ and $\mathbb{R}_k$, respectively. Analogously, let $\mathcal{X}_{k_{lr}}$ be the sub-matrix of $\mathcal{X}_k$ that consists of only the entity pairs compatible to the k-th relation. The rows and columns of $\mathcal{X}_{k_{lr}}$ map to the entities in $\mathbf{A}_{k_l}$ and $\mathbf{A}_{k_r}$, respectively. In other words, entries of $\mathcal{X}_k$ but not in $\mathcal{X}_{k_{lr}}$ do not satisfy the type constraint and are ignored from the computation.

[2]Notice that we use a slightly different definition of p from the one in (Nickel et al., 2012). The time complexity of multiplying an $n \times n$ sparse matrix $\mathcal{X}_k$ with p non-zero entries by an $n \times r$ dense matrix is $O(pr)$ assuming $n \gg r$.

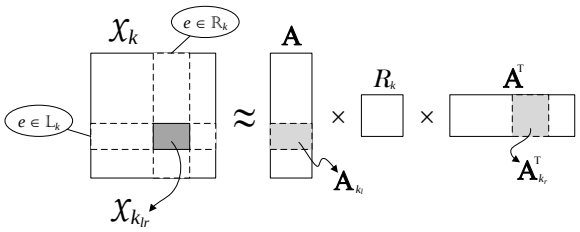


Figure 2: The construction of TRESCAL. Suppose the k-th relation is *born-in*. $\mathbb{L}_k$ is then a set of *person* entities and $\mathbb{R}_k$ is a set of *location* entities. Only the sub-matrix corresponds to the compatible entity pairs (i.e., $\mathcal{X}_{k_{lr}}$) and the sub-matrices of the associated entities (i.e., $\mathbf{A}_{k_l}$ and $\mathbf{A}^T_{k_r}$) will be included in the loss.

Fig. 2 illustrates this construction.

TRESCAL solves the following optimization problem:

$$\min_{\mathbf{A},\mathcal{R}_k} f'(\mathbf{A},\mathcal{R}_k) + \lambda \cdot g(\mathbf{A},\mathcal{R}_k), \qquad (4)$$

where $f'(\mathbf{A},\mathcal{R}_k) = \frac{1}{2}\sum_k \|\mathcal{X}_{k_{lr}} - \mathbf{A}_{k_l}\mathcal{R}_k\mathbf{A}^T_{k_r}\|^2_F$ and $g(\mathbf{A},\mathcal{R}_k) = \frac{1}{2}\left(\|\mathbf{A}\|^2_F + \sum_k \|\mathcal{R}_k\|^2_F\right)$.

Similarly, $\mathbf{A}$ and $\mathcal{R}_k$ can be solved using the alternating least-squares method. The update rule of $\mathbf{A}$ is

$$\mathbf{A} \leftarrow \left[\sum_k \left(\mathcal{X}_{k_{lr}}\mathbf{A}_{k_r}\mathcal{R}^T_k + \mathcal{X}^T_{k_{lr}}\mathbf{A}_{k_l}\mathcal{R}_k\right)\right] \times \left[\sum_k \mathcal{B}_{k_r} + \mathcal{C}_{k_l} + \lambda\mathbf{I}\right]^{-1},$$

where $\mathcal{B}_{k_r} = \mathcal{R}_k\mathbf{A}^T_{k_r}\mathbf{A}_{k_r}\mathcal{R}^T_k$ and $\mathcal{C}_{k_l} = \mathcal{R}^T_k\mathbf{A}^T_{k_l}\mathbf{A}_{k_l}\mathcal{R}_k$.

The update of $\mathcal{R}_k$ becomes:

$$\mathbf{vec}(\mathcal{R}_k) \leftarrow \left(\mathbf{A}^T_{k_r}\mathbf{A}_{k_r} \otimes \mathbf{A}^T_{k_l}\mathbf{A}_{k_l} + \lambda\mathbf{I}\right)^{-1} \times \mathbf{vec}({\mathbf{A}_{k_l}}^T\mathcal{X}_{k_{lr}}\mathbf{A}_{k_r}), \qquad (5)$$

Complexity Analysis Let $\bar{n}$ be the average number of entities with a compatible type to a relation. Follow a similar derivation in Sec. 3.2, the time complexity of updating $\mathbf{A}$ is $O(pr + \bar{n}r^2)$ and the time complexity of updating $\mathcal{R}_k$ remains to be $O(r^6 + pr^2)$.

4.2 Handling Regularization Efficiently

Examining the update rules of both RESCAL and TRESCAL, we can see that the most time-consuming part is the matrix inversions. For RESCAL, this is the term $(\mathbf{Z}^T\mathbf{Z}+\lambda\mathbf{I})^{-1}$ in Eq. (3), where $\mathbf{Z} = \mathbf{A} \otimes \mathbf{A}$. Nickel et al. (2011) made the observation that if $\lambda = 0$, the matrix inversion can be calculated by

$$(\mathbf{Z}^T\mathbf{Z})^{-1} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A} \otimes (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}.$$

Then, it only involves an inversion of an $r \times r$ matrix, namely $\mathbf{A}^T\mathbf{A}$. However, if $\lambda > 0$, directly calculating Eq. (3) requires to invert an $r^2 \times r^2$ matrix and thus becomes a bottleneck in solving Eq. (2).

To reduce the computational complexity of the update rules of $\mathcal{R}_k$, we compute the inversion $\left(\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I}\right)^{-1}$ by applying singular value decomposition (SVD) to $\mathbf{A}$, such that $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and $\mathbf{\Sigma}$ is a diagonal matrix. Then by using properties of the Kronecker product we have:

$$\begin{aligned}
&\left(\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I}\right)^{-1} \\
&\quad= \left(\lambda\mathbf{I} + \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T \otimes \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T\right)^{-1} \\
&\quad= \left(\lambda\mathbf{I} + (\mathbf{V}\otimes\mathbf{V})(\mathbf{\Sigma}^2\otimes\mathbf{\Sigma}^2)(\mathbf{V}\otimes\mathbf{V})^T\right)^{-1} \\
&\quad= (\mathbf{V}\otimes\mathbf{V})\left(\lambda\mathbf{I} + \mathbf{\Sigma}^2\otimes\mathbf{\Sigma}^2\right)^{-1}(\mathbf{V}\otimes\mathbf{V})^T.
\end{aligned}$$

The last equality holds because $\mathbf{V} \otimes \mathbf{V}$ is also an orthogonal matrix. We leave the detailed derivations in Appendix A. Notice that $\left(\lambda\mathbf{I} + \mathbf{\Sigma}^2 \otimes \mathbf{\Sigma}^2\right)^{-1}$ is a diagonal matrix. Therefore, the inversion calculation is trivial.

This technique can be applied to TRESCAL as well. By applying SVD to both $\mathbf{A}_{k_l}$ and $\mathbf{A}_{k_r}$, we have $\mathbf{A}_{k_l} = \mathbf{U}_{k_l}\mathbf{\Sigma}_{k_l}\mathbf{V}^T_{k_l}$ and $\mathbf{A}_{k_r} = \mathbf{U}_{k_r}\mathbf{\Sigma}_{k_r}\mathbf{V}^T_{k_r}$, respectively. The computation of $\left(\mathbf{A}^T_{k_r}\mathbf{A}_{k_r} \otimes \mathbf{A}^T_{k_l}\mathbf{A}_{k_l} + \lambda\mathbf{I}\right)^{-1}$ of Eq. (5) thus becomes:

$$(\mathbf{V}_{k_l} \otimes \mathbf{V}_{k_r})\left(\lambda\mathbf{I} + \mathbf{\Sigma}^2_{k_l} \otimes \mathbf{\Sigma}^2_{k_r}\right)^{-1}(\mathbf{V}_{k_l} \otimes \mathbf{V}_{k_r})^T.$$

The procedure of updating $\mathcal{R}$ is depicted in Algorithm 1.

Complexity Analysis For RESCAL, $\mathbf{V}$ and $\mathbf{\Sigma}$ can be computed by finding eigenvectors of $\mathbf{A}^T\mathbf{A}$. Therefore, computing SVD of $\mathbf{A}$ costs $O(nr^2 + r^3) = O(nr^2)$. Computing Step 4 in Algorithm 1 takes $O(nr^2 + pr)$. Step 5 and Step 6 require

Algorithm 1 Updating $\mathcal{R}$ in TRESCAL

Require: $\mathcal{X}$, $\mathbf{A}$, and entity sets $\mathbb{R}_k, \mathbb{L}_k, \forall k$
Ensure: $\mathcal{R}_k, \forall k$.

1: **for** $k = 1 \dots m$ **do**
2: $[\mathbf{U}_{k_l}, \mathbf{\Sigma}^2_{k_l}, \mathbf{V}_{k_l}] \leftarrow \text{SVD}(\mathbf{A}^T_{k_l}\mathbf{A}_{k_l})$.
3: $[\mathbf{U}_{k_r}, \mathbf{\Sigma}^2_{k_r}, \mathbf{V}_{k_r}] \leftarrow \text{SVD}(\mathbf{A}^T_{k_r}\mathbf{A}_{k_r})$.
4: $\mathbf{M}_1 \leftarrow \mathbf{V}^T_{k_l}\mathbf{A}^T_{k_l}\mathcal{X}_{k_{lr}}\mathbf{A}_{k_r}\mathbf{V}_{k_r}$.
5: $\mathbf{M}_2 \leftarrow \text{diag}(\mathbf{\Sigma}^2_{k_l})\,\text{diag}(\mathbf{\Sigma}^2_{k_r})^T + \lambda\mathbf{1}$.
(**1** is a matrix of all ones. Function diag converts the diagonal entries of a matrix to a vector.)
6: $\mathcal{R}_k \leftarrow \mathbf{V}_{k_l}(M_1./M_2)\mathbf{V}^T_{k_r}$.
(The operator "./" is element-wise division.)
7: **end for**

$O(r^2)$ and $O(r^3)$, respectively. The overall time complexity of updating $\mathcal{R}_k$ becomes $O(nr^2 + pr)$.

Using a similar derivation, the time complexity of updating $\mathcal{R}_k$ in TRESCAL is $O(\bar{n}r^2 + pr)$. Therefore, the total complexity of each iteration is $O(\bar{n}r^2 + pr)$.

5 Experiments

We conduct two sets of experiments. The first evaluates the proposed TRESCAL algorithm on inferring unknown facts using existing relation–entity triples, while the second demonstrates its application to relation extraction when a text corpus is available.

5.1 Knowledge Base Completion

We evaluate our approach on a knowledge base generated by the CMU Never Ending Language Learning (NELL) project (Carlson et al., 2010). NELL collects human knowledge from the web and has generated millions of entity-relation triples. We use the data generated from version 165 for training[3], and collect the new triples generated between NELL versions 166 and 533 as the development set and those generated between version 534 and 745 as the test set[4]. The data statistics of the training set are summarized in Table 1. The numbers of triples in the development and test sets are 19,665 and 117,889, respectively. Notice that this dataset is substantially larger than the datasets used in recent work. For example, the Freebase data used in (Socher et al., 2013) and (Bordes et al., 2013a) have 316k and 483k[5] triples, respectively, compared to 1.8M in this dataset.

NELL	
# entities	753k
# relation types	229
# entity types	300
# entity-relation triples	1.8M

Table 1: Data statistics of the training set from NELL in our experiments.

In the NELL dataset, the entity type information is encoded in a specific relation, called *Generalization*. Each entity in the knowledge base is assigned to at least one category presented by the Generalization relationship. Based on this information, the compatible entity type constraint of each relation can be easily identified. Specifically, we examined the entities and relations that occur in the triples of the training data, and counted all the types appearing in these instances of a given relation legitimate.

We implement RESCAL and TRESCAL in MATLAB with the Matlab tensor Toolbox (Bader et al., 2012). With the efficient implementation described in Section 4.2, all experiments can be conducted on a commodity PC with 16 GB memory. We set the maximal number of iterations of both RESCAL and TRESCAL to be 10, which we found empirically to be enough to generate a stable model. Note that Eq. (4) is non-convex, and the optimization process does not guarantee to converge to a global minimum. Therefore, initializing the model properly might be important for the performance. Following the implementation of RESCAL, we initialize $\mathbf{A}$ by performing singular value decomposition over $\bar{\mathcal{X}} = \sum_k(\mathcal{X}_k + \mathcal{X}_k^T)$, such that $\bar{\mathcal{X}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ and set $\mathbf{A} = \mathbf{U}$. Then, we apply the update rule of $\mathcal{R}_k$ to initialize $\{\mathcal{R}_k\}$. RESCAL and TRESCAL have two types of parameters: (1) the rank r of the decomposed tensor and (2) the regularization parameter λ. We tune the rank parameter on development set in a range of $\{100, 200, 300, 400\}$ and the regularization parameter in a range of $\{0.01, 0.05, 0.1, 0.5, 1\}$.

For comparison, we also use the code released by Bordes et al. (2013a), which is implemented using Python and the Theano library (Bergstra et al., 2010), to train a TransE model using the

[3] http://www.cs.cmu.edu/~nlao/
[4] http://bit.ly/trescal
[5] In (Bordes et al., 2013a), there is a much larger dataset, FB1M, that has 17.5M triples used for evaluation. However, this dataset has not been released.

	Entity Retrieval			Relation Retrieval		
	TransE	RESCAL	TRESCAL	TransE	RESCAL	TRESCAL
w/o type checking	51.41%‡	51.59%	54.79%	75.88%	73.15%†	76.12%
w/ type checking	67.56%	62.91%‡	69.26%	70.71%‡	73.08%†	75.70%

Table 2: Model performance in mean average precision (MAP) on *entity retrieval* and *relation retrieval*. † and ‡ indicate the comparison to TRESCAL in the same setting is statistically significant using a paired-t test on average precision of each query, with $p < 0.01$ and $p < 0.05$, respectively. Enforcing type constraints during test time improves entity retrieval substantially, but does not help in relation retrieval.

same NELL dataset. We reserved randomly 1% of the training triples for the code to evaluate the model performance in each iteration. As suggested in their paper, we experiment with several hyper-parameters, including learning rate of {0.01, 0.001}, the latent dimension of {50, 100} and the similarity measure of {L1, L2}. In addition, we also adjust the number of batches of {50, 100, 1000}. Of all the configurations, we keep the models picked by the method, as well as the final model after 500 training iterations. The final model is chosen by the performance on our development set.

5.1.1 Training Time Reduction

We first present experimental results demonstrating that TRESCAL indeed reduces the time required to factorize a knowledge database, compared to RESCAL. The experiment is conducted on NELL with $r = 300$ and $\lambda = 0.1$. When $\lambda \neq 0$, the original RESCAL algorithm described in (Nickel et al., 2011; Nickel et al., 2012) cannot handle a large r, because updating matrices $\{\mathcal{R}_k\}$ requires $O(r^4)$ memory. Later in this section, we will show that in some situation a large rank r is necessary for achieving good testing performance.

Comparing TRESCAL with RESCAL, each iteration of TRESCAL takes 1,608 seconds, while that of RESCAL takes 7,415 seconds. In other words, by inducing the entity type information and constraints, TRESCAL enjoys around 4.6 times speed-up, compared to an improved regularized version of RESCAL. When updating $\mathbf{A}$ and $\{\mathcal{R}_k\}$ TRESCAL only requires operating on sub-matrices of $\mathbf{A}$, $\{\mathcal{R}_k\}$ and $\{\mathcal{X}_k\}$, which reduces the computation substantially. In average, TRESCAL filters 96% of entity triples that have incompatible types.

In contrast, it takes TransE at least 2 days and 19 hours to finish training the model (the default 500 iterations)[6], while TRESCAL finishes the training in roughly 4 to 5 hours[7].

5.1.2 Test Performance Improvement

We consider two different types of tasks to evaluate the prediction accuracy of different models – *entity retrieval* and *relation retrieval*.

Entity Retrieval In the first task, we collect a set of entity-relation pairs $\{(e_i, r_k)\}$ and aim at predicting e_j such that the tuple (e_i, r_k, e_j) is a recorded triple in the NELL knowledge base. For each pair (e_i, r_k), we collect triples $\{(e_i, r_k, e_j^*)\}$ from the NELL test corpus as positive samples and randomly pick 100 entries e'_j to form negative samples $\{e_i, r_k, e'_j\}$. Given $\mathbf{A}$ and $\mathcal{R}_k$ from the factorization generated by RESCAL or TRESCAL, the score assigned to a triple $\{e_i, r_k, e'_j\}$ is computed by $a_i^T R_k a_j$ where a_i and a_j are the i-th and j-th rows of $\mathbf{A}$. In TransE, the score is determined by the negative dissimilarity measures of the learned embeddings: $-d(e_i, r_k, e'_j) = -\|\mathbf{e_i} + \mathbf{r_k} - \mathbf{e'_j}\|_2^2$.

We evaluate the performance using mean average precision (MAP), which is a robust and stable metric (Manning et al., 2008). As can be observed in Table 2 (left), TRESCAL achieves 54.79%, which outperforms 51.59% of RESCAL and 51.41% of TransE. Adding constraints during test time by assigning the lowest score to the entity triples with incompatible types improves results of all models – TRESCAL still performs the best (69.26%), compared to TransE (67.56%) and RESCAL (62.91%).

Relation Retrieval In the second task, given a relation type r_k, we are looking for the entity pairs (e_i, e_j) that have this specific relationship. To generate test data, for each relation type, we collect

[6]It took almost 4 days to train the best TransE model that is included in Table 2.

[7]We also tested the released code from (Socher et al., 2013) for training a neural tensor network model. However, we are not able to finish the experiments as each iteration of this method takes almost 5 hours.

gold entity pairs from the NELL knowledge base as positive samples and randomly pick a set of entity pairs as negative samples such that the number of positive samples are the same as negative ones.

Results presented in Table 2 (right) show that TRESCAL achieves 76.12%, while RESCAL and TransE are 73.15% and 75.88%, respectively. Therefore, incorporating the type information in training seems to help in this task as well. Enforcing the type constraints during test time does not help as in *entity retrieval*. By removing incompatible entity pairs, the performance of TRESCAL, RESCAL and TransE drop slightly to 75.70%, 73.08% and 70.71% respectively. One possible explanation is that the task of *relation retrieval* is easier than *entity retrieval*. The incorrect type information of some entities ends up filtering out a small number of entity pairs that were retrieved correctly by the model.

Notice that TRESCAL achieves different levels of performance on various relations. For example, it performs well on predicting *AthletePlaysSport* (81%) and *CoachesInLeague* (88%), but achieves suboptimal performance on predicting *WorksFor* (49%) and *BuildingLocatedInCity* (35%). We hypothesize that it is easier to generalize entity-relation triples when the relation has several related relations. For examples, *AthletePlaysForTeam* and *TeamPlaysSport* may help discover entity-relation triples of AthletePlaysSport.

5.1.3 Sensitivity to Parameters

We also study if TRESCAL is sensitive to the rank parameter r and the regularization parameter λ, where the detailed results can be found in Appendix B. In short, we found that increasing the rank r generally leads to better models. Also, while the model is not very sensitive to the value of the regularization parameter λ, tuning λ is still necessary for achieving the best performance.

5.2 Relation Extraction

Next, we apply TRESCAL to the task of extracting relations between entities, jointly from a text corpus and a structured knowledge base. We use a corpus from (Riedel et al., 2013) that is created by aligning the entities in NYTimes and Freebase. The corpus consists of a training set and a test set. In the training set, a list of entity pairs are provided, along with surface patterns extracted from NYTimes and known relations obtained from Freebase. In the test set, only the surface patterns are given. By jointly factoring a matrix consisting of the surface patterns and relations, Riedel et al. (2013) show that their model is able to capture the mapping between the surface patterns and the structured relations and hence is able to extract the entity relations from free text. In the following, we show that TRESCAL can be applied to this task.

We focus on the 19 relations listed in Table 1 of (Riedel et al., 2013) and only consider the surface patterns that co-occur with these 19 relations. We prune the surface patterns that occur less than 5 times and remove the entities that are not involved in any relation and surface pattern. Based on the training and test sets, we build a 80,698$\times$80,698$\times$1,652 tensor, where each slice captures a particular structured relation or a surface pattern between two entities. There are 72 fine types extracted from Freebase assigned to 53,836 entities that are recorded in Freebase. In addition, special types, PER, LOC, ORG and MISC, are assigned to the remaining 26,862 entities based on the predicted NER tags provided by the corpus. A type is considered incompatible to a relation or a surface pattern if in the training data, none of the argument entities of the relation belongs to the type. We use $r = 400$ and $\lambda = 0.1$ in TRESCAL to factorize the tensor.

We compare the proposed TRESCAL model to RI13 (Riedel et al., 2013), YA11 (Yao et al., 2011), MI09 (Mintz et al., 2009) and SU12 (Surdeanu et al., 2012)[8]. We follow the protocol used in (Riedel et al., 2013) to evaluate the results. Given a relation as query, the top 1,000 entity pairs output by each system are collected and the top 100 ones are judged manually. Besides comparing individual models, we also report the results of combined models. To combine the scores from two models, we simply normalize the scores of entity-relation tuples to zero mean and unit variance and take the average. The results are summarized in Table 3.

As can been seen in the table, using TRESCAL alone is not very effective and its performance is only compatible to MI09 and YA11, and is significantly inferior to RI13. This is understandable because the problem setting favors RI13 as only entity pairs that have occurred in the text or the database will be considered in RI13, both during model training and testing. In contrast, TRESCAL

[8]The corpus and the system outputs are from `http://www.riedelcastro.org/uschema`

Relation	#	MI09	YA11	SU12	RI13	TR	TR+SU12	TR+RI13
person/company	171	0.41	0.40	0.43	0.49	0.43	0.53	**0.64**
location/containedby	90	0.39	0.43	0.44	0.56	0.23	0.46	**0.58**
parent/child	47	0.05	0.10	0.25	0.31	0.19	0.24	**0.35**
person/place_of_birth	43	0.32	0.31	0.34	0.37	0.50	0.61	**0.66**
person/nationality	38	0.10	**0.30**	0.09	0.16	0.13	0.16	0.22
author/works_written	28	0.52	0.53	0.54	**0.71**	0.00	0.39	0.62
person/place_of_death	26	0.58	0.58	0.63	0.63	0.54	0.72	**0.89**
neighborhood/neighborhood_of	13	0.00	0.00	0.08	0.67	0.08	0.13	**0.73**
person/parents	8	0.21	0.24	**0.51**	0.34	0.01	0.16	0.38
company/founders	7	0.14	0.14	0.30	0.39	0.06	0.17	**0.44**
film/directed_by	4	0.06	0.15	0.25	0.30	0.03	0.13	**0.35**
sports_team/league	4	0.00	0.43	0.18	0.63	0.50	0.29	**0.63**
team/arena_stadium	3	0.00	0.06	0.06	0.08	0.00	0.04	**0.09**
team_owner/teams_owned	2	0.00	0.50	0.70	*0.75*	0.00	0.00	*0.75*
roadcast/area_served	2	*1.00*	0.50	*1.00*	*1.00*	0.50	0.83	*1.00*
structure/architect	2	0.00	0.00	*1.00*	*1.00*	0.00	0.02	*1.00*
composer/compositions	2	0.00	0.00	0.00	**0.12**	0.00	0.00	0.12
person/religion	1	0.00	*1.00*	*1.00*	*1.00*	0.00	*1.00*	*1.00*
film/produced_by	1	*1.00*	*1.00*	*1.00*	0.33	0.00	*1.00*	0.25
Weighted MAP		0.33	0.36	0.39	0.47	0.30	0.44	**0.57**

Table 3: Weighted Mean Average Precisions. The # column shows the number of true facts in the pool. Bold faced are winners per relation, italics indicate ties based on a sign test.

predicts all the possible combinations between entities and relations, which makes the model less fit to the task. However, when combining TRESCAL with a pure text-based method, such as SU12, we can clearly see TRESCAL is complementary to SU12 (0.39 to 0.44 in weighted MAP score), which makes the results competitive to RI13.

Interestingly, although both TRESCAL and RI13 leverage information from the knowledge base, we find that by combining them, the performance is improved quite substantially (0.47 to 0.57). We suspect that the reason is that in our construction, each entity has its own vector representation, which is lacked in RI13. As a result, the new triples that TRESCAL finds are very different from those found by RI13. Nevertheless, combining more methods do not always yield an improvement. For example, combining TR, RI13 and SU12 together (not included in Table 3) achieves almost the same performance as TR+RI13.

6 Conclusions

In this paper we developed TRESCAL, a tensor decomposition method that leverages relational domain knowledge. We use relational domain knowledge to capture which triples are potentially valid and found that, by excluding the triples that are incompatible when performing tensor decomposition, we can significantly reduce the training time and improve the prediction performance as compared with RESCAL and TransE. Moreover, we demonstrated its effectiveness in the application of relation extraction. Evaluated on the dataset provided in (Riedel et al., 2013), the performance of TRESCAL alone is comparable to several existing systems that leverage the idea of distant supervision. When combined with the state-of-the-art systems, we found that the results can be further improved. For instance, the weighted mean average precision of the previous best approach in (Riedel et al., 2013) has been increased by 10 points (47% to 57%).

There are a number of interesting potential extensions of our work. First, while the experiments in this paper are on traditional knowledge bases and textual data, the idea of leveraging relational domain knowledge is likely to be of value to other linguistic databases as well. For instance, part-of-speech tags can be viewed as the "types" of words. Incorporating such information in other tensor decomposition methods (e.g., (Chang et al., 2013)) may help lexical semantic representations. Second, relational domain knowledge goes beyond entity types and their compatibility with specific relations. For instance, the entity-relation triple (e_1, *child-of*, e_2) can be *valid* only if $e_1.\text{type} = \text{person} \wedge e_2.\text{type} = \text{person} \wedge e_1.\text{age} < e_2.\text{age}$. It would be interesting to explore the possibility of developing efficient methods to leverage other types of relational domain knowledge. Finally, we would like to create more sophisticated models of knowledge base embedding, targeting complex in-

ference tasks to better support semantic parsing and question answering.

Acknowledgments

We thank Sebastian Riedel for providing the data for experiments. We are also grateful to the anonymous reviewers for their valuable comments.

References

Brett W Bader, Richard A Harshman, and Tamara G Kolda. 2007. Temporal analysis of semantic graphs using ASALSAN. In *ICDM*, pages 33–42. IEEE.

Brett W. Bader, Tamara G. Kolda, et al. 2012. Matlab tensor toolbox version 2.5. Available online, January.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June. Oral Presentation.

Joanna Biega, Erdal Kuzey, and Fabian M Suchanek. 2013. Inside YOGO2s: a transparent information extraction architecture. In *WWW*, pages 325–328.

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. 2013a. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26*.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2013b. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, pages 1–27.

Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 576–583, Prague, Czech Republic, June. Association for Computational Linguistics.

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

Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1602–1612, Seattle, Washington, USA, October. Association for Computational Linguistics.

Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. 2000. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278.

Thomas Franz, Antje Schultz, Sergej Sizov, and Steffen Staab. 2009. Triplerank: Ranking semantic web data by tensor decomposition. In *The Semantic Web-ISWC 2009*, pages 213–228. Springer.

U Kang, Evangelos Papalexakis, Abhay Harpale, and Christos Faloutsos. 2012. Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. In *KDD*, pages 316–324. ACM.

Henk AL Kiers. 2000. Towards a standardized notation and terminology in multiway analysis. *Journal of chemometrics*, 14(3):105–122.

Tamara G. Kolda and Brett W. Bader. 2009. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, September.

Joseph B Kruskal. 1977. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear algebra and its applications*, 18(2):95–138.

Alan J Laub, 2005. *Matrix analysis for scientists and engineers*, chapter 13, pages 139–150. SIAM.

Ben London, Theodoros Rekatsinas, Bert Huang, and Lise Getoor. 2013. Multi-relational learning using weighted tensor decomposition with modular loss. Technical report, University of Maryland College Park. http://arxiv.org/abs/1303.1733.

C. Manning, P. Raghavan, and H. Schutze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, August. Association for Computational Linguistics.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, pages 809–816.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing YAGO: scalable machine learning for linked data. In *WWW*, pages 271–280.

Evangelos E Papalexakis, Tom M Mitchell, Nicholas D Sidiropoulos, Christos Faloutsos, Partha Pratim Talukdar, and Brian Murphy. 2014. Turbo-smt: Accelerating coupled sparse matrix-tensor factorizations by 200x. In *SDM*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML/PKDD 2010*. Springer.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL*, pages 74–84.

Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. 2013. Modeling missing data in distant supervision for information extraction. *Transactions of the Association for Computational Linguistics*, 1:367–378, October.

Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658. ACM.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In *Advances in Neural Information Processing Systems 26*.

Robert Speer, Catherine Havasi, and Henry Lieberman. 2008. Analogyspace: Reducing the dimensionality of common sense knowledge. In *AAAI*, pages 548–553.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Ilya Sutskever, Joshua B Tenenbaum, and Ruslan Salakhutdinov. 2009. Modelling relational data using Bayesian clustered tensor factorization. In *NIPS*, pages 1821–1828.

Ledyard R Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1366–1371, Seattle, Washington, USA, October. Association for Computational Linguistics.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1456–1466, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Appendix A Detailed Derivation

We first introduce some lemmas that will be useful for our derivation. Lemmas 2, 3 and 4 are the basic properties of the Kronecker product. Their proofs can be found at (Laub, 2005).

Lemma 1. *Let* $\mathbf{V}$ *be an orthogonal matrix and* $\mathbf{\Sigma}$ *a diagonal matrix. Then* $(\mathbf{I} + \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T)^{-1} = \mathbf{V}(\mathbf{I} + \mathbf{\Sigma})^{-1}\mathbf{V}^T$.

Proof.

$$\begin{aligned}(\mathbf{I} + \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T)^{-1} &= (\mathbf{V}\mathbf{I}\mathbf{V}^T + \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T)^{-1} \\ &= \mathbf{V}(\mathbf{I} + \mathbf{\Sigma})^{-1}\mathbf{V}^T \quad \square\end{aligned}$$

Lemma 2. $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}$.

Lemma 3. $(\mathbf{A} \otimes \mathbf{B})^T = \mathbf{A}^T \otimes \mathbf{B}^T$.

Lemma 4. *If* $\mathbf{A}$ *and* $\mathbf{B}$ *are orthogonal matrices, then* $\mathbf{A} \otimes \mathbf{B}$ *will also be an orthogonal matrix.*

Let $\mathbf{Z} = \mathbf{A} \otimes \mathbf{A}$ and apply singular value decomposition to $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. The term $\left(\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I}\right)^{-1}$ can be rewritten as:

$$\begin{aligned}&\left(\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I}\right)^{-1} \\ &\quad= \left(\lambda\mathbf{I} + (\mathbf{A}^T \otimes \mathbf{A}^T)(\mathbf{A} \otimes \mathbf{A})\right)^{-1} && (6)\\ &\quad= \left(\lambda\mathbf{I} + \mathbf{A}^T\mathbf{A} \otimes \mathbf{A}^T\mathbf{A}\right)^{-1} && (7)\\ &\quad= \left(\lambda\mathbf{I} + \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T \otimes \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T\right)^{-1} && (8)\\ &\quad= \left(\lambda\mathbf{I} + (\mathbf{V} \otimes \mathbf{V})(\mathbf{\Sigma}^2 \otimes \mathbf{\Sigma}^2)(\mathbf{V} \otimes \mathbf{V})^T\right)^{-1} && (9)\\ &\quad= (\mathbf{V} \otimes \mathbf{V})\left(\lambda\mathbf{I} + \mathbf{\Sigma}^2 \otimes \mathbf{\Sigma}^2\right)^{-1}(\mathbf{V} \otimes \mathbf{V})^T && (10)\end{aligned}$$

Eq. (6) is from replacing $\mathbf{Z}$ with $\mathbf{A} \otimes \mathbf{A}$ and Lemma 3. Eq. (7) is from Lemma 2. Eq. (8) is from the properties of SVD, where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices. Eq. (9) is from Lemma 2 and Lemma 3. Finally, Eq. (10) comes from Lemma 1.

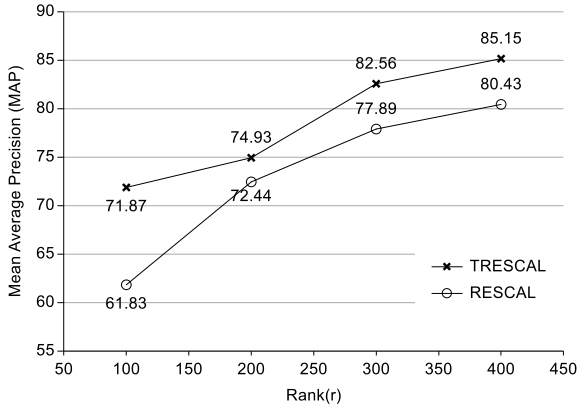


Figure 3: Prediction performance of TRESCAL and RESCAL with different rank (r).

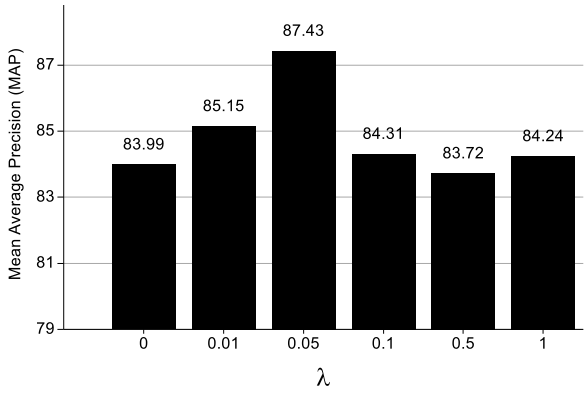


Figure 4: Prediction performance of TRESCAL with different regularization parameter (λ).

Appendix B Hyper-parameter Sensitivity

We study if TRESCAL is sensitive to the rank parameter r and the regularization parameter λ. We use the task of *relation retrieval* and present the model performance on the development set. Fig. 3 shows the performance of TRESCAL and RESCAL with different rank (r) values while fixing $\lambda = 0.01$. Results show that both TRESCAL and RESCAL achieve better performance when r is reasonably large. TRESCAL obtains a better model with smaller r than RESCAL, because TRESCAL only needs to fit the triples of the compatible entity types. Therefore, it allows to use smaller number of latent variables to fit the training data.

Fixing $r = 400$, Fig. 4 shows the performance of TRESCAL at different values of the regularization parameter λ, including no regularization at all ($\lambda = 0$). While the results suggest that the method is not very sensitive to λ, tuning λ is still necessary for achieving the best performance.

A convex relaxation for weakly supervised relation extraction

Édouard Grave
EECS Department
University of California, Berkeley
grave@berkeley.edu

Abstract

A promising approach to relation extraction, called weak or distant supervision, exploits an existing database of facts as training data, by aligning it to an unlabeled collection of text documents. Using this approach, the task of relation extraction can easily be scaled to hundreds of different relationships. However, distant supervision leads to a challenging multiple instance, multiple label learning problem. Most of the proposed solutions to this problem are based on non-convex formulations, and are thus prone to local minima. In this article, we propose a new approach to the problem of weakly supervised relation extraction, based on discriminative clustering and leading to a convex formulation. We demonstrate that our approach outperforms state-of-the-art methods on the challenging dataset introduced by Riedel et al. (2010).

1 Introduction

Information extraction refers to the broad task of automatically extracting structured information from unstructured documents. An example is the extraction of named entities and the relations between those entities from natural language texts. In the age of the world wide web and big data, information extraction is quickly becoming pervasive. For example, in 2013, more than $130,000$ scientific articles were published about cancer. Keeping track with that quantity of information is almost impossible, and it is thus of utmost importance to transform the knowledge contained in this massive amount of documents into structured databases.

Traditional approaches to information extraction relies on supervised learning, yielding high precision and recall results (Zelenko et al., 2003). Unfortunately, these approaches need large amount of labeled data, and thus do not scale well to the great number of different types of fact found on the Web or in scientific articles. A promising approach, called distant or weak supervision, is to exploit an existing database of facts as training data, by aligning it to an unlabeled collection of text documents (Craven and Kumlien, 1999).

Knowledge base

r	e_1	e_2
BornIn	Lichtenstein	New York City
DiedIn	Lichtenstein	New York City

Sentences	Latent labels
Roy Lichtenstein was born in New York City, into an upper-middle-class family.	BornIn
In 1961, Leo Castelli started displaying Lichtenstein's work at his gallery in New York.	None
Lichtenstein died of pneumonia in 1997 in New York City.	DiedIn

Figure 1: An example of a knowledge database comprising two facts and training sentences obtained by aligning this database to unlabeled text.

In this article, we are interested in weakly supervised extraction of binary relations. A challenge pertaining to weak supervision is that the obtained training data is noisy and ambiguous (Riedel et al., 2010). Let us start with an example: if the fact Attended(Turing, King's College) exists in the knowledge database and we observe the sentence

Turing studied as an undergraduate from 1931 to 1934 at King's College, Cambridge.

which contains mentions of both entities Turing

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1580–1590,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

and King's College, then this sentence might express the fact that Alan Turing attended King's College, and thus, might be a useful example for learning to extract the relation Attended. However, the sentence

Celebrations for the centenary of Alan Turing are being planned at King's College.

also contains mentions of Turing and King's College, but do not express the relation Attended. Thus, weak supervision lead to noisy examples. As noted by Riedel et al. (2010), such negative extracted sentences for existing facts can represent more than 30% of the data. Moreover, a given pair of entities, such as (Roy Lichtenstein, New York City), car verify multiple relations, such as BornIn and DiedIn. Weak supervision thus lead to ambiguous examples.

This challenge is illustrated in Fig. 1. A solution to address it is to formulate the task of weakly supervised relation extraction as a multiple instance, multiple label learning problem (Hoffmann et al., 2011; Surdeanu et al., 2012). However, these formulations are often non-convex and thus suffer from local minimum.

In this article, we make the following contributions:

- We propose a new convex relaxation for the problem of weakly supervised relation extraction, based on discriminative clustering,
- We propose an efficient algorithm to solve the associated convex program,
- We demonstrate that our approach obtains state-of-the-art results on the dataset introduced by Riedel et al. (2010).

To our knowledge, this paper is the first to propose a convex formulation for solving the problem of weakly supervised relation extraction.

2 Related work

Supervised learning. Many approaches based on supervised learning have been proposed to solve the problem of relation extraction, and the corresponding literature is to large to be summarized here. One of the first supervised method for relation extraction was inspired by syntactic parsing: the system described by Miller et al. (1998) combines syntactic and semantic knowledge, and thus, part-of-speech tagging, parsing, named entity recognition and relation extraction all happen at the same time. The problem of relation extraction was later formulated as a classification problem: Kambhatla (2004) proposed to solve this problem using maximum entropy models using lexical, syntactic and semantic features. Kernel methods for relation extraction, based on shallow parse trees or dependency trees were introduced by Zelenko et al. (2003), Culotta and Sorensen (2004) and Bunescu and Mooney (2005).

Unsupervised learning. The open information extraction paradigm, simultaneously proposed by Shinyama and Sekine (2006) and Banko et al. (2007), does not rely on any labeled data or even existing relations. Instead, open information extraction systems only use an unlabeled corpus, and output a set of extracted relations. Such systems are based on clustering (Shinyama and Sekine, 2006) or self-supervision (Banko et al., 2007). One of the limitations of these systems is the fact that they extract uncanonicalized relations.

Weakly supervised learning. Weakly supervised learning refers to a broad class of methods, in which the learning system only have access to partial, ambiguous and noisy labeling. Craven and Kumlien (1999) were the first to propose a weakly supervised relation extractor. They aligned a knowledge database (the Yeast Protein Database) with scientific articles mentioning a particular relation, and then used the extracted sentences to learn a classifier for extracting that relation.

Later, many different sources of weak labelings have been considered. Bellare and McCallum (2007) proposed a method to extract bibliographic relations based on conditional random fields and used a database of BibTex entries as weak supervision. Wu and Weld (2007) described a method to learn relations based on Wikipedia infoboxes. Knowledge databases, such as Freebase[1] (Mintz et al., 2009; Sun et al., 2011) and YAGO[2] (Nguyen and Moschitti, 2011) were also considered as a source of weak supervision.

Multiple instance learning. The methods we previously mentionned transform the weakly supervised problem into a fully supervised one, leading to noisy training datasets (see Fig. 1). Mul-

[1] www.freebase.com

[2] www.mpi-inf.mpg.de/yago-naga/yago

tiple instance learning (Dietterich et al., 1997) is a paradigm in which the learner receives bags of examples instead of individual examples. A positively labeled bag contains *at least one* positive example, but might also contains negative examples. In the context of relation extraction, Bunescu and Mooney (2007) introduced a kernel method for multiple instance learning, while Riedel et al. (2010) proposed a solution based on a graphical model.

Both these methods allow only one label per bag, which is an asumption that is not true for relation extraction (see Fig. 1). Thus, Hoffmann et al. (2011) proposed a multiple instance, multiple label method, based on an undirected graphical model, to solve the problem of weakly supervised relation extraction. Finally, Surdeanu et al. (2012) also proposed a graphical model to solve this problem. One of their main contributions is to capture dependencies between relation labels, such as the fact that two labels cannot be generated jointly (*e.g.* the relations `SpouseOf` and `BornIn`).

Discriminative clustering. Our approach is based on the discriminative clustering framework, introduced by Xu et al. (2004). The goal of discriminative clustering is to find a labeling of the data points leading to a classifier with low classification error. Different formulations of discriminative clustering have been proposed, based on support vector machines (Xu et al., 2004), the squared loss (Bach and Harchaoui, 2007) or the logistic loss (Joulin et al., 2010). A big advantage of discriminative clustering is that weak supervision or prior information can easily be incorporated. Our work is closely related to the method proposed by Bojanowski et al. (2013) for learning the names of characters in movies.

3 Weakly supervised relation extraction

In this article, our goal is to extract *binary relations* between entities from natural language text. Given a set of entities, a binary relation r is a collection of ordered pairs of entities. The statement that a pair of entities (e_1, e_2) belongs to the relation r is denoted by $r(e_1, e_2)$ and this triple is called a *fact* or *relation instance*. For example, the fact that Ernest Hemingway was born in Oak Park is denoted by `BornIn(Ernest Hemingway, Oak Park)`. A given pair of entities, such as (`Edouard Manet`, `Paris`), can belong to different relations, such as `BornIn` and `DiedIn`.

An *entity mention* is a contiguous sequence of tokens refering to an entity, while a *pair mention* or *relation mention candidate* is a sequence of text in which a pair of entities is mentioned. In the following, relation mention candidates will be restricted to pair of entities that are mentioned in the same sentence. For example, the sentence:

Ernest Hemingway was born in Oak Park.

contains two entity mentions, corresponding to two relation mention candidates. Indeed, the pairs (`Hemingway`, `Oak Park`) and (`Oak Park`, `Hemingway`) are two distinct pairs of entities, where only the first one verifies the relation `BornIn`.

Given a text corpus, *aggregate extraction* corresponds to the task of extracting a set of facts, such that each extracted fact is expressed at least once in the corpus. On the other hand, the task of *sentential extraction* corresponds to labeling each relation mention candidate by the relation it expresses, or by a `None` label if it does not express any relation. Given a solution to the sentential extraction problem, it is possible to construct a solution for the aggregate extraction problem by returning all the facts that were detected. We will follow this approach, by building an instance level classifier, and aggregating the results by extracting the facts that were detected at least once in the corpus.

In the following, we will describe a method to learn such a classifier using a database of facts instead of a set of labeled sentences. This setting is known as *distant supervision* or *weak supervision*, since we do not have access to labeled data on which we could directly train a sentence level relation extractor.

4 General approach

In this section, we propose a two step procedure to solve the problem of weakly supervised relation extraction:

1. First, we describe a method to infer the relation labels corresponding to each relation mention candidate of our training set,

2. Second, we train a supervised instance level relation extractor, using the labels infered during step 1.

In the second step of our approach, we will simply use a multinomial logistic regression model. We

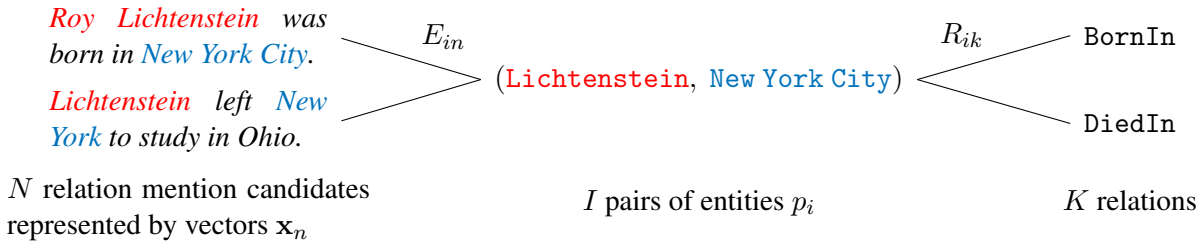


Figure 2: Instance of the weakly supervised relation extraction problem, with notations used in the text.

now describe the approach we propose for the first step.

4.1 Notations

Let $(p_i)_{1\leq i\leq I}$ be a collection of I pairs of entities. We suppose that we have N relation mention candidates, represented by the vectors $(\mathbf{x}_n)_{1\leq n\leq N}$. Let $\mathbf{E}\in\mathbb{R}^{I\times N}$ be a matrix such that $E_{in}=1$ if the relation mention candidate n corresponds to the pair of entities i, and $E_{in}=0$ otherwise. The matrix $\mathbf{E}$ thus indicates which relation mention candidate corresponds to which pair of entities. We suppose that we have K relations, indexed by the integers $\{1,...,K\}$. Let $\mathbf{R}\in\mathbb{R}^{I\times K}$ be a matrix such that $R_{ik}=1$ if the pair of entities i verifies the relation k, and $R_{ik}=0$ otherwise. The matrix $\mathbf{R}$ thus represents the knowledge database. See Fig. 2 for an illustration of these notations.

4.2 Problem formulation

Our goal is to infer a binary matrix $\mathbf{Y}\in\{0,1\}^{N\times(K+1)}$, such that $Y_{nk}=1$ if the relation mention candidate n express the relation k and $Y_{nk}=0$ otherwise (and thus, the integer $K+1$ represents the relation `None`).

We take an approach inspired by the discriminative clustering framework of Xu et al. (2004). We are thus looking for a $(K+1)$-class indicator matrix $\mathbf{Y}$, such that the classification error of an optimal multiclass classifier f is minimum. Given a multiclass loss function ℓ and a regularizer Ω, this problem can be formulated as:

$$\min_{\mathbf{Y}}\min_{f}\ \sum_{n=1}^{N}\ell(\mathbf{y}_n,f(\mathbf{x}_n))+\Omega(f),$$
$$\text{s.t.}\quad \mathbf{Y}\in\mathcal{Y}$$

where $\mathbf{y}_n$ is the nth line of $\mathbf{Y}$. The constraints $\mathbf{Y}\in\mathcal{Y}$ are added in order to take into account the information from the weak supervision. We will describe in the next section what kind of constraints are considered.

4.3 Weak supervision by constraining $\mathbf{Y}$

In this section, we show how the information from the knowledge base can be expressed as constraints on the matrix $\mathbf{Y}$.

First, we suppose that each relation mention candidate express exactly one relation (including the `None` relation). This means that the matrix $\mathbf{Y}$ contains exactly one 1 per line, which is equivalent to the constraint:

$$\forall n\in\{1,...,N\},\ \sum_{k=1}^{K}Y_{nk}=1.$$

Second, if the pair i of entities verifies the relation k we suppose that *at least one* relation mention candidate indeed express that relation. Thus we want to impose that for at least one relation mention candidate n such that $E_{in}=1$, we have $Y_{nk}=1$. This is equivalent to the constraint:

$$\forall(i,k)\text{ such that }R_{ik}=1,\sum_{n=1}^{N}E_{in}Y_{nk}\geq 1.$$

Third, if the pair i of entities does not verify the relation k, we suppose that no relation mention candidate express that relation. Thus, we impose that for all mention candidate n such that $E_{in}=1$, we have $Y_{nk}=0$. This is equivalent to the constraint:

$$\forall(i,k)\text{ such that }R_{ik}=0,\sum_{n=1}^{N}E_{in}Y_{nk}=0.$$

Finally, we do not want too many relation mention candidates to be classified as `None`. We thus impose

$$\forall i\in\{1,...,I\},\ \sum_{n=1}^{N}E_{in}Y_{n(K+1)}\leq c\sum_{n=1}^{N}E_{in},$$

where c is the proportion of relation mention candidates that do not express a relation, for entity pairs that appears in the knowledge database.

We can rewrite these constraints using only matrix operations in the following way:

$$\begin{aligned} &\mathbf{Y1} = \mathbf{1} \\ &(\mathbf{EY}) \circ \mathbf{S} \geq \tilde{\mathbf{R}}, \end{aligned} \tag{1}$$

where $\circ$ is the Hadamard product (a.k.a. the elementwise product), the matrix $\mathbf{S} \in \mathbb{R}^{I \times (K+1)}$ is defined by

$$S_{ik} = \begin{cases} 1 & \text{if } R_{ik} = 1 \\ -1 & \text{if } R_{ik} = 0 \text{ or } k = K+1, \end{cases}$$

and the matrix $\tilde{\mathbf{R}} \in \mathbb{R}^{I \times (K+1)}$ is defined by

$$\tilde{\mathbf{R}} = [\mathbf{R}, -c\mathbf{E1}].$$

The set $\mathcal{Y}$ is thus defined as the set of matrices $\mathbf{Y} \in \{0,1\}^{N \times (K+1)}$ that verifies those two linear constraints. It is important to note that besides the boolean constraints, the two other constraints are convex.

5 Squared loss and convex relaxation

In this section, we describe the problem we obtain when using the squared loss, and its associated convex relaxation. We then introduce an efficient algorithm to solve this problem, by computing its dual.

5.1 Primal problem

Following Bach and Harchaoui (2007), we use linear classifiers $\mathbf{W} \in \mathbb{R}^{D \times (K+1)}$, the squared loss and the squared ℓ_2-norm as the regularizer. In that case, our formulation becomes:

$$\begin{aligned} \min_{\mathbf{Y},\mathbf{W}} \quad & \frac{1}{2}\|\mathbf{Y} - \mathbf{XW}\|_F^2 + \frac{\lambda}{2}\|\mathbf{W}\|_F^2, \\ \text{s.t.} \quad & \mathbf{Y} \in \{0,1\}^{N \times (K+1)} \\ & \mathbf{Y1} = \mathbf{1}, \\ & (\mathbf{EY}) \circ \mathbf{S} \geq \mathbf{R}. \end{aligned}$$

where $\|\cdot\|_F$ is the Frobenius norm and the matrix $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^\top \in \mathbb{R}^{N \times D}$ represents the relation mention candidates. Thanks to using the squared loss, we have a closed form solution for the matrix $\mathbf{W}$:

$$\mathbf{W} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_D)^{-1}\mathbf{X}^\top\mathbf{Y}.$$

Replacing the matrix $\mathbf{W}$ by its optimal solution, we obtain the following cost function:

$$\min_{\mathbf{Y}} \frac{1}{2}\mathbf{Y}^\top(\mathbf{I}_N - \mathbf{X}(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_D)^{-1}\mathbf{X}^\top)\mathbf{Y}.$$

Then, by applying the Woodbury matrix identity and relaxing the constraint $\mathbf{Y} \in \{0,1\}^{N \times (K+1)}$ into $\mathbf{Y} \in [0,1]^{N \times (K+1)}$, we obtain the following convex quadratic problem in $\mathbf{Y}$:

$$\begin{aligned} \min_{\mathbf{Y}} \quad & \frac{1}{2}\text{tr}\left(\mathbf{Y}^\top(\mathbf{XX}^\top + \lambda\mathbf{I}_N)^{-1}\mathbf{Y}\right), \\ \text{s.t.} \quad & \mathbf{Y} \geq \mathbf{0}, \\ & \mathbf{Y1} = \mathbf{1}, \\ & (\mathbf{EY}) \circ \mathbf{S} \geq \mathbf{R}. \end{aligned}$$

Since the inequality constraints might be infeasible, we add the penalized slack variables $\xi \in \mathbb{R}^{I \times (K+1)}$, finally obtaining:

$$\begin{aligned} \min_{\mathbf{Y},\xi} \quad & \frac{1}{2}\text{tr}\left(\mathbf{Y}^\top(\mathbf{XX}^\top + \lambda\mathbf{I}_N)^{-1}\mathbf{Y}\right) + \mu\|\xi\|_1 \\ \text{s.t.} \quad & \mathbf{Y} \geq \mathbf{0}, \quad \xi \geq 0, \\ & \mathbf{Y1} = \mathbf{1}, \\ & (\mathbf{EY}) \circ \mathbf{S} \geq \mathbf{R} - \xi. \end{aligned}$$

This convex problem is a quadratic program. In the following section, we will describe how to solve this problem efficiently, by exploiting the structure of its dual problem.

5.2 Dual problem

The matrix $\mathbf{Q} = (\mathbf{XX}^\top + \lambda\mathbf{I}_N)$ appearing in the quadratic program is an N by N matrix, where N is the number of mention relation candidates. Computing its inverse is thus expensive, since N can be large. Instead, we propose to solve the dual of this problem. Introducing dual variables $\Lambda \in \mathbb{R}^{I \times (K+1)}$, $\Sigma \in \mathbb{R}^{N \times (K+1)}$ and $\nu \in \mathbb{R}^N$, the dual problem is equal to

$$\begin{aligned} \min_{\Lambda,\Sigma,\nu} \quad & \frac{1}{2}\text{tr}\left(\mathbf{Z}^\top\mathbf{QZ}\right) - \text{tr}\left(\Lambda^\top\mathbf{R}\right) - \nu^\top 1 \\ \text{s.t.} \quad & 0 \leq \Lambda_{ik} \leq \mu, \quad 0 \leq \Sigma_{nk}, \end{aligned}$$

where

$$\mathbf{Z} = \mathbf{E}^\top(\mathbf{S} \circ \Lambda) + \Sigma + \nu\mathbf{1}^\top.$$

The derivation of this dual problem is given in Appendix A.

Solving the dual problem instead of the primal has two main advantages. First, the dual does not depend on the inverse of the matrix $\mathbf{Q}$, while the primal does. Since traditional features used for relation extraction are indicators of lexical, syntactic and named entities properties of the relation mention candidates, the matrix $\mathbf{X}$ is extremely sparse.

Using the dual problem, we can thus exploit the sparsity of the matrix $\mathbf{X}$ in the optimization procedure. Second, the constraints imposed on dual variables are simpler than constraints imposed on primal variables. Again, we will exploit this structure in the proposed optimization procedure.

Given a solution of the dual problem, the associated primal variable $\mathbf{Y}$ is equal to:

$$\mathbf{Y} = (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_N)\mathbf{Z}.$$

Thus, we do not need to compute the inverse of the matrix $(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_N)$ to obtain a solution to the primal problem once we have solved the dual.

5.3 Optimization of the dual problem

We propose to solve the dual problem using the accelerated projected gradient descent algorithm (Nesterov, 2007; Beck and Teboulle, 2009). Indeed, computing the gradient of the dual cost function is efficient, since the matrix $\mathbf{X}$ is sparse. Moreover, the constraints on the dual variables are simple and it is thus efficient to project onto this set of constraints. See Appendix B for more details.

Complexity. The overall complexity of one step of the accelerated projected gradient descent algorithm is $O(NFK)$, where F is the average number of features per relation mention candidate. This means that the complexity of solving the quadratic problem corresponding to our approach is linear with respect to the number N of relation mention candidates, and thus our algorithm can scale to large datasets.

5.4 Discussion

Before moving to the experimental sections of this article, we would like to discuss some properties of our approach.

Kernels. First of all, one should note that our proposed formulation only depends on the (linear) kernel matrix $\mathbf{X}\mathbf{X}^T$. It is thus possible to replace this matrix by any other kernel. However, in the case of a general kernel, the optimization algorithm presented in the previous section has a quadratic complexity $O(KN^2)$ with respect to the number N of relation mention candidates, and it is thus not applicable *as is*. We plan to explore the use of kernels in future work.

Rounding. Given a continuous solution $\mathbf{Y} \in [0,1]^{N\times(K+1)}$ of the relaxed problem, a very simple way to obtain a relation label for each relation mention candidate of the training set is to compute the orthogonal projection of the matrix $\mathbf{Y}$ on the set of indicator matrices

$$\left\{\mathbf{M} \in \{0,1\}^{N\times(K+1)} \mid \mathbf{M}\mathbf{1} = \mathbf{1}\right\}.$$

This projection consists in taking the maximum value along the rows of the matrix $\mathbf{Y}$. It should be noted that the obtained matrix does not necessarily verify the inequality constraints defined in Eq. 1. In the following, we will use this rounding, refered to as *argmax rounding*, to obtain relation labels for each relation mention candidate.

6 Dataset and features

In this section, we describe the dataset used in the experimental section and the features used to represent the data.

6.1 Dataset

We consider the dataset introduced by Riedel et al. (2010). This dataset consists of articles from the New York Times corpus (Sandhaus, 2008), from which named entities where extracted and tagged using the Stanford named entity recognizer (Finkel et al., 2005). Consecutive tokens with the same category were treated as a single mention. These named entity mentions were then aligned with the Freebase knowledge database, by using a string match between the mentions and the canonical names of entities in Freebase.

6.2 Features

We use the features extracted by Riedel et al. (2010), which were first introduced by Mintz et al. (2009). These features capture how two entity mentions are related in a given sentence, based on syntactic and lexical properties. Lexical features include: the sequence of words between the two entities, a window of k words before the first entity and after the second entity, the corresponding part-of-speech tags, *etc.*. Syntactic features are based on the dependency tree of the sentence, and include: the path between the two entities, neighbors of the two entities that do not belong to the path. The OpenNLP[3] part-of-speech tagger and the Malt parser (Nivre et al., 2007) were used to extract those features.

[3] `opennlp.apache.org`

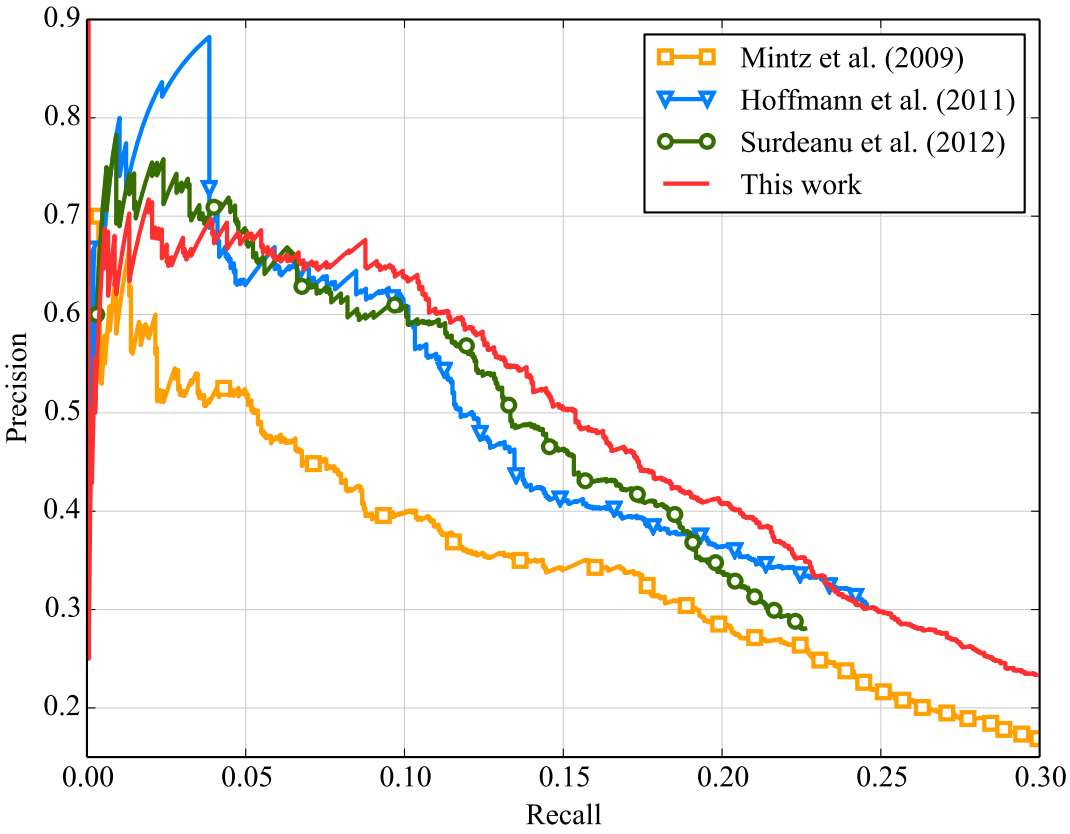


Figure 3: Precision/recall curves for different methods on the Riedel et al. (2010) dataset, for the task of aggregate extraction.

6.3 Implementation details

In this section, we discuss some important implementation details.

Kernel normalization. We normalized the kernel matrix $\mathbf{X}\mathbf{X}^\top$, so that its diagonal coefficients are equal to 1. This corresponds to normalizing the vectors $\mathbf{x}_n$ so that they have a unit ℓ_2-norm.

Choice of parameters. We kept 20% of the examples from the training set as a validation set, in order to choose the parameters of our method. We then re-train a model on the whole training set, using the chosen parameters.

7 Experimental evaluation

In this section, we evaluate our approach to weakly supervised relation extraction by comparing it to state-of-the art methods.

7.1 Baselines

We now briefly present the different methods we compare to.

Mintz et al. This baseline corresponds to the method described by Mintz et al. (2009). We use the implementation of Surdeanu et al. (2012), which slightly differs from the original method: each relation mention candidate is treated independently (and not collapsed across mentions for a given entity pair). This strategy allows to predict multiple labels for a given entity pair, by OR-ing the predictions for the different mentions.

Hoffmann et al. This method, introduced by Hoffmann et al. (2011), is based on probabilistic graphical model of multi-instance multi-label learning. They proposed a learning method for this model, based on the perceptron algorithm (Collins, 2002) and a greedy search for the inference. We use the publicly available code of Hoffmann *et al.*[4].

Surdeanu et al. Finally, we compare our method to the one described by Surdeanu et al. (2012). This method is based on a two-layer graphical model, the first layer corresponding to

[4] www.cs.washington.edu/ai/raphaelh/mr/

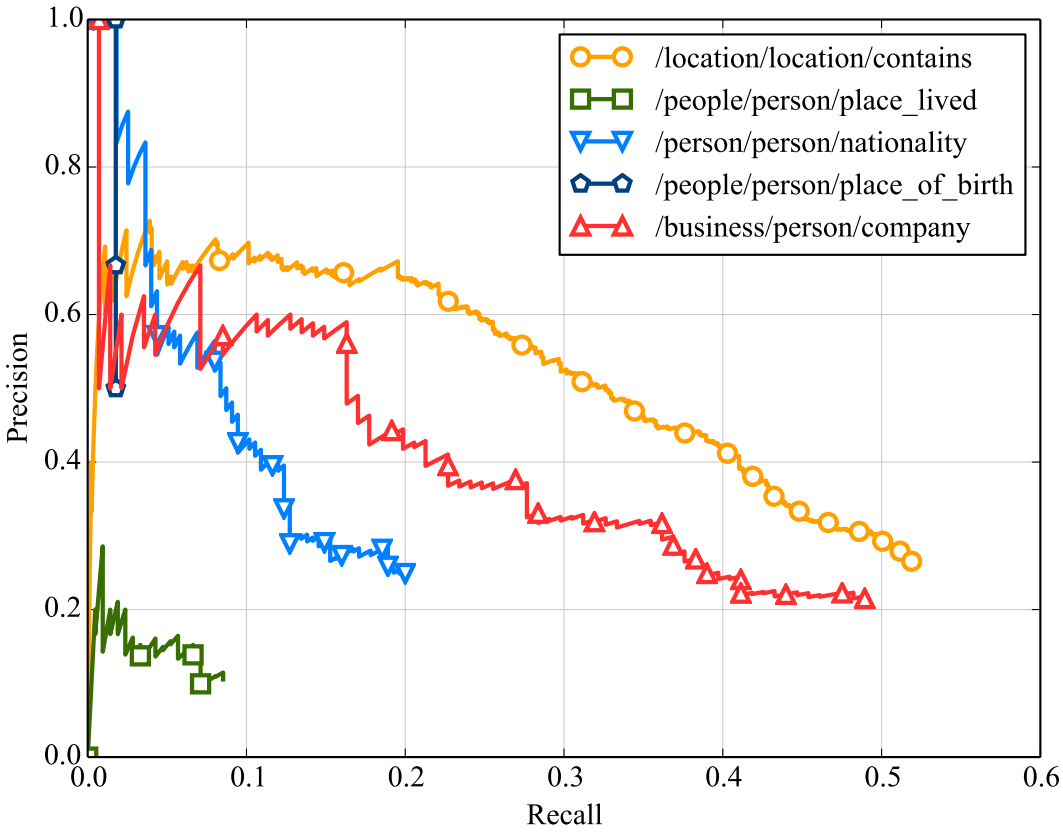


Figure 4: Precision/recall curves per relation for our method, on the Riedel et al. (2010) dataset, for the task of aggregate extraction.

a relation classifier at the mention level, while the second layer is aggregating the different prediction for a given entity pair. In particular, this second layer capture dependencies between relation labels, such as the fact that two labels cannot be generated jointly (*e.g.* the relations `SpouseOf` and `BornIn`). This model is trained by using hard discriminative Expectation-Maximization. We use the publicly available code of Surdeanu *et al.*[5].

7.2 Precision / recall curves

Following standard practices in relation extraction, we report precision/recall curves for the different models. In order to rank aggregate extractions for our model, the score of an extracted fact $r(e_1, e_2)$ is set to the maximal score of the different extractions of that fact. This is sometimes refered to as the soft-OR function.

7.3 Discussion

Comparison with the state-of-the-art. We report results for the different methods on the dataset introduced by Riedel et al. (2010) in Fig. 3. We observe that our approach generally outperforms the state of the art. Indeed, at equivalent recall, our method achieves better (or similar) precision than the other methods, except for very low recall (smaller than 0.05). The improvement over the methods proposed by Hoffmann et al. (2011) and Surdeanu et al. (2012), which are currently the best published results on this dataset, can be as high as 5 points in precision for the same recall point. Moreover, our method achieves a higher recall (0.30) than these two methods (0.25).

Performance per relation. The dataset introduced by Riedel et al. (2010) is highly unbalanced: for example, the most common relation, `/location/location/contains`, represents almost half of the positive relations, while some relations are mentioned less than ten times. We thus decided to also report precision/recall curves for the five most common relations of that dataset in Fig. 4. First, we observe that the perfomances vary a lot from a relation to another. The frequence of the different relations is not the

[5] `nlp.stanford.edu/software/mimlre.shtml`

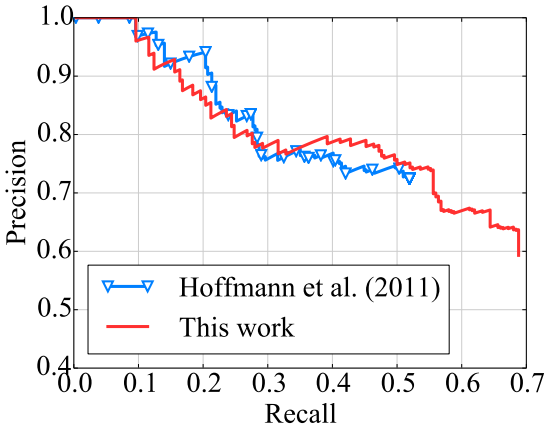


Figure 5: Precision/recall curves for the task of sentential extraction, on the manually labeled dataset of Hoffmann et al. (2011).

only factor in those discrepancies. Indeed, the relation /people/person/place_lived and the relation /people/person/place_of_birth are more frequent than the relation /business/person/company, but the extraction of the later works much better than the extraction of the two first.

Upon examination of the data, this can partly be explained by the fact that almost no sentences extracted for the relation /people/person/place_of_birth in fact express this relation. In other words, many facts present in Freebase are not expressed in the corpus, and are thus impossible to extract. On the other hand, most facts for the relation /people/person/place_lived are missing in Freebase. Therefore, many extractions produced by our system are considered false, but are in fact true positives. The problem of incomplete knowledge base was studied by Min et al. (2013).

Sentential extraction. We finally report precision/recall curves for the task of sentential extraction, in Fig. 5, using the manually labeled dataset of Hoffmann et al. (2011). We observe that for most values of recall, our method achieves similar precision that the one proposed by Hoffmann et al. (2011), while extending the highest recall from 0.52 to 0.68. Thanks to this higher recall, our method achieves a highest F1 score of 0.66, compared to 0.61 obtained by the method proposed by Hoffmann et al. (2011).

Method	Runtime
Mintz et al. (2009)	7 min
Hoffmann et al. (2011)	2 min
Surdeanu et al. (2012)	3 hours
This work	3 hours

Table 1: Comparison of running times for the different methods compared in the experimental section.

8 Conclusion

In this article, we introduced a new formulation for weakly supervised relation extraction. Our method is based on a constrained discriminative formulation of the multiple instance, multiple label learning problem. Using the squared loss, we obtained a convex relaxation of this formulation, allowing us to obtain an approximate solution to the initial integer quadratic program. Thus, our method is not sensitive to initialization. We demonstrated the competitiveness of our approach on the dataset introduced by Riedel et al. (2010), on which our method outperforms the state of the art methods for weakly supervised relation extraction, on both aggregate and sentential extraction.

As noted earlier, another advantage of our method is the fact that it is easily kernelizable. We would like to explore the use of kernels, such as the ones introduced by Zelenko et al. (2003), Culotta and Sorensen (2004) and Bunescu and Mooney (2005), in future work. We believe that such kernels could improve the relatively low recall obtained so far by weakly supervised method for relation extraction.

Acknowledgments

The author is supported by a grant from Inria (Associated-team STATWEB) and would like to thank Armand Joulin for helpful discussions.

References

Francis Bach and Zaïd Harchaoui. 2007. DIFFRAC: a discriminative and flexible framework for clustering. In *Adv. NIPS*.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction for the web. In *IJCAI*.

Amir Beck and Marc Teboulle. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1).

Kedar Bellare and Andrew McCallum. 2007. Learning extractors from unlabeled text using relevant databases. In *Sixth international workshop on information integration on the web*.

Piotr Bojanowski, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. 2013. Finding actors and actions in movies. In *Proceedings of ICCV*.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of HLT-EMNLP*.

Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the ACL*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the ACL*.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1).

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the ACL*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the ACL*.

Armand Joulin, Jean Ponce, and Francis Bach. 2010. Efficient optimization for discriminative latent class models. In *Adv. NIPS*.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of the ACL*.

Scott Miller, Michael Crystal, Heidi Fox, Lance Ramshaw, Richard Schwartz, Rebecca Stone, and Ralph Weischedel. 1998. Algorithms that learn to extract information. In *Proceedings of MUC-7*.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of HLT-NAACL*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the ACL-IJCNLP*.

Yurii Nesterov. 2007. Gradient methods for minimizing composite objective function.

Truc-Vien T Nguyen and Alessandro Moschitti. 2011. End-to-end relation extraction using distant supervision from external semantic repositories. In *Proceedings of the ACL*.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02).

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12).

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the HLT-NAACL*.

Ang Sun, Ralph Grishman, Wei Xu, and Bonan Min. 2011. New york university 2011 system for kbp slot filling. In *Proceedings of the Text Analytics Conference*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP-CoNLL*.

Fei Wu and Daniel S Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*.

Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. 2004. Maximum margin clustering. In *Adv. NIPS*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3.

Appendix A Derivation of the dual

In this section, we derive the dual problem of the quadratic program of section 5. We introduce dual variables $\Lambda \in \mathbb{R}^{I\times(K+1)}$, $\Sigma \in \mathbb{R}^{N\times(K+1)}$, $\Omega \in \mathbb{R}^{I\times(K+1)}$ and $\nu \in \mathbb{R}^N$, such that $\Lambda \geq 0$, $\Sigma \geq 0$ and $\Omega \geq 0$.

The Lagrangian of the problem is

$$\begin{aligned}\frac{1}{2}\text{tr}\left(\mathbf{Y}^\top(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_N)^{-1}\mathbf{Y}\right) + \mu\sum_{i,k}\xi_{ik} \\ - \text{tr}\left(\Lambda^\top((\mathbf{E}\mathbf{Y})\circ\mathbf{S} - \mathbf{R} + \xi)\right) \\ - \text{tr}(\Sigma^\top\mathbf{Y}) - \text{tr}(\Omega^\top\xi) - \nu^\top(\mathbf{Y}\mathbf{1} - \mathbf{1}).\end{aligned}$$

To find the dual function g we minimize the Lagrangian over $\mathbf{Y}$ and ξ. Minimizing over ξ, we find that the dual function is equal to $-\infty$ unless $\mu - \Lambda_{ik} - \Omega_{ik} = 0$, in which case, we are left with

$$\begin{aligned}\frac{1}{2}\text{tr}\left(\mathbf{Y}^\top(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_N)^{-1}\mathbf{Y}\right) \\ - \text{tr}((\Lambda\circ\mathbf{S})^\top\mathbf{E}\mathbf{Y}) - \text{tr}(\Sigma^\top\mathbf{Y}) - \text{tr}(\mathbf{1}\nu^\top\mathbf{Y}) \\ + \text{tr}(\Lambda^\top\mathbf{R}) + \nu^\top\mathbf{1}.\end{aligned}$$

Minimizing over $\mathbf{Y}$, we then obtain

$$\mathbf{Y} = (\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_N)(\mathbf{E}^\top(\mathbf{S}\circ\Lambda) + \Sigma + \nu\mathbf{1}^\top).$$

Replacing $\mathbf{Y}$ by its optimal value, we then obtain the dual function

$$-\frac{1}{2}\text{tr}\left(\mathbf{Z}^\top\mathbf{Q}\mathbf{Z}\right) + \text{tr}\left(\Lambda^\top\mathbf{R}\right) + \nu^\top\mathbf{1}.$$

where

$$\begin{aligned}\mathbf{Q} &= (\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_N), \\ \mathbf{Z} &= \mathbf{E}^\top(\mathbf{S}\circ\Lambda) + \Sigma + \nu\mathbf{1}^\top.\end{aligned}$$

Thus, the dual problem is

$$\begin{aligned}\max_{\Lambda,\Sigma,\nu} \quad & -\frac{1}{2}\text{tr}\left(\mathbf{Z}^\top\mathbf{Q}\mathbf{Z}\right) + \text{tr}\left(\Lambda^\top\mathbf{R}\right) + \nu^\top 1 \\ \text{s.t.} \quad & 0 \leq \Lambda_{ik}, \quad 0 \leq \Sigma_{nk}, \quad 0 \leq \Omega_{ik}, \\ & \mu - \Lambda_{ik} - \Omega_{ik} = 0.\end{aligned}$$

We can then eliminate the dual variable Ω, since the constraints $\Omega_{ik} = \mu - \Lambda_{ik}$ and $\Omega_{ik} \geq 0$ are equivalent to $\mu \geq \Lambda_{ik}$. We finally obtain

$$\begin{aligned}\max_{\Lambda,\Sigma,\nu} \quad & -\frac{1}{2}\text{tr}\left(\mathbf{Z}^\top\mathbf{Q}\mathbf{Z}\right) + \text{tr}\left(\Lambda^\top\mathbf{R}\right) + \nu^\top 1 \\ \text{s.t.} \quad & 0 \leq \Lambda_{ik} \leq \mu, \quad 0 \leq \Sigma_{nk}.\end{aligned}$$

Appendix B Optimization details

Gradient of the dual cost function. The gradient of the dual cost function f with respect to the dual variables Σ, Λ and ν is equal to

$$\begin{aligned}\nabla_\Sigma f &= (\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_N)\mathbf{Z}, \\ \nabla_\Lambda f &= \left((\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_N)\mathbf{Z}\mathbf{E}^\top\right)\circ\mathbf{S} - \mathbf{R}, \\ \nabla_\nu f &= (\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_N)\mathbf{Z}\mathbf{1} - \mathbf{1}.\end{aligned}$$

The most expensive step to compute those gradients is to compute the matrix product $\mathbf{X}\mathbf{X}^\top\mathbf{Z}$. Since the matrix $\mathbf{X}$ is sparse, we efficiently compute this product by first computing the product $\mathbf{X}^\top\mathbf{Z}$, and then by left multiplying the result by $\mathbf{X}$. The complexity of these two operations is $O(NFK)$, where F is the average number of features per relation mention candidate.

Projecting Σ and Λ. The componentwise projection operators associated to the constraints on Σ and Λ are defined by:

$$\begin{aligned}\text{proj}_\Sigma(\Sigma_{nk}) &= \max(0, \Sigma_{nk}), \\ \text{proj}_\Lambda(\Lambda_{ik}) &= \max(0, \min(\mu, \Lambda_{ik})).\end{aligned}$$

The complexity of projecting Σ and Λ is $O(NK)$. Thus, the cost of those operations is ne gligible compared to the cost of computing the gradients of the dual cost function.

Knowledge Graph and Text Jointly Embedding

Zhen Wang[†§] **, Jianwen Zhang**[†] **, Jianlin Feng**[§] **, Zheng Chen**[†]
[†]{v-zw,jiazhan,zhengc}@microsoft.com
[§]{wangzh56@mail2,fengjlin@mail}.sysu.edu.cn
[†]Microsoft Research [§]Sun Yat-sen University

Abstract

We examine the embedding approach to reason new relational facts from a large-scale knowledge graph and a text corpus. We propose a novel method of jointly embedding entities and words into the same continuous vector space. The embedding process attempts to preserve the relations between entities in the knowledge graph and the concurrences of words in the text corpus. Entity names and Wikipedia anchors are utilized to align the embeddings of entities and words in the same space. Large scale experiments on Freebase and a Wikipedia/NY Times corpus show that jointly embedding brings promising improvement in the accuracy of predicting facts, compared to separately embedding knowledge graphs and text. Particularly, jointly embedding enables the prediction of facts containing entities out of the knowledge graph, which cannot be handled by previous embedding methods. At the same time, concerning the quality of the word embeddings, experiments on the analogical reasoning task show that jointly embedding is comparable to or slightly better than word2vec (Skip-Gram).

1 Introduction

Knowledge graphs such as Freebase (Bollacker et al., 2008) and WordNet (Miller, 1995) have become important resources for many AI & NLP applications such as Q & A. Generally, a knowledge graph is a collection of relational facts that are often represented in the form of a triplet (head entity, relation, tail entity), e.g., "(Obama, Born-in, Honolulu)". An urgent issue for knowledge graphs is the coverage, e.g., even the largest knowledge graph of Freebase is still far from complete.

Recently, targeting knowledge graph completion, a promising paradigm of embedding was proposed, which is able to reason new facts only from the knowledge graph (Bordes et al., 2011; Bordes et al., 2013; Socher et al., 2013; Wang et al., 2014). Generally, in this series of methods, each entity is represented as a k-dimensional vector and each relation is characterized by an operation in $\Re^k$ so that a candidate fact can be asserted by simple vector operations. The embeddings are usually learnt by minimizing a global loss function of all the entities and relations in the knowledge graph. Thus, the vector of an entity may encode global information from the entire graph, and hence scoring a candidate fact by designed vector operations plays a similar role to long range "reasoning" in the graph. However, since this requires the vectors of both entities to score a candidate fact, this type of methods can only complete missing facts for which both entities exist in the knowledge graph. However, a missing fact often contains entities out of the knowledge graph (called *out-of-kb* for short in this paper), e.g., one or both entities are phrases appearing in web text but not included in the knowledge graph yet. How to deal with these facts is a significant obstacle to widely applying the embedding paradigm.

In addition to knowledge embedding, another interesting approach is the word embedding method word2vec (Mikolov et al., 2013b), which shows that learning word embeddings from an unlabeled text corpus can make the vectors connecting the pairs of words of some certain relation almost parallel, e.g., vec("China") $-$ vec("Beijing") $\approx$ vec("Japan") $-$ vec("Tokyo"). However, it does not know the exact relation between the pairs. Thus, it cannot be directly applied to complete knowledge graphs.

The capabilities and limitations of knowledge embedding and word embedding have inspired us to design a mechanism to mosaic the knowledge

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1591–1601,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

graph and the "word graph" together in a vector space so that we can score any candidate relational facts between entities and words[1]. Therefore, we propose a novel method to jointly embed entities and words into the same vector space. In our solution, we define a coherent probabilistic model for both knowledge and text, which is composed of three components: the knowledge model, text model, and alignment model. Both the knowledge model and text model use the same core translation assumption for the fact modeling: a candidate fact (h, r, t) is scored based on $\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$. The only difference is, in the knowledge model the relation r is explicitly supervised and the goal is to fit the fact triplets, while in the text model we assume any pair of words h and t that concur in some text windows are of certain relation r but r is a hidden variable, and the goal is to fit the concurring pairs of words. The alignment model guarantees the embeddings of entities and words/phrases lie in the same space and impels the two models to enhance each other. Two mechanisms of alignment are introduced in this paper: utilizing names of entities and utilizing Wikipedia anchors. This way of jointly embedding knowledge and text can be considered to be semi-supervised knowledge embedding: the knowledge graph provides explicit supervision of facts while the text corpus provides much more "relation-unlabeled" pairs of words.

We conduct extensive large scale experiments on Freebase and Wikipedia corpus, which show jointly embedding brings promising improvements to the accuracy of predicting facts, compared to separately embedding the knowledge graph and the text corpus, respectively. Particularly, jointly embedding enables the prediction of a candidate fact with out-of-kb entities, which can not be handled by any existing embedding methods. We also use embeddings to provide a prior score to help fact extraction on the benchmark data set of Freebase+NYTimes and also observe very promising improvements. Meanwhile, concerning the quality of word embeddings, experiments on the analogical reasoning task show that jointly embedding is comparable to or slightly better than word2vec (Skip-Gram).

[1]We do not distinguish between "words" and "phrases", i.e., "words" means "words/phrases".

2 Related Work

Knowledge Embedding. A knowledge graph is embedded into a low-dimensional continuous vector space while certain properties of it are preserved (Bordes et al., 2011; Bordes et al., 2013; Socher et al., 2013; Chang et al., 2013; Wang et al., 2014). Generally, each entity is represented as a point in that space while each relation is interpreted as an operation over entity embeddings. For instance, TransE (Bordes et al., 2013) interprets a relation as a *translation* from the head entity to the tail entity. The embedding representations are usually learnt by minimizing a global loss function involving all entities and relations so that each entity embedding encodes both local and global connectivity patterns of the original graph. Thus, we can reason new facts from learnt embeddings.

Word Embedding. Generally, word embeddings are learned from a given text corpus without supervision by predicting the context of each word or predicting the current word given its context (Bengio et al., 2003; Collobert et al., 2011; Mikolov et al., 2013a; Mikolov et al., 2013b). Although relations between words are not explicitly modeled, continuous bag-of-words (CBOW) and Skip-gram (Mikolov et al., 2013a; Mikolov et al., 2013b) learn word embeddings capturing many syntactic and semantic relations between words where a relation is also represented as the *translation* between word embeddings.

Relational Facts Extraction. Another pivotal channel for knowledge graph completion is extracting relational facts from external sources such as free text (Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012; Zhang et al., 2013; Fan et al., 2014). This series of methods focuses on identifying local text patterns that express a certain relation and making predictions based on them. However, they have not fully utilized the evidences from a knowledge graph, e.g., knowledge embedding is able to reason new facts without any external sources. Actually, knowledge embedding is very complementary to traditional extraction methods, which was first confirmed by (Weston et al., 2013). To estimate the plausibility of a candidate fact, they added scores from embeddings to scores from an extractor, which showed significant improvement. However, as pointed out in the introduction, their knowledge embedding method cannot predict facts involving out-of-kb entities.

3 Jointly Embedding Knowledge and Text

We will first describe the notation used in this paper. A knowledge graph Δ is a set of triplets in the form $(h, r, t), h, t \in \mathcal{E}$ and $r \in \mathcal{R}$ where $\mathcal{E}$ is the entity vocabulary and $\mathcal{R}$ is a collection of predefined relations. We use bold letters $\mathbf{h}, \mathbf{r}, \mathbf{t}$ to denote the corresponding embedding representations of h, r, t. A text corpus is a sequence of words drawn from the word vocabulary $\mathcal{V}$. Note that we perform some preprocessing to detect phrases in the text and the vocabulary here already includes the phrases. For simplicity's sake, without special explanation, when we say "word(s)", it means "word(s)/phrase(s)". Since we consider triplets involving not only entities but also words, we denote $\mathcal{I} = \mathcal{E} \cup \mathcal{V}$. Additionally, we denote anchors by $\mathcal{A}$.

3.1 Modeling

Our model is composed of three components: the knowledge model, text model, and alignment model.

Before defining the component models, we first define the element model for a fact triplet. Inspired by TransE, we also represent a relation r as a vector $\mathbf{r} \in \Re^k$ and score a fact triplet (h, r, t) by $z(\mathbf{h}, \mathbf{r}, \mathbf{t}) = b - \frac{1}{2}\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|^2$ where b is a constant for bias designated for adjusting the scale for better numerical stability and $b = 7$ is a sensible choice. $z(\mathbf{h}, \mathbf{r}, \mathbf{t})$ is expected to be large if the triplet is true. Based on the same element model of fact, we define the component models as follows.

3.1.1 Knowledge Model

We define the following conditional probability of a fact (h, r, t) in a knowledge graph:

$$\Pr(h|r,t) = \frac{\exp\{z(\mathbf{h}, \mathbf{r}, \mathbf{t})\}}{\sum_{\tilde{h} \in \mathcal{I}} \exp\{z(\tilde{\mathbf{h}}, \mathbf{r}, \mathbf{t})\}} \quad (1)$$

and we have named our model *pTransE* (Probabilistic TransE) to show respect to TransE. We also define $\Pr(r|h,t)$ and $\Pr(t|h,r)$ in the same way by choosing corresponding normalization terms respectively. We define the likelihood of observing a fact triplet as:

$$\begin{aligned}\mathcal{L}_f(h,r,t) = \log \Pr(h|r,t) + \log \Pr(t|h,r) \\ + \log \Pr(r|h,t)\end{aligned} \quad (2)$$

The goal of the knowledge model is to maximize the conditional likelihoods of existing fact triplets in the knowledge graph:

$$\mathcal{L}_K = \sum_{(h,r,t) \in \Delta} \mathcal{L}_f(h,r,t) \quad (3)$$

3.1.2 Text Model

We propose the following key assumption for modeling text, which connects word embedding and knowledge embedding: there are relations between words although we do not know what they are.

Relational Concurrence Assumption. *If two words w and v concur in some context, e.g., a window of text, then there is a relation r_{wv} between the two words. That is, we can state the triplet of (w, r_{wv}, v) is a fact.*

We define the conditional probability $\Pr(w|r_{wv}, v)$ following the same formulation of Eq.(1) to model why two words concur in some context. In contrast to knowledge embedding, here r_{wv} is a hidden variable rather than explicitly supervised.

The challenge is to deal with the hidden variable r_{wv}. Obviously, without any more assumptions, the number of distinct r_{wv} is around $|\mathcal{V}| \times \bar{N}$, where $\bar{N}$ is the average number of unique words concurred with each word. This number is extremely large. Thus it is almost impossible to estimate a vector for each r_{wv}. And the problem is actually ill-posed. We need to constrain the freedom degree of r_{wv}. Here we use auxiliary variables to reduce the size of variables we need to estimate: let $\mathbf{w}' = \mathbf{w} + \mathbf{r}_{wv}$, then

$$z(\mathbf{w}, \mathbf{r}_{wv}, \mathbf{v}) \triangleq z(\mathbf{w}', \mathbf{v}) = b - \frac{1}{2}\|\mathbf{w}' - \mathbf{v}\|^2 \quad (4)$$

and

$$\Pr(w|r_{wv}, v) \triangleq \Pr(w|v) = \frac{\exp\{z(\mathbf{w}', \mathbf{v})\}}{\sum_{\tilde{w} \in \mathcal{V}} \exp\{z(\tilde{\mathbf{w}}', \mathbf{v})\}} \quad (5)$$

In this way we need to estimate vectors $\mathbf{w}$ and $\mathbf{w}'$ for each word w, and a total of $2 \times |\mathcal{V}|$ vectors.

The goal of the text model is to maximize the likelihood of the concurrences of pairs of words in text windows:

$$\mathcal{L}_T = \sum_{(w,v) \in \mathcal{C}} n_{wv} \log \Pr(w|v). \quad (6)$$

In the above equation, $\mathcal{C}$ is all the distinct pairs of words concurring in text windows of a fixed size. And n_{wv} is the number of concurrences of the pair (w, v). Interestingly, as explained in Sec.(3.3), this text model is almost equivalent to Skip-Gram.

3.1.3 Alignment Model

If we only have the knowledge model and text model, the entity embeddings and word embeddings will be in different spaces and any computing between them is meaningless. Thus we need mechanisms to align the two spaces into the same one. We propose two mechanisms in this paper: utilizing Wikipedia anchors, and utilizing names of entities.

Alignment by Wikipedia Anchors. This model is based on the connection between Wikipedia and Freebase: for most Wikipedia (English) pages, there is an unique corresponding entity in Freebase. As a result, for most of the anchors in Wikipedia, each of which refers to a Wikipedia page, we know that the surface phrase v of an anchor actually refers to the Freebase entity e_v. Thus, we define a likelihood for this part of anchors as Eq.(6) but replace the word pair (w, v) with the word-entity pair (w, e_v), i.e., using the corresponding entity e_v rather than the surface word v in Eq.(5):

$$\mathcal{L}_{AA} = \sum_{(w,v)\in\mathcal{C}, v\in\mathcal{A}} \log \Pr(w|e_v) \tag{7}$$

where $\mathcal{A}$ denotes the set of anchors.

In addition to Wikipedia anchors, we can also use an entity linking system with satisfactory performance to produce the pseudo anchors.

Alignment by Names of Entities. Another way is to use the names of entities. For a fact triplet $(h, r, t) \in \Delta$, if h has a name w_h and $w_h \in \mathcal{V}$, then we will generate a new triplet of (w_h, r, t) and add it to the graph. Similarly, we also add (h, r, w_t) and (w_h, r, w_t) into the graph if the names exist and belong to the word vocabulary. We call this sub-graph containing names the *name graph* and define a likelihood for the name graph by observing its triplets:

$$\mathcal{L}_{AN} = \sum_{(h,r,t)\in\Delta} \mathbf{I}_{[w_h\in\mathcal{V}\wedge w_t\in\mathcal{V}]}\cdot\mathcal{L}_f(w_h, r, w_t)+ \mathbf{I}_{[w_h\in\mathcal{V}]}\cdot\mathcal{L}_f(w_h, r, t) + \mathbf{I}_{[w_t\in\mathcal{V}]}\cdot\mathcal{L}_f(h, r, w_t) \tag{8}$$

Both alignment models have advantages and disadvantages. Alignment by names of entities is straightforward and does not rely on additional data sources. The number of triplets generated by the names is also large and can significantly change the results. However, this model is risky. On the one hand, the name of an entity is ambiguous because different entities sometimes have the same name so that the name graph may contaminate the knowledge embedding. On the other hand, an entity often has several different aliases when mentioned in the text but we do not have the complete set, which will break the semantic balance of word embedding. For example, for the entity Apple Inc., suppose we only have the standard name "Apple Inc." but do not have the alias "apple". And for the entity Apple that is fruit, suppose we have the name "apple" included in the name graph. Then the vector of the word "apple" will be biased to the concept of fruit rather than the company. But if no name graph intervenes, the unsupervised word embedding is able to learn a vector that is closer to the concept of the company due to the polarities. Alignment by anchors relies on the additional data source of Wikipedia anchors. Moreover, the number of matched Wikipedia anchors (∼40M) is relatively small compared to the total number of word pairs (∼2.0B in Wikipedia) and hence the contribution is limited. However, the advantage is that the quality of the data is very high and there are no ambiguity/completeness issues.

Considering the above three component models together, the likelihood we maximize is:

$$\mathcal{L} = \mathcal{L}_K + \mathcal{L}_T + \mathcal{L}_A \tag{9}$$

where $\mathcal{L}_A$ could be $\mathcal{L}_{AA}$ or $\mathcal{L}_{AN}$ or $\mathcal{L}_{AA} + \mathcal{L}_{AN}$.

3.2 Training

3.2.1 Approximation to the Normalizers

It is difficult to directly compute the normalizers in $\Pr(h|r, t)$ (or $\Pr(t|h, r)$, $\Pr(r|h, t)$) and $\Pr(w|v)$ as the normalizers sum over $|\mathcal{I}|$ or $|\mathcal{V}|$ terms where both $|\mathcal{I}|$ and $|\mathcal{V}|$ reach tens of millions. To prevent having to exactly calculate the normalizers, we use negative sampling (NEG) (Mikolov et al., 2013b) to transform the original objective, i.e., Eq.(9) to a simple objective of the binary classification problem—differentiating the observed data from noise.

First, we define: (i) the probability of a given triplet (h, r, t) to be true ($D = 1$); and (ii) the probability of a given word pair (w, v) to co-occur ($D = 1$):

$$\Pr(D = 1|h, r, t) = \sigma(z(\mathbf{h}, \mathbf{r}, \mathbf{t})) \tag{10}$$

$$\Pr(D = 1|w, v) = \sigma(z(\mathbf{w}', \mathbf{v})) \tag{11}$$

where $\sigma(x) = \frac{1}{1+\exp\{-x\}}$ and $D \in \{0, 1\}$.

Instead of maximizing $\log \Pr(h|r,t)$ in Eq.(2), we maximize:

$$\log \Pr(1|h,r,t) + \sum_{i=1}^{c} \mathbb{E}_{\tilde{h}_i \sim \Pr_{\text{neg}}(\tilde{h}_i)} [\Pr(0|\tilde{h}_i,r,t)] \quad (12)$$

where c is the number of negative examples to be discriminated for each positive example. NEG guarantees that maximizing Eq.(12) can approximately maximize $\log \Pr(h|r,t)$. Thus, we also replace $\log \Pr(r|h,t)$, $\log \Pr(t|r,h)$ in Eq.(2), and $\log \Pr(w|v)$ in Eq.(6) in the same way by choosing corresponding negative distributions respectively. As a result, the objectives of both the knowledge model $\mathcal{L}_K$ (Eq.(3)) and text model $\mathcal{L}_T$ (Eq.(6)) are free from cumbersome normalizers.

3.2.2 Optimization

We use stochastic gradient descent (SGD) to maximize the simplified objectives.

Knowledge model. Δ is randomly traversed multiple times. When a positive example $(h,r,t) \in \Delta$ is considered, to maximize (12), we construct c negative triplets by sampling elements from an uniform distribution over $\mathcal{I}$ and replacing the head of (h,r,t). The transformed objective of $\log \Pr(r|h,t)$ is maximized in the same manner, but by sampling from a uniform distribution over $\mathcal{R}$ and corrupting the relation of (h,r,t). After a mini-batch, computed gradients are used to update the involved embeddings.

Text model. The text corpus is traversed one or more times. When current word v and a context word w are considered, c words are sampled from the unigram distribution raised to the 3/4rd power and regarded as negative examples $(\tilde{w}, v)$ that are never concurrent. Then we compute and update the related gradients.

Alignment model. $\mathcal{L}_{AA}$ and $\mathcal{L}_{AN}$ are absorbed by the text model and knowledge model respectively, since anchors are considered to predict context given an entity and the name graph are homogeneous to the original knowledge graph.

Joint. All three component objectives are *simultaneously* optimized. To deal with large-scale data, we implement a multi-thread version with shared memory. Each thread is in charge of a portion of the data (either knowledge or text corpus), and traverses through them, calculates gradients and commits the update to the global model and is stored in a block of shared memory. For the sake of efficiency, no lock is used on the shared memory.

Table 1: **Data**: triplets used in our experiments.

#$\mathcal{R}$	#$\mathcal{E}$	#Triplet (Train/Valid/Test)		
4,490	43,793,608	123,062,855	40,528,963	40,528,963

3.3 Connections to Related Models

TransE. (Bordes et al., 2013) proposed to model a relation r as a translation vector $\mathbf{r} \in \Re^k$ which is expected to connect $\mathbf{h}$ and $\mathbf{t}$ with low error if $(h,r,t) \in \Delta$. We also follow it. However, TransE uses a margin based ranking loss $\{\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|^2+\gamma-\|\tilde{\mathbf{h}}+\mathbf{r}-\tilde{\mathbf{t}}\|^2\}_+$. It is not a probabilistic model and hence it needs to restrict the norm of either entity embedding and/or relation embedding. Bordes et al. (2013) intuitively addresses this problem by simply normalizing the entity embeddings to the unit sphere before computing gradients at each iteration. We define pTransE as a probabilistic model, which doesn't need additional constraints on the norms of embeddings of entities/words/relations, and thus eliminates the normalization operations.

Skip-gram. (Mikolov et al., 2013a; Mikolov et al., 2013b) defines the probability of the concurrence of two words in a window as:

$$\Pr(w|v) = \frac{\exp\{\mathbf{w}'^{\mathrm{T}}\mathbf{v}\}}{\sum_{\tilde{w}\in\mathcal{V}} \exp\{\tilde{\mathbf{w}}'^{\mathrm{T}}\mathbf{v}\}} \quad (13)$$

which is based on the inner product, while our text model (Eqs. (4), (5)) is based on distance. If we constrain $\|\mathbf{w}\| = 1$ for each w, then $\mathbf{w}'^{\mathrm{T}}\mathbf{v} = 1 - \frac{1}{2}\|\mathbf{w}' - \mathbf{v}\|^2$. It is easy to see that our text model is equivalent to Skip-gram in this case. Our distance-based text model is directly derived from the triplet fact model, which clearly explains why it is able to make the pairs of entities of a certain relation parallel in the vector space.

4 Experiments

We empirically evaluate and compare related models with regards to three tasks: triplet classification (Socher et al., 2013), improving relation extraction (Weston et al., 2013), and the analogical reasoning task (Mikolov et al., 2013a). The related models include: for knowledge embedding alone, TransE (Bordes et al., 2013), pTransE (proposed in this paper); for word embedding alone, Skip-gram (Mikolov et al., 2013b); for both

Table 2: **Data**: the number of $e-e$, $w-e$, $e-w$, $w-w$ triplets/analogies where w represents the out-of-kb entity, which is regarded as word and replaced by its corresponding entity name.

Type	#Triplet (Valid/Test)		#Analogy
$e-e$	12,305,200	12,305,200	71,441
$w-e$	3,655,164	3,654,404	70,878
$e-w$	3,643,914	3,642,978	70,442
$w-w$	460,762	451,381	40,980

knowledge and text, we use "respectively" to refer to the embeddings learnt by TransE/pTransE and Skip-gram, respectively, "jointly" to refer to our jointly embedding method, in which "anchor" and "name" refer to "Alignment by Wikipedia Anchors" and "Alignment by Names of Entities", respectively.

4.1 Data

To learn the embedding representations of entities and words, we use a knowledge graph, a text corpus, and some connections between them.

Knowledge. We adopt Freebase as our knowledge graph. First, we remove the user profiles, version control, and meta data, leaving 52,124,755 entities, 4,490 relations, and 204,120,782 triplets. We call this graph *main facts*. Then we held out 8,331,147 entities from main facts and regard them as out-of-kb entities. Under such a setting, from main facts, we held out all the triplets involving out-of-kb entities, as well as 24,610,400 triplets that don't contain out-of-kb entities. Held-out triplets are used for validation and testing; the remaining triplets are used for training. See Table 1 for the statistics.

We regard out-of-kb entities as words/phrases and thus divide the held-out triplets into four types: no out-of-kb entity ($e-e$), the head is out-of-kb entity but the tail is not ($w-e$), the tail is out-of-kb entity but the head is not ($e-w$), and both the head and tail are out-of-kb entities ($w-w$). Then we replace the out-of-kb entities among the held-out triplets by their corresponding entity names. The mapping from a Freebase entity identifier to its name is done through the Freebase predicate—"/type/object/name". Since some entity names are not present in our vocabulary $\mathcal{V}$, we remove triplets involving these names (see Table 2). In such a way, besides the missing edges between existing entities, the related models can be evaluated on triplets involving words/phrases as their head and/or tail.

Table 3: **Triplet Classification**: comparison between TransE and pTransE over $e-e$ triplets.

Method	Accuracy (%)	Area under PR curve
TransE	93.1	0.86
pTransE	**93.4**	**0.97**

Text. We adopt the Wikipedia (English) corpus. After removing pages designated for navigation, disambiguation, or discussion purposes, there are 3,469,024 articles left. We apply sentence segmentation, tokenization, Part-of-Speech (POS) tagging, and named entity recognition (NER) to these articles using Apache OpenNLP package[2]. Then we conduct some simple chunking to acquire phrases: if several consecutive tokens are identically tagged as "Location"/"Person"/"Organization", or covered by an anchor, we combine them as a chunk. After the preprocessing, our text corpus contain 73,675,188 sentences consisting of 1,522,291,723 chunks. Among them, there are around 20 millions distinct chunks, including words and phrases. We filter out punctuation and rare words/phrases that occur less than three times in the text corpus, reducing $|\mathcal{V}|$ to 5,240,003.

Alignment. One of our alignment models needs Wikipedia anchors. There are around 45 million such anchors in our text corpus and 41,970,548 of them refer to entities in $\mathcal{E}$. Another mechanism utilizes the name graph constructed through names of entities. Specifically, for each training triplet (h,r,t), suppose h and t have entity names w_h and w_t, respectively and $w_h, w_t \in \mathcal{V}$, the training triplet contributes (w_h,r,w_t), (w_h,r,t), and (h,r,w_t) to the name graph. There are 81,753,310 triplets in our name graphs. Note that there is no overlapping between the name graph and held-out triplets of $e-w$, $w-e$, and $w-w$ types.

4.2 Triplet Classification

This task judges whether a triplet (h,r,t) is true or false, i.e., binary classification of a triplet.

Evaluation protocol. Following the same protocol in NTN (Socher et al., 2013), for each true triplet, we construct a false triplet for it by randomly sampling an element from $\mathcal{I}$ to corrupt its head or tail. Since $|\mathcal{E}|$ is significantly larger than $|\mathcal{V}|$ in our data, sampling from a uniform distri-

[2]https://opennlp.apache.org

Table 4: **Triplet classification**: accuracy (%) over various types of triplets.

Type	$e-e$	$w-e$	$e-w$	$w-w$	all
respectively	93.4	52.1	51.4	71.0	77.5
jointly (anchor)	94.4	67.0	66.7	79.8	81.9
jointly (name)	94.5	80.5	80.0	89.0	87.7
jointly (anchor+name)	**95.0**	**82.0**	**81.5**	**90.0**	**88.8**

bution over $\mathcal{I}$ will let triplets involving no word dominate the false triplets. To avoid that, when we corrupt the head of (h, r, t), if $h \in \mathcal{E}$, h' is sampled from $\mathcal{E}$ while if $h \in \mathcal{V}$, h' is sampled from $\mathcal{V}$. The same rule is applied when we corrupt the tail of (h, r, t). In this way, for each of the four types of triplets, we ensure the number of true triplets is equal to that of false ones.

To classify a triplet (h, r, t), we first use the considered methods to score it. TransE scores it by $-|\mathbf{h} + \mathbf{r} - \mathbf{t}|$. Our models score it by $\Pr(D = 1|h, r, t)$ (see Eq.(10)). Then the considered methods label a triplet (h, r, t) as true if its score is larger than the relation-specific threshold of r, as false otherwise. The relation-specific thresholds are chosen to maximize the classification accuracy over the validation set.

We report the classification accuracy. Additionally, we rank all the testing triplets by their scores in descending order. Then we draw a precision-recall (PR) curve based on this ranking and report the area under the PR curve.

Implementation. We implement TransE (Bordes et al., 2013), Skip-gram (Mikolov et al., 2013a), and our models.

First, we train TransE and pTransE over our training triplets with embedding dimension k in $\{50, 100, 150\}$. Adhering to (Bordes et al., 2013), we use the fixed learning rate α in $\{0.005, 0.01, 0.05\}$ for TransE during its 300 epochs. For pTransE, we use the number of negative examples per positive example c among $\{5, 10\}$, the learning rate α among $\{0.01, 0.025\}$ where α decreases along with its 40 epochs. The optimal configurations of TransE are: $k = 100$, $\alpha = 0.01$. The optimal configurations of pTransE are: $k = 100$, $c = 10$, and $\alpha = 0.025$.

Then we train Skip-gram with the embedding dimension k in $\{50, 100, 150\}$, the max skip-range s in $\{5, 10\}$, the number of negative examples per positive example c in $\{5, 10\}$, and learning rate $\alpha = 0.025$ linearly decreasing along with the 6 epochs over our text corpus. Popular words whose frequencies are larger than 10^{-5} are subsampled according to the trick proposed in (Mikolov et al., 2013b). The optimal configurations of Skip-gram are: $k = 150$, $s = 5$, and $c = 10$.

Combining entity embeddings and word embeddings learnt by pTransE and Skip-gram respectively, "respectively" model can score all types of held-out triplets. For our jointly embedding model, we consider various alignment mechanisms and use equal numbers of threads for knowledge model and text model. The best configurations of "jointly" model are: $k = 150$, $s = 5$, $c = 10$, and $\alpha = 0.025$ which linearly decreases along with the 6 epochs of traversing text corpus.

Results. We first illustrate the comparison between TransE and pTransE over $e - e$ type triplets in Table 3. Observing the scores assigned to true triplets by TransE, we notice that triplets of popular relations generally have larger scores than those of rare relations. In contrast, pTransE, as a probabilistic model, assigns comparable scores to true triplets of both popular and rare relations. When we use a threshold to separate true triplets from false triplets of the same relation, there is no obvious difference between the two models. However, when all triplets are ranked together, assigning scores in a more uniform scale is definitely an advantage. Thus, the contradiction stems from the different training strategies of the two models and the consideration of relation-specific thresholds.

Classification accuracies over various types of held-out triplets are presented in Table 4. The "jointly" model outperforms the "respectively" model no matter which alignment mechanism(s) are used. Actually, for the "respectively" model, there is no interaction between entity embeddings and word embeddings during training and thus its predictions, over triplets that involve both entity and word at the same time, are not much better than random guessing. It is also a natural result that alignment by names is more effective than alignment by anchors. The number of anchors is much smaller than the number of overall chunks in our text corpus. In addition, the number of entities mentioned by anchors is very limited com-

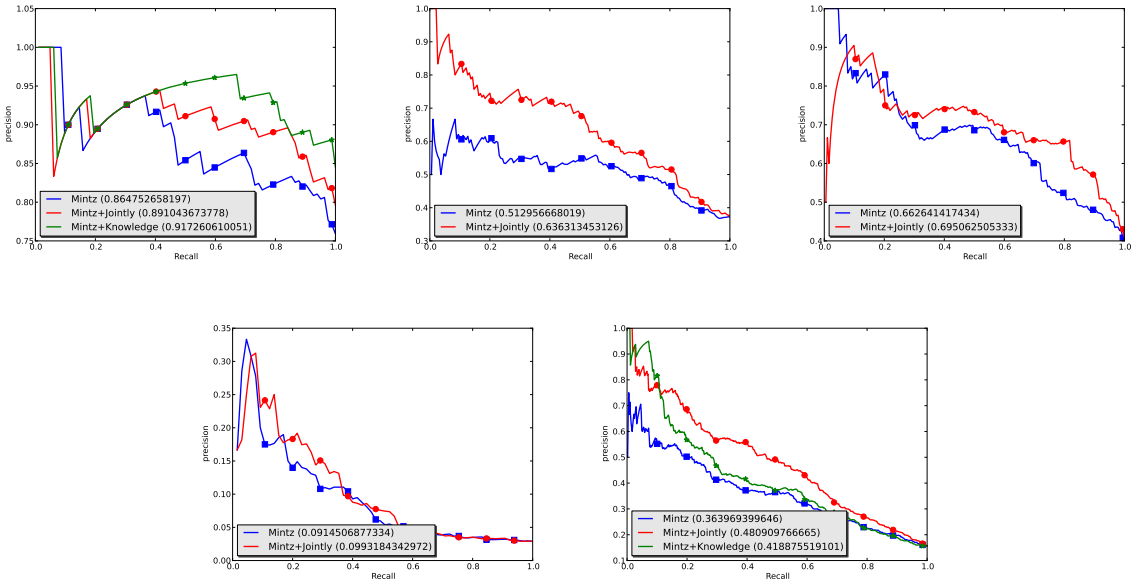


Figure 1: **Improving Relation Extraction**: PR curves of Mintz alone or combined with knowledge (pTransE) / jointly model over (a) $e-e$, (b) $w-e$, (c) $e-w$, (d) $w-w$, and (e) all triplets.

pared with $|\mathcal{E}|$. Thus, interactions brought in by anchors are not as significant as that of the name graph.

4.3 Improving Relation Extraction

It has been shown that embedding models are very complementary to extractors (Weston et al., 2013). However, some entities detected from text are out-of-kb entities. In such a case, triplets involving these entities cannot be handled by any existing knowledge embedding method, but our jointly embedding model can score them. As our model can cover more candidate triplets provided by extractors, it is expected to provide more significant improvements to extractors than any other embedding model. We confirm this point as follow.

Evaluation protocol. For relation extraction, we use a public dataset—NYT+FB (Riedel et al., 2010)[3], which distantly labels the NYT corpus by Freebase facts. We consider (Mintz et al., 2009) and Sm2r (Weston et al., 2013) as our extractors to provide candidate triplets as well as their plausibilities estimated according to text features.

For embedding, we first held out triplets from our training set that appear in the test set of NYT+FB. Then we train TransE, pTransE and the "jointly" model on the remaining training triplets as well as on our text corpus. Then we use these models to score each candidate triplet in the same way as the previous triplet classification experiment.

For combination, we first divide each candidate triplet into one of these categories: $e-e$, $e-w$, $w-e$, $w-w$, and "out-of-vocabulary". Because there is no embedding model that can score triplets involving out-of-vocabulary word/phrase, we just ignore these triplets.Please note that, for our jointly embedding model, there are no "out-of-vocabulary" triplets if we include the NYT corpus for training. We use the embedding models to score candidate triplets and combine the scores given by the embedding model with scores given by the extractors. For each type $e-e$, $e-w$, $w-e$, $w-w$ and their union (i.e. *all*), we rank the candidate triplets by their revisited scores and draw PR curve to observe which embedding method provides the most significant improvements to the extractors.

Implementation. For (Mintz et al., 2009), we use the implementation in (Surdeanu et al., 2012)[4]. We implement Sm2r by ourselves with the best hyperparameters introduced in (Weston et al., 2013). For TransE, pTransE, and the "jointly" model, we use the same implementations, scoring schemes, and optimal configurations as the triplet classification experiment.

To combine extractors with embedding mod-

[3] http://iesl.cs.umass.edu/riedel/ecml/

[4] http://nlp.stanford.edu/software/mimlre.shtml

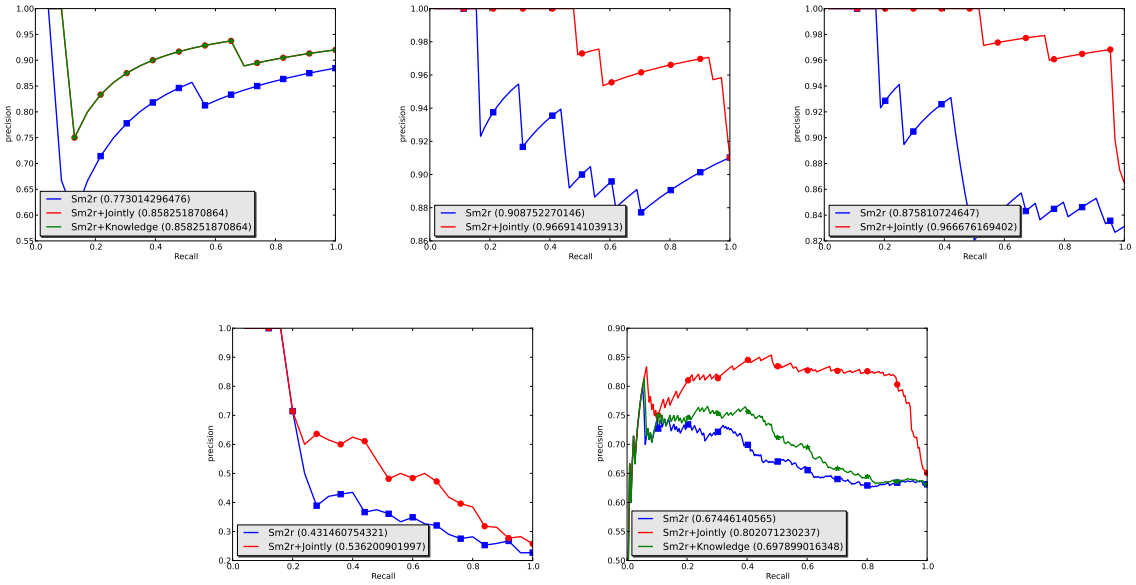


Figure 2: **Improving Relation Extraction**: PR curves of Sm2r alone or combined with knowledge (TransE) / jointly model over (a) $e - e$, (b) $w - e$, (c) $e - w$, (d) $w - w$, and (e) all triplets.

els, we consider two schemes. Since Mintz scores candidate triplets in a probabilistic manner, we linearly combine its scores with the scores given by pTransE or the "jointly" model: $\beta \Pr_{\text{Mintz}} + (1-\beta) \Pr_{\text{pTransE/Jointly}}$ where β is enumerated from 0 to 1 with 0.025 as a search step. On the other hand, neither Sm2r nor TransE is a probabilistic model. Thus, we combine Sm2r with TransE or the "jointly" model according to the scheme proposed in (Weston et al., 2013) where for each candidate (h, r, t), if $\sum_{r' \neq r} \delta(\text{Score}(h, r, t) < \text{Score}(h, r', t))$ is less than τ, we increase $\text{Score}_{Sm2r}(h, r, t)$ by p. We search for the best β, τ, and p on another dataset—Wikipedia corpus distantly labeled by Freebase.

Result. We present the PR curves in Fig. (1, 2). Over candidate triplets provided by either Mintz or Sm2r, the "jointly" model is consistently comparable with the "knowledge" model (TransE/pTransE) over $e - e$ triplets while it outperforms the "knowledge" model by a considerable margin over triplets of other types. These results confirm the advantage of jointly embedding and are actually straightforward results of our triplet classification experiment because the only difference is that the triplets here are provided by the extractor.

Table 6: **Phrases Analogical Reasoning Task**.

Method	Accuracy (%)	Hits@10 (%)
Skip-gram	18.0	56.1
Jointly (anchor)	**27.6**	**65.0**
Jointly (name)	11.3	40.6
Jointly (anchor+name)	18.3	54.0

Table 7: **Constructed Analogical Reasoning Task**.

Method	Accuracy (%)	Hits@10 (%)
Skip-gram	10.5	14.1
Jointly (anchor)	10.5	14.3
Jointly (name)	11.5	16.2
Jointly (anchor+name)	**11.6**	**16.5**

4.4 Analogical Reasoning Task

We compare our method with Skip-gram on this task to observe and study the influences of both knowledge embedding and alignment mechanisms on the quality of word embeddings.

Evaluation protocol. We use the same public datasets as in (Mikolov et al., 2013b): 19,544 word analogies[5]; 3,218 phrase analogies[6]. We also construct analogies from our held-out triplets (see Table 2) by first concatenating two entity pairs of the same relation to form an analogy and

[5] code.google.com/p/word2vec/source/browse/trunk/questions-words.txt

[6] code.google.com/p/word2vec/source/browse/trunk/questions-phrases.txt

Table 5: **Words Analogical Reasoning Task**.

Method	Accuracy (%)			Hits@10 (%)		
	Semantic	Syntactic	Total	Semantic	Syntactic	Total
Skip-gram	71.4	69.0	70.0	90.4	89.3	89.8
Jointly (anchor)	75.3	68.3	**71.2**	91.5	88.9	**89.9**
Jointly (name)	54.5	54.2	59.0	75.8	86.5	82.1
Jointly (anchor+name)	56.5	65.7	61.9	78.1	87.6	83.6

then replacing the entities by corresponding entity names, e.g., "(Obama, Honolulu, David Beckham, London)" where the relation is "Born-in".

Following (Mikolov et al., 2013b), we only consider analogies that consist of the top-K most frequent words/phrases in the vocabulary. For each analogy denoted by (h_1, t_1, h_2, t_2), we enumerate all the top-K most frequent words/phrases w except for h_1, t_1, h_2, and calculate the distance (Cosine/Euclidean according to specific model) between $\mathbf{h}_2 + (\mathbf{t}_1 - \mathbf{h}_1)$ and $\mathbf{w}$. Ordering all these words/phrases by their distances in ascending order, we obtain the rank of the correct answer t_2. Finally, we report *Hits@10* (i.e., the proportion of correct answers whose ranks are not larger than 10) and *accuracy* (i.e., Hits@1). For word analogies and constructed analogies, we set $K = 200,000$; while for phrase analogies, we set $K = 1,000,000$ to recall sufficient analogies.

Implementation. For Skip-gram and the "Jointly" (anchor/name/anchor+name) model, we use the same implementations and optimal configurations as the triplet classification experiment.

Results. Jointly embedding using Wikipedia anchors for alignment consistently outperforms Skip-gram (Table 5, 6, 7) showing that the influence of knowledge embedding, injected into word embedding through Wikipedia anchors, is beneficial. The vector of an ambiguous word is often a mixture of its several meanings but, in a specific context, the word is disambiguated and refers to a specific meaning. Using global word embedding to predict words within a specific context may pollute the embeddings of surrounding words. Alignment by anchors enables entity embeddings to alleviate the propagation of ambiguities and thus improves the quality of word embeddings.

Using entity names for alignment hurts the performance of analogies of words and phrases (Table 5, 6). The main reason is that these analogies are popular facts frequently mentioned in text while a name graph forces word embeddings to satisfy both popular and rare facts. Another reason stems from the versatility of mentioning an entity. Consider "(Japan, yen, Europe, euro)" for example. Knowledge embedding is supposed to give significant help to completing this analogy as "/location/country/currency"$\in \mathcal{R}$. However, the entity of Japanese currency is named "Japanese yen" rather than "yen" and thus the explicit translation learnt from knowledge embedding is not directly imposed on the word embedding of "yen". In contrast, using entity names for alignment improves the performances on constructed analogies (Table 7). Since there is a relation $r \in \mathcal{R}$ for each constructed analogy $(w_{h_1}, w_{t_1}, w_{h_2}, w_{t_2})$, although neither (w_{h_1}, r, w_{t_1}) nor (w_{h_2}, r, w_{t_2}) is present in the name graph, other facts involving these words act on the vectors of these words, in the same manner of traditional knowledge embedding.

Overall, any high-quality entity linking system can be used to further improve the performance.

5 Conclusions

In this paper, we introduced a novel method of jointly embedding knowledge graphs and a text corpus so that entities and words/phrases are represented in the same vector space. In such a way, our method can perform prediction on any candidate facts between entities/words/phrases, going beyond previous knowledge embedding methods, which can only predict facts whose entities exist in knowledge graph. Extensive, large-scale experiments show that the proposed method is very effective at reasoning new facts. In addition, we also provides insights into word embedding, especially on the capability of analogical reasoning. In this aspect, we empirically observed some hints that jointly embedding also helps word embedding.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250. ACM.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 301–306.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1602–1612, Seattle, Washington, USA, October. Association for Computational Linguistics.

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

Miao Fan, Deli Zhao, Qiang Zhou, Zhiyuan Liu, Thomas Fang Zheng, and Edward Y. Chang. 2014. Distant supervision for relation extraction with matrix completion. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849, Baltimore, Maryland, June. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1366–1371, Seattle, Washington, USA, October. Association for Computational Linguistics.

Xingxing Zhang, Jianwen Zhang, Junyu Zeng, Jun Yan, Zheng Chen, and Zhifang Sui. 2013. Towards accurate distant supervision for relational facts extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 810–815, Sofia, Bulgaria, August. Association for Computational Linguistics.

Abstractive Summarization of Product Reviews Using Discourse Structure

Shima Gerani[‡†] * **Yashar Mehdad**[‡] * **Giuseppe Carenini**[‡] **Raymond T. Ng**[‡] **Bita Nejat**[‡]
[†]University of Lugano
Switzerland
[‡]University of British Columbia
Vancouver, BC, Canada
{gerani,mehdad,carenini,rng,nejatb}@cs.ubc.ca

Abstract

We propose a novel abstractive summarization system for product reviews by taking advantage of their discourse structure. First, we apply a discourse parser to each review and obtain a discourse tree representation for every review. We then modify the discourse trees such that every leaf node only contains the aspect words. Second, we aggregate the aspect discourse trees and generate a graph. We then select a subgraph representing the most important aspects and the rhetorical relations between them using a PageRank algorithm, and transform the selected subgraph into an aspect tree. Finally, we generate a natural language summary by applying a template-based NLG framework. Quantitative and qualitative analysis of the results, based on two user studies, show that our approach significantly outperforms extractive and abstractive baselines.

1 Introduction

Most existing works on sentiment summarization focus on predicting the overall rating on an entity (Pang et al., 2002; Pang and Lee, 2004) or estimating ratings for product features (Lu et al., 2009; Lerman et al., 2009; Snyder and Barzilay, 2007; Titov and McDonald, 2008)). However, the opinion summaries in such systems are extractive, meaning that they generate a summary by concatenating extracts that are representative of opinion on the entity or its aspects.

Comparing extractive and abstractive summaries for evaluative texts has shown that an abstractive approach is more appropriate for summarizing evaluative text (Carenini et al., 2013; Di Fabbrizio et al., 2014). This finding is also supported by a previous study in the context of summarizing news articles (Barzilay et al., 1999). To the best of our knowledge, there are only three previous works on abstractive opinion summarization (Ganesan et al., 2010; Carenini et al., 2013; Di Fabbrizio et al., 2014). The first work (Ganesan et al., 2010) proposes a graph-based method for generating ultra concise opinion summaries that are more suitable for viewing on devices with small screens. This method does not provide a well-formed grammatical abstract and the generated summary only contains words that occur in the original texts. Therefore, this approach is more extractive than abstractive. Another limitation is that the generated summaries do not contain any information about the distribution of opinions.

In the second work, (Carenini et al., 2013) addresses some of the aforementioned problems and generates well-formed grammatical abstracts that describe the distribution of opinion over the entity and its features. However, for each product, this approach requires a feature taxonomy handcrafted by humans as an input, which is not scalable. To partially address this problem (Mukherjee and Joshi, 2013) has proposed a method for the automatic generation of a product attribute hierarchy that leverages ConceptNet (Liu and Singh, 2004). However, the resulting ontology tree has been used only for sentiment classification and not for classification.

In the third and most recent study, (Di Fabbrizio et al., 2014) proposed Starlet-H as a hybrid abstractive/extractive sentiment summarizer. Starlet-H uses extractive summarization techniques to select salient quotes from the input reviews and embeds them into the abstractive summary to exemplify, justify or provide evidence for the aggregate positive or negative opinions. However, Starlet-H assumes a limited number of aspects as input and needs a large amount of training data to learn the

*The contribution of the first two authors to this paper was equal.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1602–1613,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

ordering of aspects for summary generation.

Highlighting the reasons behind opinions in reviews was also previously proposed in (Kim et al., 2013). However, their approach is extractive and similar to (Ganesan et al., 2010) does not cover the distribution of opinions. Furthermore, it aims to explain the opinion on only one aspect, rather than explaining the overall opinion on the product, its aspects and how they affect each other.

To address some of the above mentioned limitations , in this paper we propose a novel abstractive summarization framework that generates an aspect-based abstract from multiple reviews of a product. In our framework, anything that is evaluated in the review is considered an aspect, including the product itself. We propose a natural language generation (NLG) framework that takes aspects and their structured relation as input and generates an abstractive summary. However, unlike (Carenini et al., 2013), our method assumes no domain knowledge about the entity in terms of a user-defined feature taxonomy. On the other hand, in contrast with Starlet-H, we do not limit the input reviews to a small number of aspects and our aspect ordering method takes advantage of rhetorical information and does not require any training data. Our method relies on the discourse structure and discourse relations of reviews to infer the *importance of aspects* as well as the *association* between them (e.g., which aspects relate to each other).

Researchers have recently started using the discourse structure of text in sentiment analysis and have shown its advantage in improving sentiment classification accuracy (e.g., (Lazaridou et al., 2013; Trivedi and Eisenstein, 2013; Somasundaran et al., 2009; Asher et al., 2008)). However, to the best of our knowledge, none of the existing works have looked into exploiting discourse structure in abstractive review summarization.

In our work, *importance of aspects*, derived from the reviews' discourse structure and relations, is used to rank and select aspects to be included in the summary. More specifically, we start with the most important (highest ranked) aspects to generate a summary and add more aspects to the system until a summary of desired length is obtained. *Aspect association* is considered to better explain how the opinions on aspects affect each other (e.g., opinion over specific aspects affect the opinion over the more general ones). Consider the following sentence as an example summary generated by our system for the entity **Camera Canon G3**: "*All reviewers who commented on the camera, thought that it was really good mainly because of the photo quality.*" This summary encapsulates all the following key pieces of information: 1) *camera* and *photo quality* are the most important aspects, 2) People have positive opinion on *camera* in general and on *photo quality* as one of its features, and finally 3) *photo quality* is the main reason behind users satisfaction on *camera*. Such summary helps users understand the reason behind a rating of a product or its aspects without going through all reviews or reading scattered opinions on different aspects in multiple sentences of an extractive summary.

This paper makes the following contributions:
1. We propose a novel content selection and structuring strategy for review summarization, that assumes no prior domain knowledge, by taking advantage of the discourse structure of reviews.
2. We propose a novel product-independent template-based NLG framework to generate an abstract based on the selected content, without relying on deep syntactic knowledge or sophisticated NLG methods. Our framework, similarly to (Carenini et al., 2013), can effectively convey the distribution of opinions.
3. We present the first study that investigates the use of discourse structure information in both content selection and abstract generation for multi-document summarization.

Quantitative and qualitative analysis over evaluation results of two user studies on a set of user reviews on twelve different products show that our system is an effective abstractive system for review summarization.

2 Summarization Framework

At a high-level, our summarization framework involves generating a summary from multiple input reviews based on an Aspect Hierarchy Tree (AHT) that reflects the importance of aspects as well as the relationships between them. In our framework, an AHT is generated automatically from the set of input reviews, where each sentence of every review is marked by the aspects presented in that sentence and the polarity of opinions over them. There are various methods for extracting the aspects and predicting the polarity of opinion (Hu and Liu, 2004b; Hu and Liu,

2006; Kim et al., 2011). In this paper we do not focus on aspect extraction and sentiment prediction but rather consider the aspect and their polarity/strength (P/S) information given as input to the system. P/S scores are integer values in the range [-3, +3], where +3 is the most positive and -3 is the most negative polarity value. We also do not attempt to automatically resolve coreferences between aspects. For example, the aspect *"g3"*, *"canon g3"* and *"canon"* were manually collapsed as into *"camera"*. This preprocessing step helps to reduce the noise generated by inaccurate aspect labeling in our reviews. Figure 1 shows two sample input reviews where the aspects and their P/S scores are identified. For example, in R1, aspects *camera*, *photo quality* and *auto mode* are mentioned. The P/S values for the three aspects are [+2], [+3] and [+2] respectively which indicate positive opinion on all aspects.

The first component of our system applies a discourse parser to each review and obtains a discourse tree representation for every review (e.g. Figure 1 (a) and (b)). The discourse trees are then modified such that every leaf node only contains the aspect words. The output of the first component is an aspect-based discourse tree (ADT) for every review (e.g. Figure 1 (c) and (d)). In the second component, we aggregate the ADTs and generate a graph called Aggregated Rhetorical Relation Graph (ARRG) (e.g. Figure 1 (f)). The third component of our framework, is responsible for content selection and structuring. It takes ARRG as input, runs Weighted PageRank, and selects a subgraph (e.g. Figure 1 (g)) representing the most important aspects. Finally it transforms the selected subgraph into a tree and provides an AHT as output (e.g. Figure 1 (h)). The generated AHT is the input of the last component which generates a natural language summary by applying micro planning and sentence realization. We now describe each component of our framework in more detail.

3 Discourse Parsing

Any coherent text is structured so that we can derive and interpret the information. This structure shows how discourse units (text spans such as sentences or clauses) are connected and relate to each other. Discourse analysis aims to reveal this structure. Several theories have been proposed in the past to describe the discourse structure, among which the Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) is one of the most popular. RST divides a text into minimal atomic units, called Elementary Discourse Units (EDUs). It then forms a tree representation of a discourse called a Discourse Tree (DT) using rhetorical relations (e.g., *Elaboration*, *Explanation*, etc) as edges, and EDUs as leaves. EDUs linked by a rhetorical relation are also distinguished based on their relative importance in conveying the author's message: *nucleus* is the central part, whereas *satellite* is the peripheral part.

We use a publicly available state-of-the-art discourse parser (Joty et al., 2013)[1] to generate a DT for each product review. Figure 1 (a) and (b) show DTs for two sample reviews where dotted edges identify the satellite spans. DT1 in Figure 1 (a) shows that review R1 consists of three EDUs with two relations *Elaboration* and *Background* between them. It also shows that the first EDU (i.e. *I love camera*) is the nucleus (shown by solid line) of the relation *Elaboration* and so the rest of the document (EDUs 2 and 3) is less important and aims at elaborating on what the author meant in the first EDU. Similarly, the structure shows that the third EDU is mentioned as background information for EDU2 and so is less important for realizing the core meaning of the document.

After obtaining the DTs, we remove all words from the text spans of each EDU, except the aspect words. Thus, for each review, we have a DT where a leaf node represents the aspects occurring in the corresponding EDU. Note that there may be EDUs containing no aspects in a review. In such cases, we keep the corresponding node and mark it with no aspect. We call the resulting tree an Aspect-based Discourse Tree (ADT) which will be used in the next components. Figure 1 (c) and (d) show ADTs generated from DTs.

4 Aspect Rhetorical Relation Graph (ARRG)

In the second component, we aim at generating an ARRG for a product, based on the ADTs which are the output from the previous component. There are two motivations behind aggregating the ADTs and building the ARRG: *i*) while each ADT can be rather noisy because of the informal language of the reviews and inaccuracies from

[1] http://alt.qcri.org/discourse/Discourse_Parser_Dist.tar.gz

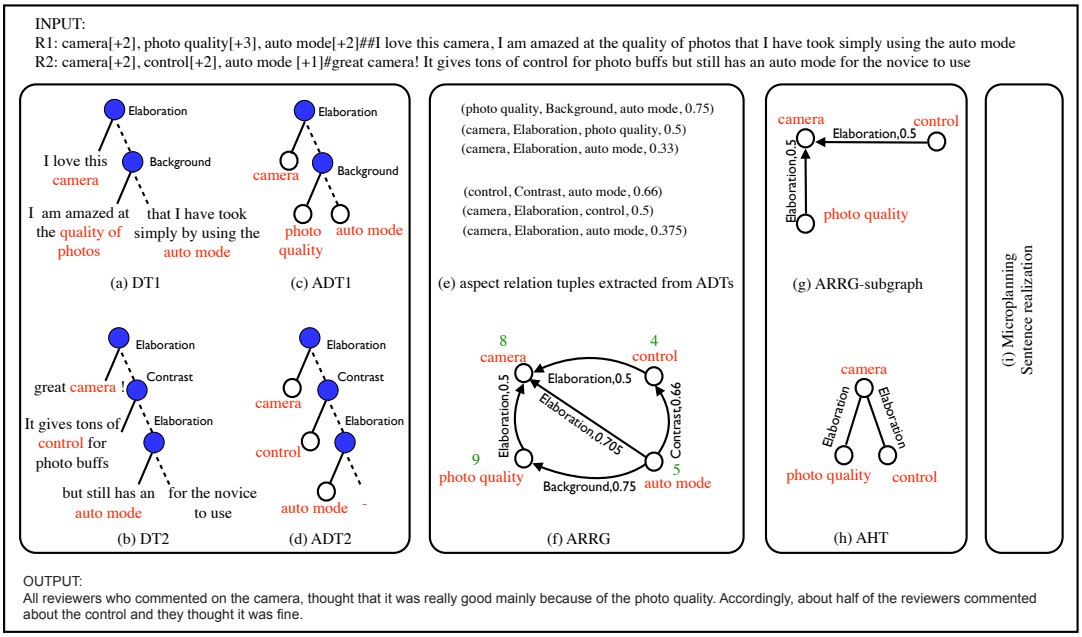


Figure 1: A simple example illustrating different components of our summarization framework.

automatic discourse parsing, aggregating all the ADTs can reveal more reliable information; and *ii*) the aggregated information highlights the most important aspects overall as well as the strongest connection between the aspects. This information can effectively drive the content selection and abstract generation phases.

$ARRG$ is a directed graph in which we allow multiple edges between two vertices. In ARRG, vertices represent aspects. We associate to each aspect/node an importance measure that aggregates all the P/S values that the aspect receives in all the reviews. By following (Carenini et al., 2013), let $PS(a)$ be the set of P/S values that an aspect a receives. The direct measure of importance of the aspect is defined as:

$$dir\text{-}moi(a) = \sum_{ps \in PS(a)} ps^2 \qquad (1)$$

In ARRG, edges indicate existence of a rhetorical relation between text spans of a review in which the aspects occurred. Edges are labeled with the type of the relation as well as a weight indicating our confidence in the presence of the relation between the two aspects. In ARRG, an edge with label r, w from node u to node v, $u \xrightarrow{\text{r, w}} v$, indicates the existance of a relation r with confidence w between two aspects u and v. Also, the direction of the arrow indicates that u and v occurred in the *satellite* and *nucleus* spans respectively. For example, *photo quality* $\xrightarrow{\text{elaboration, 0.8}}$ *camera* indicates that there is a high confidence (0.8) that aspect *photo quality* was used in a text span to elaborate aspect *camera*. Moreover, *camera* is a more important aspect compared to *photo quality*.

To build ARRG, we use all the ADTs that are output of the previous component (one for each review). From each ADT_j, we extract all tuples of the form (u, r, v, w) in which u is an aspect occurring in a satellite span, v is an aspect occurring in a nucleus span, r is a relation type and w is the weight of the tuple computed as follows:

$$w = 1 - 0.5\frac{|\text{EDUs between u and v}|}{|\text{total EDUs in } ADT_j|} - 0.5\frac{d_r}{d} \qquad (2)$$

where, $|.|$ indicates cardinality of a set. d indicates the depth of the ADT_j and d_r indicates the depth of the sub-tree of ADT_j rooted at relation r. Equation 2 weighs a tuple based on two factors: (i) the relative distance of the EDUs in which the two aspects u and v participating in relation r occur. The intuition is that aspects occurring in close proximity to each other are more related; and (ii) the depth of the sub-tree at the point of the relation relative to the depth of the whole ADT_j. This is because as we move from leaves to the root of a DT, the accuracy of the rhetorical structure has been shown to decrease. Also, at higher levels of an ADT (intra-sentential relations), it is more

likely that aspects are related through non adjacent EDUs and so are less strongly related. Figure 1 (e) shows tuples extracted from sample ADTs.

Notice that every two aspects u and v may be related by the same relation more than once in an ADT for a review. Thus, we might have i tuples with the same u,r, and v but confidence weights which are not necessarily the same. From every ADT_j, we extract all (u, r, v, w_{ij}) and select the one with maximum confidence. We then aggregate the selected tuples extracted from different reviews. Putting these two steps together, for every two aspects u and v related by relation r, we obtain a single tuple $(u, r, v, \hat{w})$ where

$$\hat{w} = \sum_j \max_i w_{ij} \quad (3)$$

Figure 1 (f) shows an example ARRG built for the sample reviews.

5 Content Selection and Structuring

The content of the summary is selected by extracting from ARRG a subgraph containing the most important aspects. Such content is then structured by transforming the subgraph into an aspect hierarchy.

5.1 Subgraph Extraction

In ARRG aspects/nodes are weighted by how frequently and strongly they are evaluated in the reviews (i.e, dir-moi) and edges are weighted by how frequently and strongly the corresponding aspects are rhetorically related in the discourse trees (Equation 3). In content selection, we want to extract aspects that not only have high weight, but that are also linked with heavy edges to other heavy aspects. This problem can be effectively addressed by Weighted Page Rank (WPR) (Xing and Ghorbani, 2004). WPR takes the importance of both the in-links and out-links of the aspects into account and distributes rank scores based on the weights of relations between aspects. In this way, the heavier aspect nodes, that are either in the nuclei of many relations or in the satellites of relations with other heavy aspects, are promoted. We then update the weight of nodes (aspects) with the new score from WPR. Finally, we rank nodes based on their updated score moi and select the top *N* aspects.

$$moi(a) = \alpha dir\text{-}moi(a) + (1 - \alpha)WPR(a) \quad (4)$$

Here α is a coefficient that can be tuned on a development set or can be set to 0.5 without tuning. Figure 1 (g) shows an example subgraph selected from the sample ARRG.

5.2 Aspects Subgraph to Aspects Hierarchy Transformation

In this step, we generate a hierarchical tree structure for aspects. Such a tree structure helps to navigate over aspects and can be easily traversed to find certain aspects and their relation to their parent or children. The hierarchy of aspects also matches the intuition that the root node is the most frequent and general aspect (often the product) and as the depth increases, nodes represent more specific aspects of the product with less frequency and weight.

To obtain a hierarchical tree structure from the extracted subgraph, we first build an undirected graph as follows: we merge the edges connecting two nodes and consider the sum of their weights as the weight of the merged graph. We also ignore the relation direction for the purpose of generating the tree. We then find the Maximum Spanning Tree of the undirected subgraph and set the highest weighted aspect as the root of the tree. This process results in a useful knowledge structure of aspects with their associated weight and sentiment polarity connected with the rhetorical relations called Aspect Hierarchical Tree (AHT). Figure 1 (h) shows the generated AHT from the sub-graph.

6 Abstract Generation

The automatic generation of a natural language summary in our system involves the following tasks (Reiter and Dale, 2000): (*i*) microplanning, which covers lexical selection; and (*ii*) sentence realization, which produces english text from the output of the microplanner.

6.1 Microplanning

Once the content is selected and structured, it is passed to the microplanning module which performs lexical choice. Lexical choice is an important component of microplanning. Lexical choice is formulated in our system based on a "*formal*" style, language "*variability*" and "*fluent*" connectivity among other lexical units. Table 1 demonstrates our lexical choice strategy.

Quantifiers:
if (relative-number == 1) : ["All users (x people) who commented about the aspect", "All costumers (x people) that reviewed the aspect", ...]
if (relative-number >= 0.8) : ["Almost all users commented about the aspect and they", "Almost all costumers mentioned the aspect and they", ...]
if (relative-number >= 0.6) : ["Most users commented about the aspect and they mainly", "Most shoppers mentioned aspect and they", ...]
if (relative-number >= 0.45) : ["Almost half of the users commented about the aspect and they", 'Almost 50% of the shoppers mentioned the aspect and they", ...]
if (relative-number >= 0.2) : ["About y% of the reviewers commented about the aspect and they", "Around y% of the shoppers mentioned the aspect and they", ...]
if (relative-number >= 0.0) : ["z reviewers commented about the aspect and in overall they", "z shoppers mentioned about the aspect and they", ...]
Polarity verbs:
if (controversial(aspect)) : ["had controversial opinions about it", "expressed controversial opinions about this feature", ...]
else: if (average <= −2) : ["hated it", "felt that it was very poor', 'thought that it was very poor", ...]
if (average <= −1) : ["disliked it", "felt that it was poor", "thought that it was poor", ...]
if (average < 0) : ["did not like it", "felt that it was weak", "thought that it was weak", ...]
if (average == 0) : ["did not express any strong positive or negative opinion about it", ...]
if (average <= +1) : ["liked it", "felt that it was fine", "thought that it was satisfactory", ...]
if (average <= +2) : ["absolutely liked it", "really liked this feature", "felt that it was a really good feature", "thought that it was really good", ...]
if (average <= +3) : ["loved it", "felt that it was great", "thought that it was great", ...]
Connectives
["Also, related to the aspect", "Accordingly, ", "Moreover, regarding the aspect, " ,"In relation to the aspect, ", "Talking about the aspect, ", ...]

Table 1: Microplanning strategy for lexical choice. The selected lexical items will fill the template in the realization step.

Sentence realization templates:
First sentence templates:
if (polarity-agreement(root,highest-weighted-child) & connecting-relation == [elaboration, explain, cause, summary, same-unit, background, evidence, justify]): *"quantifier + polarity-verb +* 'mainly because of the' *+ highest-weighted-child"*
else: "quantifier + polarity-verb"
First level children (aspects) sentences templates:
"connective + ', ' + quantifier + ' ' + polarity-verb"
Supporting sentences templates:
if (#children(aspect)==1): "connective + quantifier + verb "
elseif (#children(aspect)>1 & polarity-agreement(children)): "connective + quantifier + verb + [and, similarly, while, ...] + quantifier + verb"
elseif (#children(aspect)>1 & !polarity-agreement(children)): "connective + quantifier + verb + [but, in contrast, on contrary, ...] + quantifier + verb"

Table 2: Sentence realization templates.

Quantifiers: for each aspect, a quantifier is selected based on both the absolute and relative number of users whose opinions contributed to the evaluation of the aspect.

Polarity verbs: for each aspect, a polarity verb is selected based on the average sentiment polarity strength for that aspect. Although the average, in most cases, can be a good metric to evaluate the polarity of an aspect, it fails when the distribution of evaluations is centered on zero, for instance, if there are equal numbers of positive and negative evaluations (i.e., controversial). To partially solve this problem, we first check whether the aspect evaluation is controversial by applying the formula proposed by (Carenini and Cheung, 2008). In the case of controversiality, our microplanner selects a lexical item to express the controversiality of the aspect. In other cases, we use the average and select the polarity verb based on that.

Connectives: in order to form more fluent and readable sentences and to increase the language variability, we randomly select our connectives from the list shown in Table 1. Moreover, when a parent aspect (excluding the root in AHT) has two children, they are connected by one of the coordinating conjunction "[and, similarly]" if they agree on polarity, and they will be connected by a choice of "[on the contrary, in contrast]" otherwise (see Supporting sentences templates in Table 2). As an alternative we could have selected connectives based on the discourse relations specified in the aspects tree. However, this is left as future work.

6.2 Sentence Realization

The realization of our abstract generation is performed by applying a rather simple and comprehensive template-based strategy. Depending on the specific lexical choice in microplanning step, an appropriate template and corresponding fillers are selected as shown in Table 2. We develop three different templates: *i)* generates the first abstract sentence; *ii)* generates the abstract sentence for the aspects with no children; and *iii)* generates supporting sentences for aspects with children.

For illustration, assuming that we apply this strategy to a 5-node variation of the AHT in Figure 1 (h), where the aspect "*control*" has two children "*auto mode*" and "*setting*", we obtain "*All reviewers (45 people) who commented on the camera, thought that it was really good mainly because of the photo quality. Accordingly, about 24% of the reviewers commented about the control and they*

thought it was fine. Also, related to the control, 7 users expressed their opinion about the auto mode and they liked it, similarly, 6 shoppers commented about the setting and they thought that it was satisfactory."

7 Experimental Setup

7.1 Dataset and Baselines

We conduct our experiments using the customer reviews of twelve products obtained from (Hu and Liu, 2004a): 4 digital cameras, 1 DVD player, 1 MP3 player, 2 routers, 2 phones, 1 diaper and 1 antivirus. The reviews were collected from Amazon.com and Cnet.com. We use manually annotated aspects and their associated sentiment from the same dataset.

We compare the summaries generated by our system with two state-of-the-art extractive baselines and a simpler version of our abstractive system, as follows:

1) MEAD-LexRank (LR): we use the LexRank (Erkan and Radev, 2004) implementation inside the MEAD summarization framework (Radev et al., 2004), which outperforms other algorithms implemented in the MEAD framework.

2) MEADStar (MEAD*): a state-of-the-art extractive opinion summarization system (Carenini et al., 2013), which is adapted from the open source summarization framework MEAD. MEAD* orders aspects by the number of sentences evaluating that aspect, and selects a sentence from each aspect until it reaches the word limit. The sentence that is selected for each aspect is the one with the highest sum of polarity/strength evaluations for any aspect.

3) Simple Abstractive (SA): we sort the aspects of each product based on $dir\text{-}moi$ (Equation 1). Then, for each aspect, we generate a sentence based on a simple template "*quantifier* + *polarity-verb*" until the summary reaches the word limit.

We limit the length of our summaries to 150 words. In our experiment we use the default parameter in Equation 4 without tuning (i.e. $\alpha = 0.5$). Our system starts the content selection process with 10 aspects and generates a summary based on a AHT with 10 aspects. We add one aspect, reproduce the AHT and regenerate the summary. We repeat this process until the word limit is reached.

7.2 Evaluation Framework

On one hand, the lack of product reviews datasets with human written summaries, and on the other hand, the difficulty of generating human-written summaries for reviews, makes review summary evaluation a very challenging task.

We evaluate the summaries generated by our system by performing two user studies based on pairwise preferences using a popular crowdsourcing service.[2] The user preference evaluation is an effective method for opinion summarization (e.g., (Lerman et al., 2009)). The main motivations behind pairwise preferences evaluation is two-fold: *i*) raters can make a preference decision more efficiently than a scoring judgment; and *ii*) rater agreement is higher in preference decisions than in scoring judgments (Ariely et al., 2003).

In both user studies, for each product, we run six pairwise comparisons for four summaries. In each rating assignment, two summaries of the same product were placed in random order. Raters were shown the name of each product along with the relevant summaries and were asked to express their preference for one summary over the other using a simple set of criteria. For two summaries S_1 and S_2 raters should choose one of the following three options: 1) Prefer S_1, 2) Prefer S_2, 3) No preference.

Raters were specifically instructed that their rating should express "*overall satisfaction with the information provided by the summary*". Raters were also asked to provide a brief comment justifying their choice. Over 48 raters participated in each study, and each comparison was evaluated by at least five raters generating more than 360 judgments for each user study. We pre-select the high skilled raters to ensure a higher quality results.

The main difference between the two user studies is that in "user study 1", we show two summaries to the raters and ask them to choose the one they prefer without showing them the original reviews. In contrast, in "user study 2", we show two summaries with links to the full text of the reviews for the raters to explore. In order to make sure that the raters read the reviews, we ask them to write a short summary of the reviews before rating the automatic summaries. We ran two different user studies because: *i*) for each product there might be many reviews to be included; *ii*) there is no guaranty that raters, in various evaluation settings, read

[2] www.crowdflower.com

System I vs System II	Agreement		No preference		Preferred Sys I		Preferred Sys II	
User Studies	1	2	1	2	1	2	1	2
LR vs MEAD*	0.33	0.75	7%	6%	35%	20%	58%	74%
LR vs SA	0.42	0.83	0%	0%	38%	21%	*62%*	*79%*
LR vs Our System	0.50	1.00	0%	3%	26%	13%	***74%***	***84%***
MEAD* vs SA	0.58	0.83	0%	0%	38%	20%	*62%*	*80%*
MEAD* vs Our System	0.67	0.50	0%	3%	25%	30%	***75%***	***67%***
SA vs Our System	0.42	0.50	12%	11%	23%	32%	**65%**	**57%**

Table 3: Results of pairwise preference user studies. Statistically significant improvements ($p < 0.01$) over the baselines are demonstrated by bold fonts. Italic fonts indicate statistical significance ($p < 0.01$) of abstractive methods (SA and Our System) over extractive approaches (LR and MEAD*).

Systems	LR		MEAD*		SA		Our System	
User Studies	1	2	1	2	1	2	1	2
Preference	33%	18%	41%	41%	49%	63%	**71%**	**69%**

Table 4: System preference results. Statistically significant improvements ($p < 0.01$) over the baselines are demonstrated by bold fonts.

the reviews (partially or completely); and *iii*) there is no evidence regarding the depth that each rater would look into the reviews. Therefore, choosing between user study 1 and 2 is not a straightforward decision. In other words, designing the two user studies in this way helps us to answer the question: "*Does the fact that raters can read all the reviews affect their ratings?*".

8 Results

This section provides a quantitative and qualitative analysis of the evaluation results[3].

8.1 Quantitative Analysis

Quantitative results for both user studies are shown in Table 3. The second column indicates the percentage of judgments for which the raters were in agreement. Agreement here is a weak agreement, where four (out of five) raters are defined to be in agreement if they all gave the same rating. The next three columns indicate the percentage of judgments for each preference category, grouped into two user studies. In addition, we measure the preference for each system in both user studies (Table 4). For each system, the preference is the number of times raters prefer the system, divided by the total number of judgments for that system (e.g., if A is preferred over B 10 out of 30 times, and A is preferred over C 15 out of 20 times, the overall preference of A is (10+15)/(30+20)=50%)

Abstractive vs. Extractive: the results of our system and SA in Table 3 show statistically significant improvements in pairwise preference over extractive baselines (LR and MEAD*) in both user studies.[4] Moreover, the results of overall preference in Table 4 demonstrates that two abstractive systems are preferred over the extractive ones in both studies. This further supports the findings in the previous studies (e.g., (Carenini et al., 2013)) that users prefer abstractive summarization. We can observe that, in both user studies, raters prefer our system over other abstractive and extractive baselines. Also, the highest pairwise preference percentages occur comparing an extractive and an abstractive system (e.g., LR vs Our System).

Abstractive Systems: the raters prefer our system over SA in both user studies (65% and 57%), and our system ranks first in our pairwise preference user studies. Knowing that both systems are abstractive and the differences between them comes from using the rhetorical structure in the content selection and abstract generation phases, proves the effectiveness of using rhetorical structure and relations in abstractive summarization of reviews.

Extractive Systems: the result in Table 3 and 4 demonstrate that raters prefer MEAD* over LR. Although both systems are extractive, the MEAD* system has been proposed for extractive opinion

[3]The evaluation results and summaries obtained from CrowdFlower are publicly available and can be downloaded from: `https://www.cs.ubc.ca/cs-research/lci/research-groups/natural-language-processing/reviews/user_study_results.zip`

[4]The statistical significance tests was calculated by approximate randomization, as described in (Yeh, 2000).

Preference Sys 1 to Sys 2	Reasons	Examples of preference justification taken from the raters comments
Our System to LR and MEAD*	Readability, coverage of aspects, aggregation of opinions	*better wording, more objective, more depth, I like the stats, more detail about people opinion, less personal experience, detail comparison from different reviews, a summary in a summary, mentions more features, ...*
LR and MEAD* to Our System	Descriptiveness, personal point of views, product capabilities	*explain how the product is positive, good characteristics about the product, has lot more to tell, more descriptive about features, personal perspective, not only characteristics but also ability, more true to the product itself, ...*
Our System to SA	The relations between the aspects, more language variability	*provides a bit more information, is very complete, not repetitive, more elegant, coherent, ...*
SA to Our System	Simpler structure, more aspects	*written better, has touched variety of features, ...*

Table 5: System preference results. The reasons are classified based on raters justifications preferring the underlined systems.

summarization. In contrast, LR is a generic extractive summarization system which is not optimized for opinion summarization. This also further demonstrates the need for opinion and reviews summarization systems.

User Study 1 vs. User Study 2: the first interesting observation is that, although the overall ranking of systems in both user studies does not change, there are some changes in the results. This indicates that reading the reviews effects preference decisions. We can observe that in all cases except one (MEAD* vs Our System) the agreement between the raters increases significantly when they are given the reviews. This can be interpreted as reading the reviews helps the rater to choose a better summary easier and more effectively. Moreover, we calculate the overall agreement for both user studies.[5] Case study 2 reports a higher overall agreement (70%) in comparison with the user study 1 (65%). This further proves our finding that showing the reviews can help the raters with their preference judgment.

In Table 3, the preference of sys 2 (last column) significantly rises for all cases when compared with the LR system. This proves that raters strongly prefer the summaries that cover opinionated sentences, specifically when they are exposed to the reviews. The same result is reflected in Table 4, where the overall preference of LR drops when the raters are given the reviews. We also observe a significant rise in preference of sys 2 when MEAD* is compared with SA (Table 3) and in the overall preference of SA (Table 4) in user study 2. This proves that raters become more confident in preferring an abstractive summary over an extractive one when the reviews are given to them. In contrast, we notice that the preference of sys 2 drops comparing "MEAD* vs Our System" and "SA vs Our System". Knowing that the drop is not significant and the the overall ranking of systems remains unchanged, this case is less straight forward to interpret.

[5]The agreement is calculated based on 100 randomly sampled units selected from our crowdsourcing job.

8.2 Qualitative Analysis

We collect and group the rater justifications in the results we obtain by crowdsourcing our evaluation framework, when preferring a summary over another, in Table 5. To make the comparison more clear, Example 1 shows the summaries generated by MEAD* and our system.

Comparing our system with the extractive baselines, raters' justifications are classified in three main categories. Although the language of the extractive summaries is less formal, raters often prefer our system in terms of presentation and language. They justify their selections by expressing phrases such as "*better grammar*" or "*fewer errors*". They also comment about the coverage of aspects in the summaries generated by our system and they realize that our system was capable of aggregating the opinions for each aspect. In contrast, when they prefer the extractive summaries, they like the descriptive language of the summary and the technical details of the products that were missing in our system summaries.

We also notice that raters realize the usage of structure (AHT) in our system (both of content selection and summary generation) and they appreciate it by expressing phrases such as "*very complete*", "*more elegant*" or "*related features*". In contrast, they sometimes appreciate a simpler language in summaries generated by SA. Moreover, few raters prefer the higher coverage in SA summaries. This is mainly because not using connectives and structure in SA leaves more space to include more aspects.

Product: Nikon Coolpix 4300
MEAD*: it is very compact but the controls are so well designed that they 're still easy to use . It 's easy for beginners to use , but has features that more serious photographers will love , so it 's an excellent camera to grow into . But overall this is a good camera with a ' really good ' picture clarity ; an exceptional close-up shooting capability .The battery life is very good , i got about 90 minutes with the lcd turned on all the time , the first time around , and i have been using it with the lcd off every now and then , and have yet needed to recharge it . Yes , the picture quality and features which are too numerous to mention are unmatched for any camera in this price range.
Our System: All reviewers (34 people), who commented on the camera, felt that it was really good mainly because of the picture. Around 26% of the reviewers expressed their opinion about the picture quality and they really liked it. Around 24% of the reviewers noted the use and they thought that it was satisfactory. Talking about the use, around 24% of the reviewers expressed their opinion about the size and they felt that it was fine. Only 6 reviewers commented about the scene mode and in overall they thought that it was satisfactory. Moreover, regarding the scene mode, 4 shoppers mentioned about the manual mode and they thought that it was satisfactory, and similarly only 4 reviewers commented about the auto mode and in overall they did not express any strong positive or negative opinion about it. Only 4 costumers mentioned the software and they felt that it was really good.

Example 1. Summaries generated by our system and MEAD* baseline for the Nikon Coolpix 4300 camera. For brevity we exclude other baselines.

9 Conclusions

We have presented a framework for abstractive summarization of product reviews based on discourse structure. For content selection, we propose a graph model based on the importance and association relations between aspects, that assumes no prior domain knowledge, by taking advantage of the discourse structure of reviews. For abstract generation, we propose a product independent template-based natural language generation (NLG) framework that takes aspects and their structured relation as input and generates an abstractive summary. Quantitative evaluation results, based on two pairwise preference user studies, show substantial improvement over extractive and abstractive baselines, including MEAD*, which is considered a state-of-the-art opinion extractive summarization system, and a simpler version of our abstractive system. In future work, we plan to extend the microplanning phase by taking advantage of the highly weighted rhetorical relations between the aspects and select connective phrases based on the discourse relations specified in the aspects tree. In addition, we plan to develop and evaluate an end-to-end system, in which the aspect extraction and polarity estimation of aspects are automated. In this way, we can achieve an end-to-end automatic summarizaion system for product reviews.

Acknowledgments

This work was supported in part by Swiss National Science Foundation (PBTIP2-145659) and NSERC Business Intelligence Network. We would like to thank the anonymous reviewers for their valuable comments. We also acknowledge Shafiq Rayhan Joty for his help regarding rhetorical parser.

References

Dan Ariely, George Loewenstein, and Drazen Prelec. 2003. "Coherent Arbitrariness": Stable Demand Curves Without Stable Preferences. *Quarterly Journal of Economics*, 118:73–105.

Nicholas Asher, Farah Benamara, and Yvette Yannick Mathieu. 2008. Distilling opinion in discourse: A preliminary study. In *Coling 2008: Companion volume: Posters and Demonstrations*, pages 5–8.

Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 550–557, Stroudsburg, PA, USA. Association for Computational Linguistics.

Giuseppe Carenini and Jackie Chi Kit Cheung. 2008. Extractive vs. nlg-based abstractive summarization of evaluative text: the effect of corpus controversiality. In *INLG '08: Proceedings of the Fifth International Natural Language Generation Conference*, pages 33–41, Morristown, NJ, USA. Association for Computational Linguistics.

Giuseppe Carenini, Jackie Chi Kit Cheung, and Adam Pauls. 2013. Multi-document summarization of evaluative text. *Computational Intelligence*, 29(4):545–576.

Giuseppe Di Fabbrizio, Amanda Stent, and Robert Gaizauskas. 2014. A hybrid approach to multi-document summarization of opinions in reviews. In *Proceedings of the 8th International Natural Language Generation conference*, INLG 2014.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, December.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 340–348, Stroudsburg, PA, USA. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004a. Mining and summarizing customer reviews. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2004 (KDD 2004)*, pages 168–177, Seattle, Washington.

Minqing Hu and Bing Liu. 2004b. Mining opinion features in customer reviews. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-2004)*.

Minqing Hu and Bing Liu. 2006. Opinion feature extraction using class sequential rules. In *Proceedings of AAAI 2006 Spring Sympoia on Computational Approaches to Analyzing Weblogs (AAAI-CAAW 2006)*.

Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Yashar Mehdad. 2013. Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 486–496, Sofia, Bulgaria, August. Association for Computational Linguistics.

Hyun Duk Kim, Kavita Ganesan, Parikshit Sondhi, and ChengXiang Zhai. 2011. Comprehensive review of opinion summarization.

Hyun Duk Kim, Malu Castellanos, Meichun Hsu, ChengXiang Zhai, Umeshwar Dayal, and Riddhiman Ghosh. 2013. Compact explanatory opinion summarization. In *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management*, CIKM '13, pages 1697–1702, New York, NY, USA. ACM.

Angeliki Lazaridou, Ivan Titov, and Caroline Sporleder. 2013. A bayesian model for joint unsupervised induction of sentiment, aspect and discourse representations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1630–1639, Sofia, Bulgaria, August. Association for Computational Linguistics.

Kevin Lerman, Sasha Blair-Goldensohn, and Ryan McDonald. 2009. Sentiment summarization: Evaluating and learning user preferences. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 514–522, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hugo Liu and Push Singh. 2004. Conceptnet: A practical commonsense reasoning toolkit. *BT Technology Journal*, 22(4):211–226.

Yue Lu, ChengXiang Zhai, and Neel Sundaresan. 2009. Rated aspect summarization of short comments. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 131–140, New York, NY, USA. ACM.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Subhabrata Mukherjee and Sachindra Joshi. 2013. Sentiment aggregation using conceptnet ontology. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, IJCNLP 2013.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. 2004. MEAD — A platform for multidocument multilingual text summarization. In *Conference on Language Resources and Evaluation (LREC)*, Lisbon, Portugal.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA.

Benjamin Snyder and Regina Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *HLT-NAACL*, pages 300–307.

Swapna Somasundaran, Galileo Namata, Janyce Wiebe, and Lise Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 170–179, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ivan Titov and Ryan T. McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, ACL 2008, pages 308–316. Association for Computational Linguistics.

Rakshit S. Trivedi and Jacob Eisenstein. 2013. Discourse connectors for latent subjectivity in sentiment analysis. In *HLT-NAACL*, pages 808–813. The Association for Computational Linguistics.

Wenpu Xing and Ali Ghorbani. 2004. Weighted pagerank algorithm. In *Proceedings of the Second Annual Conference on Communication Networks and Services Research*, CNSR '04, pages 305–314. IEEE Computer Society.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics - Volume 2*, COLING '00, pages 947–953, Stroudsburg, PA, USA. Association for Computational Linguistics.

Clustering Aspect-related Phrases by Leveraging Sentiment Distribution Consistency

Li Zhao, Minlie Huang, Haiqiang Chen*, Junjun Cheng*, Xiaoyan Zhu
State Key Laboratory of Intelligent Technology and Systems
National Laboratory for Information Science and Technology
Dept. of Computer Science and Technology, Tsinghua University, Beijing, PR China
*China Information Technology Security Evaluation Center
zhaoli19881113@126.com aihuang@tsinghua.edu.cn

Abstract

Clustering aspect-related phrases in terms of product's property is a precursor process to aspect-level sentiment analysis which is a central task in sentiment analysis. Most of existing methods for addressing this problem are context-based models which assume that domain synonymous phrases share similar co-occurrence contexts. In this paper, we explore a novel idea, *sentiment distribution consistency*, which states that different phrases (e.g. "price", "money", "worth", and "cost") of the same aspect tend to have consistent sentiment distribution. Through formalizing *sentiment distribution consistency* as soft constraint, we propose a novel unsupervised model in the framework of Posterior Regularization (PR) to cluster aspect-related phrases. Experiments demonstrate that our approach outperforms baselines remarkably.

1 Introduction

Aspect-level sentiment analysis has become a central task in sentiment analysis because it can aggregate various opinions according to a product's properties, and provide much detailed, complete, and in-depth summaries of a large number of reviews. Aspect finding and clustering, a precursor process of aspect-level sentiment analysis, has attracted more and more attentions (Mukherjee and Liu, 2012; Chen et al., 2013; Zhai et al., 2011a; Zhai et al., 2010).

Aspect finding and clustering has never been a trivial task. People often use different words or phrases to refer to the same product property (also called *product aspect or feature* in the literature). Some terms are lexically dissimilar while semantically close, which makes the task more challenging. For example, "price", "money" , "worth" and "cost" all refer to the aspect "price" in reviews. In order to present aspect-specific summaries of opinions, we first of all, have to cluster different aspect-related phrases. It is expensive and time-consuming to manually group hundreds of aspect-related phrases. In this paper, we assume that the aspect phrases have been extracted in advance and we keep focused on clustering domain synonymous aspect-related phrases.

Existing studies addressing this problem are mainly based on the assumption that different phrases of the same aspect should have similar co-occurrence contexts. In addition to the traditional assumption, we develop a new angle to address the problem, which is based on *sentiment distribution consistency* assumption that different phrases of the same aspect should have consistent sentiment distribution, which will be detailed soon later.

Pros:	***LCD***, *nice* ***touch screen***, *longer* **battery life**
Cons:	*Horrible* ***picture quality***
Review:	The ***touch screen*** was the selling feature for me. The ***LCD touch screen*** is nice and large. This camera also has very impressive ***battery life***. However the ***picture quality*** is very grainy.

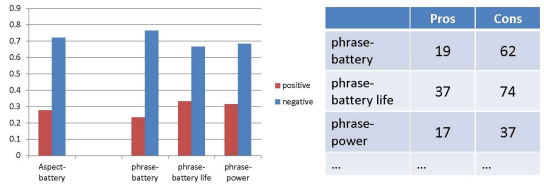
Figure 1: A semi-structured Review.

This new angle is inspired by this simple observation (as illustrated in Fig. 1): two phrases within the same cluster are not likely to be simultaneously placed in Pros and Cons of the same review. A straightforward way to use this information is to formulate cannot-link knowledge in clustering algorithms (Chen et al., 2013; Zhai et al., 2011b). However, we have a particularly different manner to leverage the knowledge.

Due to the availability of large-scale semi-structured customer reviews (as exemplified in Fig. 1) that are supported by many web sites, we can easily get the estimation of sentiment distribution for each aspect phrase by simply counting how many times a phrase appears in *Pros* and

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1614–1623,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

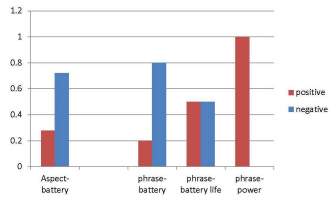
Cons respectively. As illustrated in Fig. 2, we can see that the estimated sentiment distribution of a phrase is close to that of its aspect. The above observation suggests the *sentiment distribution consistency* assumption: **different phrases of the same aspect tend to have the same sentiment distribution, or to have statistically close distributions**. This assumption is also verified by our data: for most (above 91.3%) phrase with relatively reliable estimation (whose occurrence $\geq$50), the KL-divergence between the sentiment distribution of a phrase and that of its corresponding aspect is less than 0.05.



	Pros	Cons
phrase-battery	19	62
phrase-battery life	37	74
phrase-power	17	37
...	...	...

Figure 2: The sentiment distribution of aspect "battery" and its related-phrases on *nokia 5130* with a large amount of reviews.

It is worth noting that, the sentiment distribution of a phrase can be estimated accurately only when we obtain a sufficient number of reviews. When the number of reviews is limited, however, the estimated sentiment distribution for each phrase is unreliable (as shown in Fig. 3). A key issue, arisen here, is how to formulate this assumption in a statistically robust manner. The proposed model should be robust when only a limited number of reviews are available.



	Pros	Cons
phrase-battery	1	4
phrase-battery life	2	2
phrase-power	1	0
...	...	...

Figure 3: The sentiment distribution of aspect "battery" and its related-phrases on *nokia 3110c* with a small mumber of reviews.

To deal with this issue, we model *sentiment distribution consistency* as soft constraint, integrated into a probabilistic model that maximizes the data likelihood. We design the constraint to work in the following way: when we have sufficient observations, the constraint becomes tighter, which plays a more important role in the learning process; when we have limited observations, the constraint becomes very loose so that it will have less effect on the model.

In this paper, we propose a novel unsupervised model, Sentiment Distribution Consistency Regularized Multinomial Naive Bayes (SDC-MNB). The context part is modeled by Multinomial Naive Bayes in which aspect is treated as latent variable, and *Sentiment distribution consistency* is encoded as soft constraint within the framework of Posterior Regularization (PR) (Graca et al., 2008). The main contributions of this paper are summarized as follows:

- We study the problem of clustering phrases by integrating both context information and sentiment distribution of aspect-related phrases.
- We explore a novel concept, *sentiment distribution consistency(SDC)*, and model it as soft constraint to guide the clustering process.
- Experiments show that our model outperforms the state-of-art approaches for aspect clustering.

The rest of this paper is organized as follows. We introduce the SDC-MNB model in Section 2. We present experiment results in Section 3. In Section 4, we survey related work. We summarize the work in Section 5.

2 Sentiment Distribution Consistency Regularized Multinomial Naive Bayes

In this section, we firstly introduce our assumption *sentiment distribution consistency* formally and show how to model the above assumption as soft constraint , which we term SDC-constraint. Secondly, we show how to combine SDC-constraint with the probabilistic context model. Finally, we present the details for context and sentiment extraction.

2.1 Sentiment Distribution Consistency

We define *aspect* as a set of phrases that refer to the same property of a product and each phrase is termed *aspect-related phrase* (or *aspect phrase* in short). For example, the aspect "battery" contains aspect phrases such as "battery", "battery life", "power", and so on.

F	the aspect phrase set
f_j	the j^{th} aspect phrase
y_j	the aspect for aspect phrase f_j
A	the aspect set
a_i	the i^{th} aspect
D	the set of context documents
d_j	the context document of f_j
V	the word vocabulary
w_t	the t^{th} word in vocabulary V
$w_{d_j,k}$	the k^{th} word in d_j
N_{tj}	the number of times word w_t occurs in d_j
P	the product set
p_k	the k^{th} product
u_{ik}	the *sentiment distribution parameter* of aspect a_i on p_k
$\hat{s}_{jk}$	the estimated *sentiment distribution parameter* of phrase f_j on p_k
n_{jk}	the occurrence times of aspect phrase f_j on p_k
$\hat{\sigma}_{jk}$	the sample standard deviation
θ	the model parameters
$p_\theta(a_i\|d_j)$	the posterior distribution of a_i given d_j
$q(y_j = a_i)$	the projected posterior distribution of a_i given d_j

Table 1: Notations

Let us consider the sentiment distribution on a certain aspect a_i. In a large review dataset, aspect a_i could receive many comments from different reviewers. For each comment, we assume that people either praise or complain about the aspect. So each comment on the aspect can be seen as a Bernoulli trial, where the aspect receives positive comments with probability p_{a_i}[1]. We introduce a *random variable* X_{a_i} to denote the sentiment on aspect a_i, where $X_{a_i} = 1$ means that aspect a_i receives positive comments, $X_{a_i} = 0$ means that aspect a_i receives negative comments. Obviously, the sentiment on aspect a_i follows the Bernoulli distribution,

$$Pr(X_{a_i}) = p_{a_i}^{X_{a_i}} * (1 - p_{a_i})^{1 - X_{a_i}}, X_{a_i} \in \{0, 1\}. \quad (1)$$

Or in short,

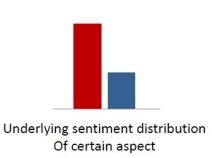
$$X_{a_i} \sim Bernoulli(p_{a_i})$$

Let us see the case for aspect phrase f_j, where $f_j \in$ aspect a_i. Similarly, each comment on an aspect phrase f_j can also be seen as a Bernoulli trial. We introduce a *random variable* X_{f_j} to denote the sentiment on aspect phrase f_j, where $X_{f_j} = 1$ means that aspect f_j receives positive comments, $X_{f_j} = 0$ means that aspect f_j receives negative comments. As just discussed, we assume that each aspect phrase follows the same distribution with the corresponding aspect. This leads to the following formal description:

- ***Sentiment Distribution Consistency*** : The sentiment distribution of *aspect phrase* is the same as that of the corresponding *aspect*. Formally, for all aspect phrase $f_j \in$ aspect a_i, $X_{f_j} \sim Bernoulli(p_{a_i})$.

2.2 Sentiment Distribution Consistency Constraint

Assuming the sentiment distribution of aspect a_i is given in advance, we need to judge whether an aspect phrase f_j belongs to the aspect a_i with limited observations for f_j. Let's consider the example in Fig. 4. For aspect phrase 3, we have no definite answer due to the limited number of observations. For aspect phrase 1, it seems that the sentiment distribution is consistent with that of the left aspect. However, we can not say that the phrase belongs to the aspect because the distribution may be the same for two different aspects. For aspect phrase 2, we are confident that its sentiment distribution is different from that of the left aspect, given sufficient observations.



	pros	cons	Belong to left aspect?
Aspect phrase1	21	10	Reliable estimation, Use to judge. =>maybe
Aspect phrase2	3	100	Reliable estimation, Use to judge. => not likely
Aspect phrase3	1	2	Unreliable estimation =>maybe

Figure 4: Sentiment distribution of an aspect, and observations on aspect phrases.

To be concise, *we judge an aspect phrase doesn't belong to certain aspect only when we are confident that they follow different sentiment distributions*.

Inspired by the intuition, we conduct interval parameter estimation for parameter p_{f_j}(sentiment distribution for phrase f_j) with limited observations, and thus get a confidence interval for p_{f_j}. If p_{a_i}(sentiment distribution for aspect a_i) is not in the confidence interval of p_{f_j}, we then are confident that they follow different distributions. In other words, if aspect phrase $f_j \in$ aspect a_i, we are confident that p_{a_i} is in the confidence interval of p_{f_j}.

More formally, we use u_{ik} to denote the sentiment distribution parameter of aspect a_i on product p_k, and assume that u_{ik} is given in advance.

[1] *positive comment* means that an aspect term is observed in *Pros* of a review.

We want to know whether the sentiment distribution on aspect phrase f_j is the same as that of aspect a_i on product p_k given a limited number of observations (samples). It's straightforward to calculate the confidence interval for parameter s_{jk} in the Bernoulli distribution function. Let the sample mean of n_{jk} samples be $\hat{s}_{jk}$, and the sample standard deviation be $\hat{\sigma}_{jk}$. Since the sample size is small here, we use the Student-t distribution to calculate the confidence interval. According to our assumption, we are confident that u_{ik} is in the confidence interval if $f_j \in a_i$.

$$\hat{s}_{jk} - C\frac{\hat{\sigma}_{jk}}{\sqrt{n_{jk}}} \leq u_{ik} \leq \hat{s}_{jk} + C\frac{\hat{\sigma}_{jk}}{\sqrt{n_{jk}}}, \forall f_j \in a_i, \forall k. \quad (2)$$

where we look for t-table to find C corresponding to a certain confidence level(such as 95%) with the freedom of $n_{jk} - 1$. For simplicity, we represent the above confidence interval by $[\hat{s}_{jk} - d_{jk}, \hat{s}_{jk} + d_{jk}]$, where $d_{jk} = C\frac{\hat{\sigma}_{jk}}{\sqrt{n_{jk}}}$.

We introduce an indicator variable z_{ij} to represent whether the aspect phrase f_j belongs to aspect a_i, as follows:

$$z_{ji} = \begin{cases} 1 & ;\text{if } f_j \in a_i \\ 0 & ;\text{otherwise} \end{cases} \quad (3)$$

This leads to our SDC-constraint function.

$$\phi = z_{ji}|u_{ik} - \hat{s}_{jk}| \leq d_{jk}, \forall i, j, k \quad (4)$$

SDC-constraint are flexible for modeling Sentiment Distribution Consistency. The more observations we have, the smaller d_{jk} is. For frequent aspect phrase, the constraint can be very informative because it can filter unrelated aspects for aspect phrase f_j. The less observations we have, the larger d_{jk} is. For rare aspect phrases, the constraint can be very loose, and will not have much effect on the clustering process for aspect phrase f_j. In this way, the model can work very robustly.

SDC-constraints are data-driven constraints. Usually we have many reviews about hundreds of products in our dataset. For each aspect phrase, there are $|A| * |P|$ constraints (the number of aspects times the number of product). With thousands of constraints about which aspect it is not likely to belong to, the model learns to which aspect a phrase f_j should be assigned. Although most constraints may be loose because of the limited observations, SDC-constraint can still play an important role in the learning process.

2.3 Sentiment Distribution Consistency Regularized Multinomial Naive Bayes (SDC-MNB)

In this section, we present our probabilistic model which employs both context information and sentiment distribution.

First of all, we extract a context document d for each aspect phrase, which will be described in Section 2.5. In other word, a phrase is represented by its context document. Assuming that the documents in D are independent and identically distributed, the probability of generating D is then given by:

$$p_\theta(D) = \prod_{j=1}^{|D|} p_\theta(d_j) = \prod_{j=1}^{|D|} \sum_{y_j \in A} p_\theta(d_j, y_j) \quad (5)$$

where y_j is a latent variable indicating the aspect label for aspect phrase f_j, and θ is the model parameter.

In our problem, we are actually more interested in the posterior distribution over aspect, i.e., $p_\theta(y_j|d_j)$. Once the learned parameter θ is obtained, we can get our clustering result from $p_\theta(y_j|d_j)$, by assigning aspect a_i with the largest posterior to phrase f_j. We can also enforce SDC-constraint in expectation(on posterior p_θ). We use $q(Y)$ to denote the valid posterior distribution that satisfy our SDC-constraint, and Q to denote the valid posterior distribution space, as follows:

$$Q = \{q(Y) : E_q[z_{ji}|u_{ik} - \hat{s}_{jk}|] \leq d_{jk}, \forall i, j, k\}. \quad (6)$$

Since posterior plays such an important role in joining the context model and SDC-constraint, we formulate our problem in the framework of Posterior Regularization (PR). PR is an efficient framework to inject constraints on the posteriors of latent variables. Instead of restricting p_θ directly, which might not be feasible, PR penalizes the distance of p_θ to the constraint set Q. The posterior-regularized objective is termed as follows:

$$\max_\theta \{\log p_\theta(D) - \min_{q \in Q} KL(q(Y)||p_\theta(Y|D))\} \quad (7)$$

By trading off the data likelihood of the observed context documents (as defined in the first term), and the KL divergence of the posteriors to the valid posterior subspace defined by SDC-constraint (as defined in the second term), the objective encourages models with both desired posterior distribution and data likelihood. In essence, the model attempts to maximize data likelihood of context subject (softly) to SDC-constraint.

2.3.1 Multinomial Naive Bayes

In spirit to (Zhai et al., 2011a), we use Multinomial Naive Bayes (MNB) to model the context document. Let $w_{d_j,k}$ denotes the k^{th} word in document d_j, where each word is from the vocabulary $V = \{w_1, w_2, ..., w_{|V|}\}$. For each aspect phrase f_j, the probability of its latent aspect being a_i and generating context document d_i is

$$p_\theta(d_j, y_j = a_i) = p(a_i) \prod_{k=1}^{|d_j|} p(w_{d_j,k}|a_i) \quad (8)$$

where $p(a_i)$ and $p(w_{d_j,k}|a_i)$ are parameters of this model. Each word $w_{d_j,k}$ is conditionally independent of all other words given the aspect a_i.

Although MNB has been used in existing work for aspect clustering, all of the studies used it in a semi-supervised manner, with labeled data or pseudo-labeled data. In contrast, MNB proposed here is used in an unsupervised manner for aspect-related phrases clustering.

2.3.2 SDC-constraint

As mentioned above, the constraint posterior set Q is defined by

$$Q = \{q(Y) : q(y_j = a_i)|u_{ik} - \hat{s}_{jk}| \leq d_{jk}, \forall i, j, k\}. \quad (9)$$

We can see that Q denotes a set of linear constraints on the projected posterior distribution q. Note that we do not directly observe u_{ik}, the sentiment distribution of aspect a_i on product p_k. For aspect phrase f_j that belongs to aspect a_i, we estimate u_{ik} by counting all sentiment samples. We use the posterior $p_\theta(a_i|d_j)$ to approximately represent how likely phrase f_j belongs to aspect a_i.

$$u_{ik} = \frac{1}{\sum_{j=1}^{|D|} n_{jk} p_\theta(a_i|d_j)} \sum_{j=1}^{|D|} n_{jk} p_\theta(a_i|d_j) \hat{s}_{jk} \quad (10)$$

where $p_\theta(a_i|d_j)$ is short for $p_\theta(y_j = a_i|d_j)$, the probability that aspect phrase f_j belongs to a_i given the context document d_j. We estimate u_{ik} in this way because observations for aspect are relatively sufficient for a reliable estimation since observations for an aspect are aggregated from those for all phrases belonging to that aspect.

2.4 The Optimization Algorithm

The optimization algorithm for the objective (see Eq. 7) is an EM-like two-stage iterative algorithm.

In E-step, we first calculate the posterior distribution $p_\theta(a_i|d_j)$, then project it onto the valid posterior distribution space Q. Given the parameters θ, the posterior distribution can be calculated by Eq. 11.

$$p_\theta(a_i|d_j) = \frac{p(a_i) \prod_{k=1}^{|d_j|} p(w_{d_j,k}|a_i)}{\sum_{r=1}^{|A|} p(a_r) \prod_{k=1}^{|d_j|} p(w_{d_j,k}|a_r)} \quad (11)$$

We use the above posterior distribution to update the sentiment parameter for each aspect by Eq. 10. The projected posterior distribution q is calculated by

$$q = \underset{q \in Q}{argmin}\, \mathbf{KL}(q(Y)||p_\theta(Y|D)) \quad (12)$$

For each instance, there are $|A| * |P|$ constraints. However, we can prune a large number of useless constraints derived from limited observations. All constraints with $d_{jk} > 1$ can be pruned, due to the fact that the parameter u_{ik}, $\hat{s_{jk}}$ is within [0,1], and the difference can not be larger than 1. This optimization problem in Eq. 12 is easily solved via the dual form by the projected gradient algorithm (Boyd and Vandenberghe, 2004):

$$\max_{\lambda \geq 0} \Bigg(-\sum_{i=1}^{|A|} \sum_{k=1}^{|P|} \lambda_{ik} d_{jk} - \log \sum_{i=1}^{|A|} p_\theta(a_i|d_j) exp\{-\sum_{k=1}^{|P|} \lambda_{ik}|u_{ik} - \hat{s}_{jk}|\} - \epsilon\|\lambda\| \Bigg) \quad (13)$$

where ϵ controls the slack size for constraint. After solving the above optimization problem and obtaining the optimal λ, we can calculate the projected posterior distribution q by

$$q(y_j = a_i) = \frac{1}{Z} p_\theta(a_i|d_j) exp\{-\sum_{k=1}^{|P|} \lambda_{ik}|u_{ik} - \hat{s}_{jk}|\} \quad (14)$$

where Z is the normalization factor. Note that *sentiment distribution consistency* is actually modeled as instance-level constraint here, which makes it very efficient to solve.

In M-step, the projected posteriors $q(Y)$ are then used to compute sufficient statistics and update the models parameters θ. Given the projected posteriors $q(Y)$, the parameters can be updated by Eq. 15,16.

$$p(a_i) = \frac{1 + \sum_{j=1}^{|D|} q(y_j = a_i)}{|A| + |D|} \quad (15)$$

$$p(w_t|a_i) = \frac{1 + \sum_{j=1}^{|D|} N_{ti} q(y_j = a_i)}{|V| + \sum_{m=1}^{|V|} \sum_{j=1}^{|D|} N_{mj} q(y_j = a_i)} \quad (16)$$

where N_{tj} is the number of times that the word w_t occurs in document d_j.

The parameters are initialized randomly, and we repeat E-step and M-step until convergence.

2.5 Data Extraction

2.5.1 Context Extraction

In order to extract the context document d for each aspect phrase, we follow the approach in Zhai et al. (2011a). For each aspect phrase, we generate its context document by aggregating the surrounding texts of the phrase in all reviews. The preceding and following t words of a phrase are taken as the context where we set $t = 3$ in this paper. Stopwords and other aspect phrases are removed. For example, the following review contains two aspect phrases, "*screen*" and "*picture*",

The LCD ***screen*** *gives clear* ***picture***.

For "*screen*", the surrounding texts are {the, LCD, gives, clear, picture}. We remove stopwords "the", and the aspect term "*picture*", and the resultant context of "*screen*" in this review is

context(*screen*) ={LCD, screen, gives, clear}.

Similarly, the context of "*picture*" in this review is

context(*picture*) ={gives, clear}.

By aggregating the contexts of all the reviews that contain aspect phrase f_j, we obtain the corresponding context document d_j.

2.5.2 Sentiment Extraction

Since we use semi-structured reviews, we obtain the estimated sentiment distribution by simply counting how many times each aspect phrase appears in *Pros* and *Cons* reviews for each product respectively. So for each aspect phrase f_j, let n^+_{jk} denotes the times that f_j appears in *Pros* of all reviews for product p_k, and let n^-_{jk} denotes the times that f_j appears in *Cons* of all reviews for product p_k. So the total number of occurrence of a phrase is $n_{jk} = n^+_{jk} + n^-_{jk}$. We have samples like (1,1,1,0,0) where 1 means a phrase occurs in *Pros* of a review, and 0 in *Cons*. Given a sequence of such observations, the sample mean is easily computed as $\hat{s}_{jk} = \frac{n^+_{jk}}{n^+_{jk}+n^-_{jk}}$. And the sample standard deviation is $\hat{\sigma}_{jk} = \sqrt{\frac{(1-\hat{s}_{jk})^2 * n^+_{jk} + (\hat{s}_{jk})^2 * n^-_{jk}}{n_{jk}-1}}$.

3 Experiments

3.1 Data Preparation

The details of our review corpus are given in Table 2. This corpus contains semi-structured customer reviews from four domains: *Camera*, *Cellphone*, *Laptop*, and *MP3*. These reviews were crawled from the following web sites: www.amazon.cn, www.360buy.com, www.newegg.com.cn, and www.zol.com. The aspect label of each aspect phrases is annotated by human curators.

	Camera	Cellphone	Laptop	MP3
#Products	449	694	702	329
#Reviews	101,235	579,402	102,439	129,471
#Aspect Phrases	236	230	238	166
#Aspect	12	10	14	8

Table 2: Statistics of the review corpus. # denotes the size.

3.2 Evaluation Measures

We adapt three measures *Purity*, *Entropy*, and *Rand Index* for performance evaluation. These measures have been commonly used to evaluate clustering algorithms.

Given a data set DS, suppose its gold-standard partition is $G = \{g_1, ..., g_j, ..., g_k\}$, where k is the number of clusters. A clustering algorithm partitions DS into k disjoint subsets, say $DS_1, DS_2, ..., DS_k$.

Entropy: For each resulting cluster, we can measure its entropy using Eq. 17, where $P_i(g_j)$ is the proportion of data points of class g_j in DS_i. The entropy of the entire clustering result is calculated by Eq. 18.

$$entropy(DS_i) = -\sum_{j=1}^{k} P_i(g_j) log_2 P_i(g_j) \quad (17)$$

$$entropy(DS) = \sum_{i=1}^{k} \frac{|DS_i|}{|DS|} entropy(DS_i) \quad (18)$$

Purity: *Purity* measures the extent that a cluster contains only data from one gold-standard partition. The cluster purity is computed with Eq. 19. The total purity of the whole clustering result (all clusters) is computed with Eq. 20.

$$purity(DS_i) = \max_j P_i(g_j) \quad (19)$$

$$purity(DS) = \sum_{i=1}^{k} \frac{|DS_i|}{|DS|} purity(DS_i) \quad (20)$$

RI: The *Rand Index(RI)* penalizes both false positive and false negative decisions during clustering. Let TP (True Positive) denotes the number of pairs of elements that are in the same set in DS and in the same set in G. TN (True Negative) denotes number of pairs of elements that are in different sets in DS and in different sets in G. FP (False

	Camera			Cellphone			Laptop			MP3		
	P	RI	E	P	RI	E	P	RI	E	P	RI	E
Kmeans	43.48%	83.52%	2.098	48.91%	84.80%	1.792	43.46%	87.11%	2.211	40.00%	70.98%	2.047
L-EM	54.89%	87.07%	**1.690**	51.96%	86.64%	1.456	48.94%	84.53%	2.039	44.24%	75.37%	1.990
LDA	36.84%	83.28%	2.426	48.65%	85.33%	1.833	35.02%	83.53%	2.660	36.12%	76.08%	2.296
Constraint-LDA	43.30%	86.01%	2.216	47.89%	86.04%	1.974	32.35%	84.86%	2.676	50.70%	81.42%	1.924
SDC-MNB	**56.42%**	**88.16%**	1.725	**67.95%**	**90.62%**	**1.266**	**55.52%**	**90.72%**	**1.780**	**58.06%**	**83.57%**	**1.578**

Table 3: Comparison to unsupervised baselines. (P is short for purity, E for entropy, and RI for random index.)

Positive) denotes number of pairs of elements in S that are in the same set in DS and in different sets in G. FN (False Negative) denotes number of pairs of elements that are in different sets in DS and in the same set in G. The *Rand Index(RI)* is computed with Eq. 21.

$$RI(DS) = \frac{TP + TN}{TP + TN + FP + FN} \tag{21}$$

3.3 Evaluation Results

3.3.1 Comparison to unsupervised baselines

We compared our approach with several existing unsupervised methods. Some of the methods augmented unsupervised models by incorporating lexical similarity and other domain knowledge. All of them are context-based models.[2] We list these models as follows.

- ***Kmeans***: Kmeans is the most popular clustering algorithm. Here we use the context distributional similarity (cosine similarity) as the similarity measure.

- ***L-EM***: This is a state-of-the-art unsupervised method for clustering aspect phrases (Zhai et al., 2011a). L-EM employed lexical knowledge to provide a better initialization for EM.

- ***LDA***: LDA is a popular topic model(Blei et al., 2003). Given a set of documents, it outputs groups of terms of different topics. In our case, each aspect phrase is processed as a term. [3] Each sentence in a review is considered as a document. Each aspect is considered as a topic. In LDA, a term may belong to more than one topic/group, but we take the topic/group with the maximum probability.

- ***Constraint-LDA***: Constraint-LDA (Zhai et al., 2011b) is a state-of-the-art LDA-based method that incorporates must-link and cannot-link constraints for this task. We set the damping factor $\lambda = 0.3$ and relaxation factor $\eta = 0.9$, as suggested in the original reference.

For all methods that depend on the random initiation, we use the average results of 10 runs as the final result. For all LDA-based models, we choose $\alpha = 50/T$, $\beta = 0.1$, and run 1000 iterations.

Experiment results are shown in Table 3. We can see that our approach almost outperforms all unsupervised baseline methods by a large margin on all domains. In addition, we have the following observations:

- LDA and Kmeans perform poorly due to the fact that the two methods do not use any prior knowledge. It is also shown that only using the context distributional information is not sufficient for clustering aspect phrases.

- Constraint-LDA and L-EM that utilize prior knowledge perform better. We can see that Constraint-LDA outperforms LDA in terms of RI (Rand Index) on all domains. L-EM achieves the best results against the baselines. This demonstrates the effectiveness to incorporate prior knowledge.

- SDC-MNB produces the optimal results among all models for clustering. Methods that use must-links and cannot-links may suffer from noisy links. For L-EM, we find that it is sensitive to noisy must-links. As L-EM assumes that must-link is transitive, several noisy must-links may totally mislabel the softly annotated data. For Constraint-LDA, it is more robust than L-EM, because it doesn't assume the transitivity of must-link. However, it only promotes the RI (Rand Index) consistently by leveraging pair-wise prior knowledge, but sometimes it hurts the

[2] In our method, we collect context document for each aspect phrase. This process is conducted for L-EM and K-means. But for LDA and Constraint-LDA, we take each sentence of reviews as a document. This setting for the LDA baselines is adapted from previous work.

[3] Each aspect phrase is pre-processed as a single word (e.g., "battery life" is treated as battery-life). Other words are normally used in LDA.

performance with respect to purity or entropy. Our method is consistently better on almost all domains, which shows the advantages of the proposed model.

- SDC-MNB is remarkably better than baselines, particularly for the *cellphone* domain. We argue that this is because we have the largest number of reviews for each product in the *cellphone* domain. The larger dataset gives us more observations on each phrase, so that we obtain more reliable estimation of model parameters.

3.3.2 Comparison to supervised baselines

We further compare our methods with two supervised models. For each supervised model, we provide a proportion of manually labeled data for training, which is randomly selected from gold-standard annotations. However, we didn't use any labeled data for our approach.

- ***MNB***: The labeled seeds are used to train a MNB classifier to classify all unlabeled aspect phrases into different classes.

- ***L-Kmeans***: In L-Kmeans, the clusters of the labeled seeds are fixed at the initiation and remain unchanged during iteration.

	Purity	RI	Entropy
MNB-5%	53.21%	85.77%	1.854
MNB-10%	**59.55%**	**86.70%**	**1.656**
MNB-15%	66.06%	88.39%	1.449
L-Kmeans-10%	53.54%	86.15%	1.745
L-Kmeans-15%	**57.00%**	**86.89%**	**1.643**
L-Kmeans-20%	60.97%	87.63%	1.528
SDC-MNB	**59.49%**	**88.26%**	**1.580**

Table 4: Comparison to supervised baselines. MNB-5% means MNB with 5% labeled data.

We experiment with several settings: taking 5%, 10% and 15% of the manually labeled aspect phrases for training, and the remainder as unlabeled data. Experiment results is shown in Table 4 (the results are averaged over 4 domains). We can see that our unsupervised approach is roughly as good as the supervised MNB with 10% labeled data. Our unsupervised approach is also slightly better than L-Kmeans with 15% labeled data. This result further demonstrates the effectiveness of our model.

3.3.3 Influence of parameters

We vary the confidence level from 90% to 99.9% to see how it impacts on the performance of SDC-MNB. The results are presented in Fig. 5 (the results are averaged over 4 domains). We can see that the performance of clustering is fairly stable when changing the confidence level, which implies the robustness of our model.

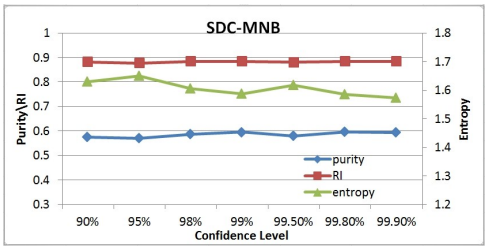


Figure 5: Influence of the confidence level on SDC-MNB.

3.3.4 Analysis of SDC-constraint

As mentioned in Section 2.2, SDC-constraint is dependent on the number of observations. More observations we get, more informative the constraint is, which means the constraint is tighter and d_{jk} (see Eq.4) is smaller. For all k, we count how many d_{jk} is less than 0.2 (and 1) on average for each aspect phrase f_j. d_{jk} is calculated with a confidence level of 99%. The statistics of constraints is given in Table 5. We can see that the cellphone domain has the most informative and largest constraint set, that may explain why SDC-MNB achieves the largest purity gain(over L-EM) in cellphone domain.

	$\#(d_{jk} < 0.2)$	$\#(0.2 < d_{jk} < 1)$	purity gain
Camera	3.02	8.78	1.53%
Cellphone	17.29	30.5	15.99%
Laptop	4.6	13.22	6.58%
MP3MP4	6.1	10.7	13.82%

Table 5: Constraint statistics on different domains.

4 Related Work

Our work is related to two important research topics: aspect-level sentiment analysis, and constraint-driven learning. For aspect-level sentiment analysis, aspect extraction and clustering are key tasks. For constraint-driven learning, a variety of frameworks and models for sentiment analysis have been studied extensively.

There have been many studies on clustering aspect-related phrases. Most existing studies are

based on context information. Some works also encoded lexical similarity and synonyms as prior knowledge. Carenini et al. (2005) proposed a method that was based on several similarity metrics involving string similarity, synonyms, and lexical distances defined with WordNet. Guo et al. (2009) proposed a multi-level latent semantic association model to capture expression-level and context-level topic structure. Zhai et al. (2010) proposed an EM-based semi-supervised learning method to group aspect expressions into user-specified aspects. They employed lexical knowledge to provide a better initialization for EM. In Zhai et al. (2011a), an EM-based unsupervised version was proposed. The so-called L-EM model first generated softly labeled data by grouping feature expressions that share words in common, and then merged the groups by lexical similarity. Zhai et al. (2011b) proposed a LDA-based method that incorporates must-link and cannot-link constraints.

Another line of work aimed to extract and cluster aspect words simultaneously using topic modeling. Titov and McDonald (2008) proposed the multi-grain topic models to discover global and local aspects. Branavan et al. (2008) proposed a method which first clustered the key-phrases in Pros and Cons into some aspect categories based on distributional similarity, then built a topic model modeling the topics or aspects. Zhao et al. (2010) proposed the MaxEnt-LDA (a Maximum Entropy and LDA combination) hybrid model to jointly discover both aspect words and aspect-specific opinion words, which can leverage syntactic features to separate aspects and sentiment words. Mukherjee and Liu (2012) proposed a semi-supervised topic model which used user-provided seeds to discover aspects. Chen et al. (2013) proposed a knowledge-based topic model to incorporate must-link and cannot-link information. Their model can adjust topic numbers automatically by leveraging cannot-link.

Our work is also related to general constraint-driven(or knowledge-driven) learning models. Several general frameworks have been proposed to fully utilize various prior knowledge in learning. Constraint-driven learning (Chang et al., 2008) (CODL) is an EM-like algorithm that incorporates per-instance constraints into semi-supervised learning. Posterior regularization (Graca et al., 2007) (PR) is a modified EM algorithm in which the E-step is replaced by the projection of the model posterior distribution onto the set of distributions that satisfy auxiliary expectation constraints. Generalized expectation criteria (Druck et al., 2008) (GE) is a framework for incorporating preferences about model expectations into parameter estimation objective functions. Liang et al. (2009) developed a Bayesian decision-theoretic framework to learn an exponential family model using general measurements on the unlabeled data. In this paper, we model our problem in the framework of posterior regularization.

Many works promoted the performance of sentiment analysis by incorporating prior knowledge as weak supervision. Li and Zhang (2009) injected lexical prior knowledge to non-negative matrix tri-factorization. Shen and Li (2011) further extended the matrix factorization framework to model dual supervision from both document and word labels. Vikas Sindhwani (2008) proposed a general framework for incorporating lexical information as well as unlabeled data within standard regularized least squares for sentiment prediction tasks. Fang (2013)proposed a structural learning model with a handful set of aspect signature terms that are encoded as weak supervision to extract latent sentiment explanations.

5 Conclusions

Aspect finding and clustering is an important task for aspect-level sentiment analysis. In order to cluster aspect-related phrases, this paper has explored a novel concept, *sentiment distribution consistency*. We formalize the concept as soft constraint, integrate the constraint with a context-based probabilistic model, and solve the problem in the posterior regularization framework. The proposed model is also designed to be robust with both sufficient and insufficient observations. Experiments show that our approach outperforms state-of-the-art baselines consistently.

Acknowledgments

This work was partly supported by the following grants from: the National Basic Research Program (973 Program) under grant No.2012CB316301 and 2013CB329403, the National Science Foundation of China project under grant No.61332007 and No. 61272227, and the Beijing Higher Education Young Elite Teacher Project.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press, New York, NY, USA.

S. R. K. Branavan, Harr Chen, Jacob Eisenstein, and Regina Barzilay. 2008. Learning document-level semantic properties from free-text annotations. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Giuseppe Carenini, Raymond T. Ng, and Ed Zwart. 2005. Extracting knowledge from evaluative text. In *Proceedings of the 3rd International Conference on Knowledge Capture*, K-CAP '05, pages 11–18, New York, NY, USA. ACM.

Ming-Wei Chang, Lev Ratinov, Nicholas Rizzolo, and Dan Roth. 2008. Learning and inference with constraints. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pages 1513–1518. AAAI Press.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Mal Castellanos, and Riddhiman Ghosh. 2013. Exploiting domain knowledge in aspect extraction. In *EMNLP*, pages 1655–1667. ACL.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 595–602, New York, NY, USA. ACM.

Lei Fang, Minlie Huang, and Xiaoyan Zhu. 2013. Exploring weakly supervised latent sentiment explanations for aspect-level review analysis. In Qi He, Arun Iyengar, Wolfgang Nejdl, Jian Pei, and Rajeev Rastogi, editors, *CIKM*, pages 1057–1066. ACM.

Joao V. Graca, Lf Inesc-id, Kuzman Ganchev, Ben Taskar, Joo V. Graa, L F Inesc-id, Kuzman Ganchev, and Ben Taskar. 2007. Expectation maximization and posterior constraints. In *In Advances in NIPS*, pages 569–576.

Honglei Guo, Huijia Zhu, Zhili Guo, XiaoXun Zhang, and Zhong Su. 2009. Product feature categorization with multilevel latent semantic association. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1087–1096, New York, NY, USA. ACM.

Tao Li, Yi Zhang, and Vikas Sindhwani. 2009. A non-negative matrix tri-factorization approach to sentiment classification with lexical prior knowledge. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 244–252, Stroudsburg, PA, USA. Association for Computational Linguistics.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning from measurements in exponential families. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 641–648, New York, NY, USA. ACM.

Arjun Mukherjee and Bing Liu. 2012. Aspect extraction through semi-supervised modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 339–348, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chao Shen and Tao Li. 2011. A non-negative matrix factorization based approach for active dual supervision from document and word labels. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 949–958, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vikas Sindhwani and Prem Melville. 2008. Document-word co-regularization for semi-supervised sentiment analysis. In *ICDM*, pages 1025–1030. IEEE Computer Society.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 111–120, New York, NY, USA. ACM.

Zhongwu Zhai, Bing Liu, Hua Xu, and Peifa Jia. 2010. Grouping product features using semi-supervised learning with soft-constraints. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1272–1280, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhongwu Zhai, Bing Liu, Hua Xu, and Peifa Jia. 2011a. Clustering product features for opinion mining. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 347–354, New York, NY, USA. ACM.

Zhongwu Zhai, Bing Liu, Hua Xu, and Peifa Jia. 2011b. Constrained lda for grouping product features in opinion mining. In *Proceedings of the 15th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part I*, PAKDD'11, pages 448–459, Berlin, Heidelberg. Springer-Verlag.

Wayne X. Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 56–65, Stroudsburg, PA, USA. Association for Computational Linguistics.

Automatic Generation of Related Work Sections in Scientific Papers: An Optimization Approach

Yue Hu and Xiaojun Wan
Institute of Computer Science and Technology
The MOE Key Laboratory of Computational Linguistics
Peking University, Beijing, China
{ayue.hu,wanxiaojun}@pku.edu.cn

Abstract

In this paper, we investigate a challenging task of automatic related work generation. Given multiple reference papers as input, the task aims to generate a related work section for a target paper. The generated related work section can be used as a draft for the author to complete his or her final related work section. We propose our Automatic Related Work Generation system called ARWG to address this task. It first exploits a PLSA model to split the sentence set of the given papers into different topic-biased parts, and then applies regression models to learn the importance of the sentences. At last it employs an optimization framework to generate the related work section. Our evaluation results on a test set of 150 target papers along with their reference papers show that our proposed ARWG system can generate related work sections with better quality. A user study is also performed to show ARWG can achieve an improvement over generic multi-document summarization baselines.

1 Introduction

The related work section is an important part of a paper. An author often needs to help readers to understand the context of his or her research problem and compare his or her current work with previous works. A related work section is often used for this purpose to show the differences and advantages of his or her work, compared with related research works. In this study, we attempt to automatically generate a related work section for a target academic paper with its reference papers. This kind of related work sections can be used as a basis to reduce the author's time and effort when he or she wants to complete his or her final related work section.

Automatic related work section generation is a very challenging task. It can be considered a topic-biased, multiple-document summarization problem. The input is a target academic paper, which has no related work section, along with its reference papers. The goal is to create a related work section that describes the related works and addresses the relationship between the target paper and the reference papers. Here we assume that the set of reference papers has been given as part of the input. Existing works in the NLP and recommendation systems communities have already focused on the task of finding reference papers. For example, citation prediction (Nallapati et al., 2008) aims at finding individual paper citation patterns.

Generally speaking, automatic related work section generation is a strikingly different problem and it is much more difficult in comparison with general multi-document summarization tasks. For example, multi-document summarization of news articles aims at synthesizing contents of similar news and removing the redundant information contained by the different news articles. However, each scientific paper has much specific content to state its own work and contribution. Even for the papers that investigate the same research topic, their contributions and contents can be totally different. The related work section generation task needs to find the specific contributions of individual papers and arrange them into one or several paragraphs.

In this study, we focus on the problem of automatic related work section generation and propose a novel system called ARWG to address the

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1624–1633,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

problem. For the target paper, we assume that the abstract and introduction sections have already been written by the author and they can be used to help generate the related work section. For the reference papers, we only consider and extract the abstract, introduction, related work and conclusion sections, because other sections like the method and evaluation sections always describe the extreme details of the specific work and they are not suitable for this task. Then we generate the related work section using both sentence sets which are extracted from the target paper and reference papers, respectively.

Firstly, we use a PLSA model to group both sentence sets of the target paper and its reference papers into different topic-biased clusters. Secondly, the importance of each sentence in the target paper and the reference papers is learned by using two different Support Vector Regression (SVR) models. At last, a global optimization framework is proposed to generate the related work section by selecting sentences from both the target paper and the reference papers. Meanwhile, the framework selects sentences from different topic-biased clusters globally.

Experimental results on a test set of 150 target papers show our method can generate related work sections with better quality than those of several baseline methods. With the ROUGE toolkit, the results indicate the related work sections generated by our system can get higher ROUGE scores. Moreover, our related work sections can get higher rating scores based on a user study. Therefore, our related work sections can be much more suitable for the authors to prepare their final related work sections.

2 Related Work

There are few studies to directly address automatic related work generation. Hoang and Kan (2010) proposed a related work summarization system given the set of keywords arranged in a hierarchical fashion that describes the paper's topic. They used two different rule-based strategies to extract sentences for general topics as well as detailed ones.

A few studies focus on multi-document scientific article summarization. Agarwal et al., (2011) introduced an unsupervised approach to the problem of multi-document summarization. The input is a list of papers cited together within the same source article. The key point of this approach is a topic based clustering of fragments extracted from each co-cited article. They rank all the clusters using a query generated from the context surrounding the co-cited list of papers. Yeloglu et al., (2011) compared four different approaches for multi-document scientific articles summarization: MEAD, MEAD with corpus specific vocabulary, LexRank and W3SS.

Other studies investigate mainly on the single-document scientific article summarization. Early works including (Luhn 1958; Baxendale 1958; Edumundson 1969) tried to use various features specific to scientific text (e.g., sentence position, or rhetorical clues features). They have proved that these features are effective for the scientific article summarization. Citation information has been already shown effective in summarize the scientific articles. Works including (Mei and Zhai 2008; Qazvinian and Radev 2008; Schwartz and Hearst 2006; Mohammad et al., 2009) employed citation information for the single scientific article summarization. Earlier work (Nakov et al., 2004) indicated that citation sentences may contain important concepts that can give useful descriptions of a paper.

Various methods have been proposed for news document summarization, including rule-based methods (Barzilay and Elhadad 1997; Marcu and Daniel 1997), graph-based methods (Mani and Bloedorn 2000; Erkan and Radev 2004; Michalcea and Tarau 2005), learning-based methods (Conroy et al., 2001; Shen et al., 2007; Ouyang et al., 2007; Galanis et al., 2008), optimization-based methods (McDonald 2007; Gillick et al., 2009; Xie et al., 2009; Berg-Kirkpatrick et al., 2011; Lei Huang et al., 2011; Woodsend et al., 2012; Galanis 2012), etc.

The most relevant work is (Hoang and Kan, 2010) as mentioned above. They also assumed the set of reference papers was given as part of the input. They also adopt the hierarchical topic tree that describes the topic structure in the target paper as an essential input for their system. However, it is non-trivial to build the hierarchical topic tree. Moreover, they do not consider the content of the target paper to construct the related work section, which is actually crucial in the related work section. To the best of our knowledge, no previous works have used supervised learning and optimization framework to deal with the multiple scientific article summarization tasks.

3 Problem Analysis and Corpus

3.1 Problem Analysis

We firstly analyze the structure of related work sections briefly. By using examples for illustration, we can gain insight on how to generate related work sections. A specific related work example is shown in Figure 1.

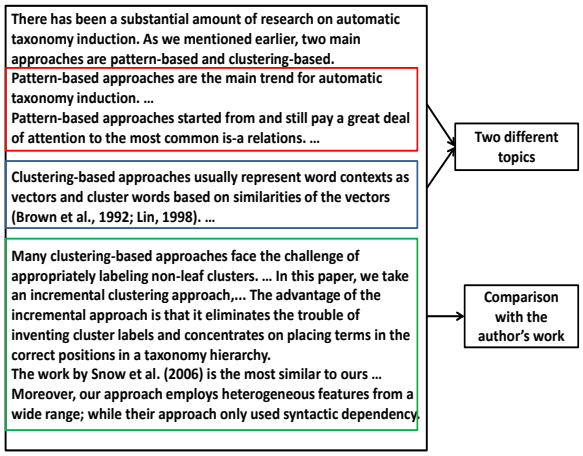


Figure 1: A sample related work section (Yang and Callan 2009)

This related work section introduces previous related works for a paper on Automatic Taxonomy Induction. From Figure 1, we can have a glance at the structure of related work sections. Related work sections usually discuss several different topics, such as "pattern-based" and "cluster-based" approaches shown in the Figure 1. Besides the knowledge of previous works, the author often compares his own work with the previous works. The differences and advantages are generally mentioned. The example in Figure 1 also indicates this phenomenon.

Therefore, we design our system to generate related work sections according to the related work section structure mentioned above. Our system takes the target paper for which a related work section needs to be drafted besides its reference papers as input. The goal of our system is to generate a related work section with the above structure. The generated related work section should have several topic-biased parts. The author's own work is also needed to be described and its difference with other works is needed to be emphasized on.

3.2 Corpus and Preprocessing

We build a corpus that contains academic papers and their corresponding reference papers. The academic papers are selected from the ACL Anthology[1]. The ACL Anthology currently hosts over 24,500 papers from major conferences such as ACL, EMNLP, COLING in the fields of computational linguistics and natural language processing. We remove the papers that contain related work sections with very short length, and randomly select 1050 target papers to construct our whole corpus.

The papers are all in PDF format. We extract their texts by using PDFlib[2] and detect their physical structures of paragraphs, subsections and sections by using ParsCit[3]. For the target papers, the related work sections are directly extracted as the gold summaries. The references are also extracted. For the references that can be found in the ACL Anthology, we download them from the ACL Anthology. The other reference papers are searched and downloaded by using Google Scholar. References to books and PhD theses are discarded, for their verbosity may change the problem drastically (Mihalcea and Ceylan, 2007).

The input of our system includes the abstract and introduction sections of the target paper, and the abstract, introduction, related work and conclusion sections of the reference papers. As mentioned above, the method and evaluation sections in the reference papers are not used as input because these sections usually describe extreme details of the methods and evaluation results and they are not suitable for related work generation. Note that it is reasonable to make use of the abstract and introduction sections of the target paper to help generate the related work section, because an author usually has already written the abstract and introduction sections before he or she wants to write the related work section for the target paper. Otherwise, we cannot get any information about the author's own work. All other sections in the target paper are not used.

4 Our Proposed System

4.1 Overview

In this paper, we propose a system called ARWG to automatically generate a related work section for a given target paper. The architecture of our system is shown in Figure 2. We take both the target paper and its reference papers as input and they are represented by several sections mentioned in Section 3.2. After preprocessing, we extract the feature vectors for sentences in the target paper and the reference papers, respective-

[1] http://aclweb.org/anthology/

[2] http://www.pdflib.com/

[3] http://aye.comp.nus.edu.sg/parsCit/

ly. The importance scores for sentences in the target paper and the reference papers are assigned by using two SVR based sentence scoring models. The two SVR models are trained for sentences in the target paper and the reference papers, respectively. Meanwhile, a topic model is applied to the whole set of sentences in both the target paper and reference papers. The sentences are grouped into several different topic-biased clusters. The sentences with importance scores and topic cluster information are taken as the input for the global optimization framework. The optimization framework extracts sentences to describe both the author's own work and background knowledge. More details of each part will be discussed in the following sections.

4.2 Topic Model Learning

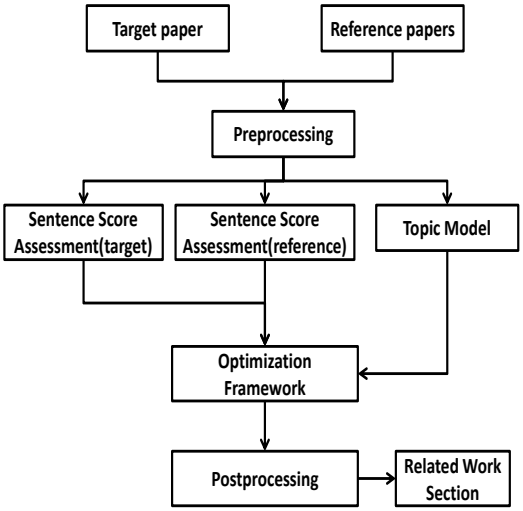


Figure 2: System Architecture

As mentioned in the previous section, the related work section usually addresses several different topics. The topics may be different research themes or different aspects of a broad research theme. The related work section should describe the specific details for each topic, respectively.

Therefore, we aim to discover the hidden topics of the input papers, and we use the Probabilistic latent semantic analysis (PLSA) (Hofmann, 1999) to solve this problem.

The PLSA approach models each word in a document as a sample from a mixture model. The mixture components are multinomial random variables that can be viewed as representations of "topics". Different words in a document may be generated from different topics. Each document is represented a list of mixing proportions for these mixture components and can be reduced to a probability distribution on a fixed set of topics.

Considering that the sentences in one paper may relate to different topics, we treat each sentence as a "document" *d*. We treat the noun phases in the sentences as the "words" *w*. In order to extract the noun phrases, chunking implemented by the OpenNLP toolkit [4] is applied to the sentences. Noun phrases that contain words such as "paper" and "data" are discarded.

Then the sentences with their corresponding noun phrases are taken as input into the PLSA model. Here both the sentences in the target paper and the sentences in the reference papers are treated the same in the model. Finally, we can get the sentence set with topic information and use it in the subsequent steps. Each sentence has a topic weight *t* in each topic.

4.3 Sentence Important Assessment

In our proposed system, sentence importance assessment aims to assign an importance score to each sentence in the target paper and reference papers. The score of each sentence will be used in the subsequent optimization framework. We propose to use the support vector regression model to achieve this goal. In the above topic model learning process, we do not distinguish the sentences in the target paper and reference papers. In contrast, we train two different support vector regression models separately for the sentences in the target paper and the sentences in the reference papers. In the related work section, the sentences that describe the author's own work usually address the differences from the related works, while the sentences that describe the related works often focus on the specific details. We think the two kinds of sentences should be treated differently.

Scoring Method

To construct training data based on the papers collected, we apply a similarity scoring method to assign the importance scores to the sentences in the papers. The main hypothesis is that the sentences in the gold related work sections should summarize the target paper and reference papers as well. Thus the sentences in the papers which are more similar to the sentences in the gold related work sections should be considered more important and suitable to be selected. Our scoring method should assign higher scores to them.

[4] http://opennlp.apache.org/

We define the importance score of a sentence in the papers as below:

$$score(s) = \max_{s_i^* \in S^*}(sim(s, s_i^*)) \quad (1)$$

where s is a sentence in the papers, S^* is the set of the sentences in the corresponding gold related work section. The standard cosine measure is employed as the similarity function.

Considering the difference between the sentences that describe the author's work and the sentences that describe the related works, we split the set of sentences in the gold related work section into two parts: one discusses the author's own work and the other introduces the related works. We observe that sentences related to the author's own work often feature specific words or phrases (such as "we", "our work", "in this paper" etc.) in the related work section. So we check the sentences about whether they contain clue words or phrases (i.e., "in this paper", "our work" and 18 other phrases). If the clue phrase check fails, the sentence belongs to the related work part. If not, it belongs the own work part.

Thus for the sentences in the target paper, S^* is the set of sentences in the own work part of the gold related work section, while for the sentences in the reference papers, S^* is the set of sentences in the related work part of the gold related work section. Then we can use the scoring method to compute the target scores of the sentences in the training set. It is noteworthy that two SVR models can be trained on the two parts of the training data, respectively.

Feature

Each sentence is represented by a set of features. The common features used for the sentences of the target paper and reference papers are shown in Table 1. The additional features applied to the sentences of the target paper are introduced in Table 2.

Here, *s* is a sentence that needs to extract features. *th* is paper title, section headings and subsection headings set of the reference papers or target paper for the two SVR models, respectively. Each feature with "*" represent a feature set that contains similar features.

All the features are scaled into [0, 1]. Thus we can learn SVR models based on the features and importance scores of the sentences, and then use the models to predict an importance score for each sentence in the test set. The SVR models are trained and applied for the target paper and reference papers, respectively.

Table 1: Common features employed in the SVR models

Feature	Description
$Sim(s,th)^*$	The similarity between *s* and each title in *th*; Stop words are removed and stemming is employed.
WS(s,th)	Number of words shared by *s* and *th*.
$SP(s)^*$	The position of *s* in its section or subsection
$PTI(s)^*$	The parse tree information of *s*, including the number of noun phrase and verb phrases, the depth of the parse tree, etc.
$IsHead(s)^*$	Indicates whether *s* is the first sentence of the section or subsection
$IsEnd(s)^*$	Indicates whether *s* is the last sentence of the section or subsection
SWP(s)	The percentage of the stop words
Length(s)	The length of sentence *s*
Length_rw(s)	The length of *s* after removing stop words
SI(s)	The section index of *s* that indicates which section *s* is from.
$CluePhrase(s)^*$	Indicates whether a clue phrase appears in *s*. the clue phrases include "our work", "propose" and other 20 words. Each clue phrase corresponds to one feature.

Table 2: Additional features for sentences in the target paper

Feature	Description
HasCitation(s)	Indicates whether *s* contains a citation
$PhraseForCmp(s)^*$	Indicates whether *s* contains words or phrases used for comparison such as "in contrast", "instead" and other 26 words. Each word or phrase corresponds to one feature.

4.4 A Global Optimization Framework

In the above steps, we can get the predicted importance score and topic information for each sentence in the target paper and reference papers. Here, we introduce a global optimization framework to generate the related work section.

According to the structure of the related work section mentioned above, the related work section usually discusses several topics. In each topic, the related works and their details are introduced. Besides, the author often compares his own work with these previous works.

Therefore, we propose to formulate the generation as an optimization problem. Basically, we will be searching for a set of sentences to optimize the objective function.

Table 3: Notations used in this section

Symbol	Description
sr_i/st_i	the sentence in the reference/target paper
lr_i/lt_i	the length of sentence sr_i/ st_i
wr_i/wt_i	the importance score of sr_i/st_i
xr_{ij}/xt_{ij}	indicates whether sr_i/st_i is selected into the part of topic *j* in the generated related work section
nr/nt	the number of sentences in the reference/target papers
m	the topic count
t_{ij}	the topic weight of sr_i/st_i in topic j from the PLSA model
B	the set of unique bigrams
y_i	indicates whether bigram b_i is included in the result
c_{b_i}	the count of the occurrences of bigram b_i in the both target paper and reference papers
L_{max}	the maximum word count of the related work section
L_j	the maximum word count of the part of topic j which depends on the percentage of sentences belong to topic j
B^*	the total set of bigrams in the whole paper set
B_i	the set of bigrams that sentence sr_i/st_i contains
Sr_m/St_m	the set of sentences that include bigram b_m in the reference/target papers
$\lambda_1, \lambda_2, \lambda_3$	parameters for tuning

To design the objective function, three aspects should be considered:

1) First, the related work section we generate should introduce the previous works well. In our assumption, sentences with higher importance scores are better to be selected. In addition, very short sentences should be penalized. So we introduce the first part of our objective function below:

$$\sum_{i=1}^{nr}(lr_i wr_i \sum_{j=1}^{m} t_{ij} xr_{ij}) \quad (2)$$

We add the sentence length as a multiplication factor in order to penalize the very short sentences, or the objective function tends to select more and shorter sentences. At the same time, the objective function does not tend to select the very long sentences. The total length of the sentences selected is fixed. So if the objective function tends to select the longer sentences, the fewer sentences can be selected. A tradeoff needs to be made between the number and the average length of the sentences selected.

The constraints introduced below ensure that the sentence can only be selected into one topic and the topic weight is used to measure the degree that the sentence is relevant to the specific topic.

2) Second, similar to the first part, we should consider the own work part of the related work section. Thus the second part of our objective function is shown as follows:

$$\sum_{i=1}^{nt}(lt_i wt_i \sum_{j=1}^{m} t_{ij} xt_{ij}) \quad (3)$$

3) At last, redundancy reduction should be considered in the objective function. The last part of the objective function is shown below:

$$\sum_{i=1}^{|B|} c_{b_i} y_i \quad (4)$$

The intuition is that the more unique bigrams the related work section contains, the less redundancy the related work section has. We add c_{b_i} as the weight of the bigram in order to include more important bigrams.

By combing all the parts defined above, we have the following full objective function:

$$\max_{xr,xt} \lambda_1 \sum_{i=1}^{nr}(\frac{lr_i}{\alpha L_{max}} wr_i \sum_{j=1}^{m} t_{ij} xr_{ij}) + \lambda_2 \sum_{i=1}^{nt}(\frac{lt_i}{(1-\alpha)L_{max}} wt_i \sum_{j=1}^{m} t_{ij} xt_{ij}) + \lambda_3 \sum_{i=1}^{|B|} \frac{c_{b_i} y_i}{|B^*|} \quad (5)$$

Subject to:

$$\sum_{i=1}^{nr} lr_i xr_{ij} + \sum_{i=1}^{nt} lt_i xt_{ij} < L_j, for\ j = 1, \dots, m \quad (6)$$

$$\sum_{i=1}^{nr} \sum_{j=1}^{m} lr_i xr_{ij} < \alpha L_{max} \quad (7)$$

$$\sum_{i=1}^{nt} \sum_{j=1}^{m} lt_i xt_{ij} < (1-\alpha) L_{max} \quad (8)$$

$$\sum_{j=1}^{m} xr_{ij} \le 1, for\ i = 1, \dots, nr \quad (9)$$

$$\sum_{j=1}^{m} xt_{ij} \le 1, for\ i = 1, \dots, nt \quad (10)$$

$$\sum_{b_k \in B_i} y_k \ge |B_i| \sum_{j=1}^{m} xr_{ij}, for\ i = 1, \dots, nr \quad (11)$$

$$\sum_{b_k \in B_i} y_k \ge |B_i| \sum_{j=1}^{m} xt_{ij}, for\ i = 1, \dots, nt \quad (12)$$

$$\sum_{sr_i \in Sr_k} \sum_{j=1}^{m} xr_{ij} + \sum_{st_i \in St_k} \sum_{j=1}^{m} xt_{ij} \ge y_k, \quad k = 1, \dots |B| \quad (13)$$

$$xr_{ij}, xt_{ij}, y_i \in \{0,1\} \quad (14)$$

All the three parts in the objective function are normalized to [0, 1] by using the maximum length L_{max} and the total number of bigrams $|B^*|$. λ_1, λ_2 and λ_3 are parameters for tuning the three parts and we set $\lambda_1+\lambda_2+\lambda_3 = 1$.

We explain the constraints as follows:

Constraint (6): It ensures that the total word count of the part of topic *j* does not exceed L_j.

Constraints (7), (8): The two constraints try to balance the lengths of the previous works part and the own work part, respectively. α is set to 2/3.

Constraints (9), (10): These two constraints guarantee that the sentence can only be included into one topic.

Constraints (11), (12): When these two constraints hold, all bigrams that s_i has are selected if s_i is selected.

Constraint (13): This constraint makes sure that at least one sentence in Sr_m or St_m is selected if bigram b_m is selected.

Therefore, we transform our optimization problem into a linear programing problem. We solve this linear programming problem by using the IBM CPLEX optimizer[5]. It generally takes tens of seconds to solve the problem and it is very efficient.

Finally, ARWG post-processes sentences to improve readability, including replacing agentive forms with a citation to the specific article (e.g., "our work" → "(Hoang and Kan, 2010)") for the sentences extracted from reference papers. The sentences belonging to different topics are placed separately.

5 Evaluation

5.1 Evaluation Setup

To set up our experiments, we divide our dataset which contains 1050 target papers and their reference papers into two parts: 700 target papers for training, 150 papers for test and the other 200 papers for validation. The PLSA topic model is applied to the whole dataset. We train two SVR regression models based on the own work part and the previous work part of the training data and apply the models to the test data. The global optimization framework is used to generate the related work sections. We set the maximum word count of the generated related work section to be equal to that of the gold related work section. The parameter values of λ_1, λ_2 and λ_3 are set to 0.3, 0.1 and 0.6, respectively. The parameter values are tuned on the validation data.

We compare our system with five baseline systems: MEAD-WT, LexRank-WT, ARWG-WT, MEAD and LexRank. MEAD[6] (Radev et al., 2004) is an open-source extractive multi-document summarizer. LexRank[7] (Eran and Radev, 2004) is a multi-document summarization system which is based on a random walk on the similarity graph of sentences. We also implement the MEAD, LexRank baselines and our method with only the reference papers (i.e. the target paper's content is not considered). Those methods are signed by "-WT".

To evaluate the effectiveness of the SVR models we employ, we implement a baseline system RWGOF that uses the random walk scores as the important scores of the sentences and take the scores as inputs for the same global optimization framework as our system to generate the related work section. The random walk scores are computed for the sentences in the reference papers and the target paper, respectively.

We use the ROUGE toolkit to evaluate the content quality of the generated related work sections. ROUGE (Lin, 2004) is a widely used automatic summarization evaluation method based on n-gram comparison. Here, we use the F-Measure scores of ROUGE-1, ROUGE-2 and ROUGE-SU4. The model texts are set as the gold related work sections extracted from the target papers, and word stemming is utilized. ROUGE-N is an n-gram based measure between a candidate text and a reference text. The recall oriented score, the precision oriented score and the F-measure score for ROUGE-N are computed as follows:

$$ROUGE - N_{Recall} = \sum_{S\in\{Reference\ Text\}}\sum_{gram_n} Count_{match}(gram_n) \,/\, \sum_{S\in\{Reference\ Text\}}\sum_{gram_n} Count(gram_n) \quad (15)$$

$$ROUGE - N_{Precision} = \sum_{S\in\{Reference\ Text\}}\sum_{gram_n} Count_{match}(gram_n) \,/\, \sum_{S\in\{Candidate\ Text\}}\sum_{gram_n} Count(gram_n) \quad (16)$$

$$ROUGE - N_{F-measure} = 2 * ROUGE - N_{Recall} * ROUGE - N_{Precision} \,/\, ROUGE - N_{Recall} + ROUGE - N_{Precision} \quad (17)$$

where n stands for the length of the n-gram $gram_n$, and $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate text and a reference text.

In addition, we conducted a user study to subjectively evaluate the related work sections to get more evidences. We selected the related work sections generated by different methods for 15 random target papers in the test set. We asked three human judges to follow an evaluation guideline we design and evaluate these related work sections. The human judges are graduate students in the computer science field and they did not know the identities of the evaluated related work sections. They were asked to give a rating on a scale of 1 (very poor) to 5 (very good) for the correctness, readability and usefulness of the related work sections, respectively:

[5] www-01.ibm.com/software/integration/optimization/cplex-optimizer/

[6] http://www.summarization.com/mead/

[7] In our experiments, LexRank performs much better than the more complex variant - C-LexRank (Qazvinian and Radev, 2008), and thus we choose LexRank, rather than C-LexRank, to represent graph-based summarization methods for comparison in this paper.

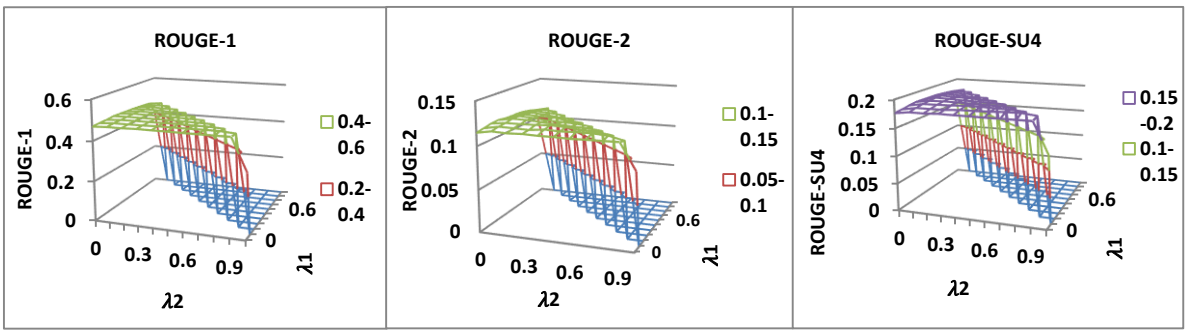


Figure 3: Parameter influences (horizontal, vertical axis are λ_1, λ_2 , respectively, $\lambda_3 = 1 - \lambda_1 - \lambda_2$)

1) Correctness: Is the related work section actually related to the target paper?
2) Readability: Is the related work section easy for the readers to read and grasp the key content?
3) Usefulness: Is the related work section useful for the author to prepare their final related work section?

Paired T-Tests are applied to both the ROUGE scores and rating scores for comparing ARWG and baselines and comparing the systems with WT and without WT.

5.2 Results and Discussion

Table 4: ROUGE F-measure comparison results

Method	ROUGE-1	ROUGE-2	ROUGE-SU4
Mead-WT	0.39720	0.08785	0.14694
LexRank-WT	0.43267	0.09228	0.16312
ARWG-WT	0.45077$^{*\{1,2\}}$	0.09987$^{*\{1,2\}}$	0.16731$^{*\{1\}\#\{2\}}$
Mead	0.41012$^{*\{1\}}$	0.09642$^{*\{1\}}$	0.15441$^{*\{1\}}$
LexRank	0.44235$^{*\{2\}}$	0.10090$^{*\{2\}}$	0.17067$^{*\{2\}}$
ARWG	**0.47940**$^{*\{1-5\}}$	**0.12176**$^{*\{1-5\}}$	**0.18618**$^{*\{1-5\}}$

(* represents pairwise t-test value $p < 0.01$; $^{\#}$ represents $p < 0.05$; the numbers in the brackets represent the indices of the methods compared, e.g. 1 for MEAD-WT, 2 for LexRank-WT, etc.)

Table 5: Average rating scores of judges

Method	Correctness	Readability	Usefulness
Mead	2.971	2.664	2.716
LexRank	2.958	2.847	2.784
ARWG	3.433$^{*\#}$	3.420$^{*\#}$	3.382$^{*\#}$

($^{*\#}$ represents pairwise t-test value $p < 0.01$, compared with Mead and LexRank, respectively.)

Table 6: ROUGE F-measure comparison of different sentence importance scores

Method	ROUGE-1	ROUGE-2	ROUGE-SU4
RWGOF	0.46932	0.11791	0.18426
ARWG	**0.47940**	**0.12176**	**0.18618**

The evaluation results over ROUGE metrics are presented in Table 4. It shows that our proposed system can get higher ROUGE scores, i.e., better content quality. In our system, we split the sentence set into different topic-biased parts, and the importance scores of sentences in the target paper and reference papers are learned differently. So the obtained importance scores of the sentences are more reliable.

The global optimization framework considers the extraction of both the previous work part and the own work part. We can see the importance of the own work part by comparing the results of the methods with or without considering the own work part. MEAD, LexRank and our method all get a significant improvement after considering the own work part by extracting sentences from the target paper. The results also prove our assumption about the related work section structure.

Figure 3 presents the fluctuation of ROUGE scores when tuning the parameters λ_1, λ_2 and λ_3. We can see our method generally performs better than the baselines. All the three parts in the objective function are useful to generate related work sections with good quality.

The average scores rated by human judges for each method are showed in Table 5. We can see that the related work sections generated by our system are more related to the target papers. Moreover, because of the good structure of our generated related work sections, our generated related work sections are considered more readable and more useful for the author to prepare the final related work sections.

T-test results show that the performance improvements of our method over baselines are statistically significant on both automatic and manual evaluations. Most of p-values for t-test are far smaller than 0.01.

Overall, the results indicate that our method can generate much better related work sections

than the baselines on both automatic and human evaluations.

Table 6 shows the comparison results between ARWG and RWGOF. We can see ARWG performs better than RWGOF. It proves that the SVR models can better estimate the importance scores of the sentences. For the SVR models are trained from the large dataset, the sentence scores predicted by the SVR models can be more reliable to be used in the global optimization framework.

6 Conclusion and Future Work

This paper proposes a novel system called ARWG to generate related work sections for academic papers. It first exploits a PLSA model to split the sentence set of the given papers into different topic-biased parts, and then applies regression models to learn the importance scores of the sentences. At last an optimization framework is proposed to generate the related work section. Evaluation results show that our system can generate much better related work sections than the baseline methods.

In future work, we will make use of citation sentences to improve our system. Citation sentences are the sentences that contains an explicit reference to another paper and they usually highlight the most important aspects of the cited papers. So citation sentences are likely to contain important and rich information for generating related work sections.

Acknowledgments

The work was supported by National Natural Science Foundation of China (61170166, 61331011), Beijing Nova Program (2008B03) and National Hi-Tech Research and Development Program (863 Program) of China (2012AA011101). We also thank the anonymous reviewers for very helpful comments. The corresponding author of this paper, according to the meaning given to this role by Peking University, is Xiaojun Wan.

Reference

Nitin Agarwal, Kiran Gvr, Ravi Shankar Reddy, and Carolyn Penstein Rosé. 2011. Towards multi-document summarization of scientific articles: making interesting comparisons with SciSumm. In *Proceedings of the Workshop on Automatic Summarization for Different Genres, Media, and Languages*, pp. 8-15. Association for Computational Linguistics.

Phyllis B. Baxendale. 1958. Machine-made index for technical literature: an experiment. *IBM Journal of Research and Development* 2, no. 4: 354-361.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 481-490. Association for Computational Linguistics.

Chih-Chung Chang, and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2, no. 3: 27.

John M. Conroy, and Dianne P. O'leary. 2001. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 406-407. ACM.

Harold P. Edmundson. 1969. New methods in automatic extracting. *Journal of the ACM (JACM)* 16, no. 2: 264-285.

Günes Erkan, and Dragomir R. Radev. 2004. LexPageRank: Prestige in Multi-Document Text Summarization. In *EMNLP*, vol. 4, pp. 365-371.

Günes Erkan, and Dragomir R. Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res.(JAIR)* 22, no. 1: 457-479.

Dimitrios Galanis, Gerasimos Lampouras, and Ion Androutsopoulos. 2012. Extractive Multi-Document Summarization with Integer Linear Programming and Support Vector Regression. In *COLING*, pp. 911-926.

Dimitrios Galanis, and Prodromos Malakasiotis. 2008. Aueb at tac 2008. In*Proceedings of the TAC 2008 Workshop*.

Dan Gillick, and Benoit Favre. 2009. A scalable global model for summarization. In*Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pp. 10-18. Association for Computational Linguistics.

Cong Duy Vu Hoang, and Min-Yen Kan. 2010. Towards automated related work summarization. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pp. 427-435. Association for Computational Linguistics.

Lei Huang, Yanxiang He, Furu Wei, and Wenjie Li. 2010. Modeling document summarization as multi-objective optimization. In *Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium on*, pp. 382-386. IEEE.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 50-57. ACM.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In*Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pp. 74-81.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development* 2, no. 2: 159-165.

Inderjeet Mani, and Eric Bloedorn. 1999. Summarizing similarities and differences among related documents. *Information Retrieval* 1, no. 1-2: 35-67.

Ryan McDonald. 2007. *A study of global inference algorithms in multi-document summarization*. Springer Berlin Heidelberg.

Qiaozhu Mei, and ChengXiang Zhai. 2008. Generating Impact-Based Summaries for Scientific Literature. In *ACL*, vol. 8, pp. 816-824.

Rada Mihalcea, and Paul Tarau. 2005. A language independent algorithm for single and multiple document summarization.

Rada Mihalcea, and Hakan Ceylan. 2007. Explorations in Automatic Book Summarization. In *EMNLP-CoNLL*, pp. 380-389.

Saif Mohammad, Bonnie Dorr, Melissa Egan, Ahmed Hassan, Pradeep Muthukrishan, Vahed Qazvinian, Dragomir Radev, and David Zajic. 2009. Using citations to generate surveys of scientific paradigms. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 584-592. Association for Computational Linguistics.

Preslav Nakov, Ariel Schwartz, and M. Hearst. 2004. Citation sentences for semantic analysis of bioscience text. In *Proceedings of the SIGIR'04 workshop on Search and Discovery in Bioinformatics*.

Ramesh M. Nallapati, Amr Ahmed, Eric P. Xing, and William W. Cohen. 2008. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 542-550. ACM.

Vahed Qazvinian, and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pp. 689-696. Association for Computational Linguistics.

You Ouyang, Sujian Li, and Wenjie Li. 2007. Developing learning strategies for topic-based summarization. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 79-86. ACM.

Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Celebi, Stanko Dimitrov, Elliott Drabek et al. 2004. MEAD-a platform for multidocument multilingual text summarization. Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004).

Ariel S. Schwartz, and Marti Hearst. 2006. Summarizing key concepts using citation sentences. In *Proceedings of the Workshop on Linking Natural Language Processing and Biology: Towards Deeper Biological Literature Analysis*, pp. 134-135. Association for Computational Linguistics.

Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. 2007. Document Summarization Using Conditional Random Fields. In *IJCAI*, vol. 7, pp. 2862-2867.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics* 26, no. 3: 339-373.

Kristian Woodsend, and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 233-243. Association for Computational Linguistics.

Shasha Xie, Benoit Favre, Dilek Hakkani-Tür, and Yang Liu. 2009. Leveraging sentence weights in a concept-based optimization framework for extractive meeting summarization. In *INTERSPEECH*, pp. 1503-1506.

Hui Yang, and Jamie Callan. 2009. A metric-based framework for automatic taxonomy induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pp. 271-279. Association for Computational Linguistics.

Ozge Yeloglu, Evangelos Milios, and Nur Zincir-Heywood. 2011. Multi-document summarization of scientific corpora. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pp. 252-258. ACM.

Fast and Accurate Misspelling Correction in Large Corpora

Octavian Popescu
Fondazione Bruno Kessler
Trento, Italy
popescu@fbk.eu

Ngoc Phuoc An Vo
University of Trento
Fondazione Bruno Kessler
Trento, Italy
ngoc@fbk.eu

Abstract

There are several NLP systems whose accuracy depends crucially on finding misspellings fast. However, the classical approach is based on a quadratic time algorithm with 80% coverage. We present a novel algorithm for misspelling detection, which runs in constant time and improves the coverage to more than 96%. We use this algorithm together with a cross document coreference system in order to find proper name misspellings. The experiments confirmed significant improvement over the state of the art.

1 Introduction

The problem of finding the misspelled words in a corpus is an important issue for many NLP systems which have to process large collections of text documents, like news or tweets corpora, digitalized libraries etc. Any accurate systems, such as the ones developed for cross document coreference, text similarity, semantic search or digital humanities, should be able to handle the misspellings in corpora. However, the issue is not easy and the required processing time, memory or the dependence on external resources grow fast with the size of the analyzed corpus; consequently, most of the existing algorithms are inefficient. In this paper, we present a novel algorithm for misspelling detection which overcomes the drawbacks of the previous approaches and we show that this algorithm is instrumental in improving the state of the art of a cross document coreference system.

Many spelling errors in a corpus are accidental and usually just one or two letters in a word are affected, like *existnece* vs. the dictionary form *existence*. Such misspellings are rather a unique phenomenon occurring randomly in a text. For an automatic speller which has access to a dictionary, finding and compiling a list of correct candidates for the misspelled words like the one above is not very difficult. However, not all misspellings are in this category. To begin with, proper nouns, especially foreign proper names, are not present in the dictionary and their misspelling may affect more than one or two characters. Moreover, the misspelling of proper names may not be random, for example there might be different spellings of the same Chinese or Russian name in English, the incorrect ones occurring with some frequency. Also, especially if the corpus contains documents written by non native speakers, the number of characters varying between the correct and the actual written form may be more than two. In this case, finding and compiling the list of correct candidates is computationally challenging for traditional algorithms, as the distance between the source string and the words in the candidates list is high.

The Levenshtein distance has been used to compile a list of correct form candidates for a misspelled word. The Levenshtein distance between two strings counts the number of changes needed to transform one string into the other, where a change is one of the basic edit operations: deletion, insertion, substitution of a character and the transposition of two characters. The Edit Distance algorithm, (ED) computes the similarity between two strings according to the Levenshtein distance. Most of the random misspellings which are produced by a native speaker are within one or maximum two basic edit operations (Damerau, 1964). For this reason the ED algorithm is the most common way to detect and correct the misspellings. However, there is a major inconvenience associated with the use of ED, namely, ED

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1634–1642,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

runs in quadratic time considering the length of the strings, O(n^2). The computation time for more than a few thousands pairs is up to several tens of seconds, which is impracticably large for most of large scale applications. By comparison, the number of proper names occurring in a medium sized English news corpus is around 200, 000, which means that there are some 200, 000, 000 pairs.

In order to cope with the need for a lower computation time, on the basis of ED, a series of algorithms have been developed that run in linear time (Navaro 2001). Unfortunately, this improvement is not enough for practical applications which involve a large amount of data coming from large corpora. The reason is two-fold: firstly, the linear time is still too slow (Mihov and Schulz, 2004) and secondly, the required memory depends both on the strings' length and on the number of different characters between the source string and the correct word, and may well exceed several GBs. Another solution is to index the corpus using structures like trie trees, or large finite state automata. However, this solution may require large amounts of memory and is inefficient when the number of characters that differ between the source string and the candidate words is more than two characters (Boytsov, 2011).

We focus specifically on misspellings for which there is no dictionary containing the correct form and/or for which the Levenshtein distance to the correct word may be higher than two characters. For this purpose, we developed a novel approach to misspelling correction based on a non indexing algorithm, which we call the prime mapping algorithm, PM. PM runs in constant time, O(1), with insignificant memory consumption. The running time of the PM algorithm does not depend either on the strings' length or on the number of different characters between the source string and the candidate word. It requires a static amount of memory, ranging from a few KBs to a maximum of a few MBs, irrespective of the size of the corpus or the number of pairs for which the misspelling relationship is tested. We run a series of experiments using PM on various corpora in English and Italian. The results confirm that PM is practical for large corpora. It successfully finds the candidate words for misspellings even for large Levenshtein distances, being more than 30 times faster than a linear algorithm, and several hundred times faster than ED. The running time difference is due to the fact that PM maps the strings into numbers and performs only one arithmetic operation in order to decide whether the two strings may be in a misspelling relationship. Instead of a quadratic number of characters comparisons, PM executes only one arithmetic operation with integers.

We also report here the results obtained when using PM inside a cross document coreference system for proper nouns. Correcting a proper name misspelling is actually a more complex task than correcting a misspelled common word. Some misspellings may not be random and in order to cope with repetitive misspellings, as the ones resulting from the transliteration of foreign names, the PM is combined with a statistical learning algorithm which estimates the probability of a certain type of misspelling considering the surrounding characters in the source string. Unlike with common words, where a misspelling is obvious, in the case of proper names, *John* vs. *Jon* for example, it is unclear whether we are looking at two different names or a misspelling. The string similarity evidence is combined with contextual evidence provided by a CDC system to disambiguate.

To evaluate the PM algorithm we use publicly available misspelling annotated corpora containing documents created by both native and non-native speakers. The PM within a CDC system for proper names is evaluated using CRIPCO (Bentivogli et al., 2008). The experiments confirm that PM is a competitive algorithm and that the CDC system gains in accuracy by using a module of misspelling correction.

The rest of the paper is organized as follows. In Section 2 we review the relevant literature. In Section 3 we introduce the PM algorithm and compare it against other algorithms. In Section 4 we present the CDC system with misspelling correction for proper names. In Section 5 we present the results obtained on English and Italian corpora.

2 Related Work

In a seminal paper (Damerau, 1964) introduced the ED algorithm. The rationale for this algorithm was the empirical observation that about 80% of the misspelled words produced by native speakers have distance 1 to the correct word. ED cannot be extended to increase the accuracy, because for k = 2, k being the maximal admissible distance to the correct word, the running time is too high. Most of the techniques developed further use ED together with indexing methods and/or parallel processing.

In (San Segundo et al., 2001) an M-best can-

didate HMM recognizer for 10,000 Spanish city names is built for speech documents. An N-gram language model is incorporated to minimize the search spaces. A 90% recognition rate is reported. The model is not easily generalizable to the situation in which the names are unknown - as it is the case with the personal proper names in a large corpus. The N-gram model is memory demanding and for 200,000 different names the dimension of the requested memory is impracticably big.

The problem related to personal proper names was discussed in (Allan and Raghavan, 2002). However, the paper addresses only the problem of clustering together the names which "sound alike" and no cross document coreference check was carried out. The technique to find similar names is based on a noisy channel model. The conditional probabilities for each two names to be similarly spelled are computed. The time complexity is quadratic, which renders this technique unfeasible for big data. In fact, the results are reported for a 100 word set. A different approach comes from considering search queries databases (Bassil and Alwani, 2012). These techniques are similar to the model based on the noisy channel, as they compute the conditional probabilities of misspellings based on their frequencies in similar queries. Unfortunately, large numbers of queries for proper names are not available. A similar technique, but using morphological features, was presented in (Veronis, 1988). The method can manage complex combinations of typographical and phonographic errors.

It has been noted in many works dedicated to error correction, see among others (Mihov and Schulz, 2004), that the ED algorithm is impracticably slow when the number of pairs is large. A solution is to build a large tries tree. While this solution improves the searching time drastically, the memory consumption may be large. Automata indexing was used in (Oflazer, 1996). While the memory consumption is much less than for the tries tree approaches, it is still high. For Turkish, the author reported 28,825 states and 118,352 transitions labeled with surface symbols. The recovery error rate is 80%. In (Boytsov, 2011) a review of indexing methods is given. Testing on 5,000 strings for k=1,2,3 is reported and the paper shows the problem the systems run into for bigger values of k. In (Huldén, 2009) a solution employing approximations via an A* strategy with finite automata is presented. The method is much faster for k bigger than the one presented in (Chodorow and Leacock, 2000). However, the usage of A* for proper names may be less accurate than the one reported in the paper, because unlike the common words in a given language, the names may have unpredictable forms, especially the foreign names. The results reported show how the time and memory vary for indexing methods according to the length of the words for k=1,2,3.

A method that uses mapping from strings to numbers is presented in (Reynaert, 2004). This method uses sum of exponentials. The value of the exponential was empirically found. However, the mapping is only approximative. Our mapping is precise and does not use exponential operations which are time consuming.

The study in (Navarro, 2001) is focused on non indexing approximate string search methods, in particular on the simple ED distance. The non-indexing methods may reach linear running time, but it is not always the case that they are scalable to big data. In (Nagata et al., 2006) a study on the type of errors produced by non-native speakers of English is carried out, but the long distance misspellings are not considered.

3 Prime Mapping Misspeling Algorithm

The algorithms based on the Levenshtein distance use the dynamic programming technique to build a table of character to character comparisons. We present here a novel approach to misspelling which does not build this table, skipping the need to compare characters. In a nutshell, the prime mapping algorithm, PM, replaces the characters compare operations to a unique arithmetic operation.This can be done by associating to any letter of the alphabet a unique prime number. For example we can associate 2 to a, 3 to b, 5 to c ... 97 to z. Any string will be mapped into a unique number which is the product of the prime numbers corresponding to its letters. For example the name $abba$ is mapped to $2 \cdot 3 \cdot 3 \cdot 2 = 36$. By computing the ratio between any two words we can detect the different letters with just one operation. For example, the difference between $abba$ and aba is 36/12 = 3, which corresponds uniquely to b because the product/ratio of prime numbers is unique.

Unlike the ED algorithm, the prime mapping does not find the number of edit operations needed to transform one string into another. In fact, two words that have just one letter in the mutual difference set may be quite distinct: all the strings

aba, aab, baa differ by one letter when compared with $abba$. In order to be in a misspelling relationship, the two strings should also have a common part, like prefix or middle, or suffix. The complete Prime Mapping (PM) algorithm consists of two successive steps: (1) find all the candidate words that differ from the target word by at most *k* characters and (2) check weather the target word and the candidate word have a common part, suffix, prefix or middle part. Both steps above are executed in constant time, therefore they do not depend either on the length of the strings or on k, the maximal number of different characters. Normally, $k = 3$, because the probability of a misspelled word having more than three distinct letters is insignificant, but unlike in the case of ED, the choice of k has no influence on the running time. The first step takes an integer ratio and a hash table key check, both being O(1). The second step checks if the first k letters at the beginning or at the end of the word are the same, and it requires $2k$ character comparisons, which is also an O(1) process, as *k* is fixed. The pseudo code and detailed description of the PM algorithm are given below.

Algorithm 1 Prime Mapping

Require: $charList$ $wordsList$, $primeList$, k
Ensure: $misspList$
1: $misspList \leftarrow \emptyset$
2: **foreach** α **in** $charList$: p(α) $\leftarrow$ p_i, p_i **in** $primeList$
3: **foreach** w **in** $wordsList$: p(w) $\leftarrow \prod p(\alpha)$, α **in** w
4: $primeKTable \leftarrow \binom{n}{k}$ of prime arithmetics
5: **for** w **in** $wordsList$ **do**
6: **for** w' **in** $wordsList$, w $\neq$ w' **do**
7: r $\leftarrow \frac{p(w)}{p(w')}$
8: **if** r **in** $primeKTable$ **then**
9: **if** commonPart (w, w') $\neq \emptyset$ **then**
10: $misspList \leftarrow misspList$ + (w, w')
11: **end if**
12: **end if**
13: **end for**
14: **end for**

map letters to prime numbers. A helpful way to assign primes to letters is according to their frequency; on average, the numbers corresponding to names are smaller and the operation gets less time.

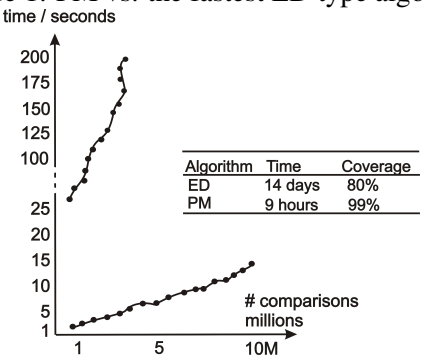
compute a hash table with prime arithmetics of K primes. In the hash table *primeKTable* we record all the combinations that can result from dividing two products which have less than *k* primes: $1/p_i$, p_i, p_i/p_j etc. If the ratio between two mappings is in the hash table, then the corresponding words have all the letters in common, except for at most *k*. The number of all the combination is $\binom{n}{k}$. The memory consumption for different values for *k* is given in Table 1. The figures compare extremely favorably with the ones of ED based approaches (gigs magnitude) . (line 7-8)

k letter difference	*#combination*	*Memory*
1	60	480B
2	435	8K
5	142,506	0.9MB
6	593, 775	3.8MB
10	30, 045, 015	180MB

Table 1: The PM algorithm memory needs

find misspelling candidates by ratio. By computing the ratio and by checking the hash table, we found the pairs which use the same letters, except for at most *k*. The procedure *commonpart* checks whether the two strings also have a common part by looking at the start and end *k*. If this is the case, the pair is in a misspelling relationship.

Figure 1: PM vs. the fastest ED type algorithm



The PM is much faster than ED. The fastest variant of ED, which does not compare strings having length difference bigger than 1, theoretically finds only 80% of the misspellings. In practice, only around 60% of the misspellings are found because of proper names and words misspelled by non-native speakers. The PM algorithm considers all possible pairs, finds more than 99% of misspellings and is 35 times faster. To obtain the same coverage, the ED algorithm must run for more than 100 days. The time comparison for millions of pairs is plotted in Figure 1. The experiments were performed on an i5, 2.8 GHz processor.

There is an immediate improvement we can bring to the basic variant of PM. The figures reported above are obtained by doing the whole set of possible pairs. By taking into account the fact that two words differing by $k + 1$ letters cannot be k similar, we can organize the number representing the names into an array which reduced drasti-

cally the number of comparisons. For example, all the words containing the letters *x, y, z* cannot be $k = 2$ similar with the words not containing any of these letters. By dividing the mapping of a word to the primes associated with the letters of an k-gram, we know if the words containing the k-gram can be misspelling candidates with at most k difference, and there is no more need to carry out all the ratios. We arrange the mappings of all words into an array such that on the first indexes we have the words containing the less frequent $k + 1$ gram, on the next indexes we have the words containing the second less frequent $k+1$ gram and do not contain the first $k + 1$ gram, on the next indexes the words containing the third less frequent $k + 1$ gram and do not contain the first two $k + 1$ gram, etc. In this way, even the most frequent $k + 1$ gram has only a few words assigned and consequently the number of direct comparisons is reduced to the minimum. The mapping corresponding to a $k+1$ gram are ordered in this array according to the length of the words. The number of trigrams is theoretically large, the $k + 1$ power of the size of the alphabet. However, the number of actually occurring k-trigrams is only a small fraction of it. For example, for $k = 2$, the number of trigrams is a few hundred, out of the 2, 700 possible ones. PM2gram runs in almost a quarter of the time needed by the basic PM. For the same set of names we obtained the results reported in Table 2. The last column indicates how many times the algorithm is slower than the PM in its basic form.

algorithm	*time*	*coverage*	*times slower*
basicED	132 days	99%	310
ED1	14 days	80%	35
PM	9 hours	99%	1
PM2gram	2 hours 42min	96%	0.26

Table 2: ED variants versus MP

4 Correcting Proper Names Misspellings

In this section we focus on a class of words which do not occur in a priorly given dictionary and for which the misspelled variants may not be random. Proper names are representative for this class. For example, the same Russian name occurs in corpus as *Berezovski*, *Berezovsky* or *Berezovschi* because of inaccurate transliteration. By convention, we consider the most frequent form as the canonical one, and all the other forms as misspelled variants.

Many times, the difference between a canonical form and a misspelled variant follows a pattern: a particular group of letters substitutes another one in the context created by the other characters in the name. A misspelling pattern specifies the context, as prefix or suffix of a string, where a particular group of characters is a misspelling of another. See Table 3 for examples of such patterns.

Pattern	*Context*	*Example*
dj→dji	ovic	djiukanovic djukanovic
k→kh	aler	kaler khaler, taler thaler
ki→ky	ovsk	berezovski berezovsky
n→ng	chan	chan-hee chang-hee
dl→del	abd	abdelkarim abdlkrim

Table 3: Name misspellings patterns

Finding and learning such patterns, along with their probability of indicating a true misspelling, bring an important gain to CDC systems both in running time and in alleviating the data-sparseness problem. The CDC system computes the probability of coreference for two mentions t and t' using a similarity metrics into a vectorial space, where vectors are made out of contextual features occurring with t and t' respectively (Grishman, 1994). However, the information extracted from documents is often too sparse to decide on coreference (Popescu, 2009). Coreference has a global effect, as the CDC systems generally improve the coverage creating new vectors by interpolating the information resulting from the documents which were coreferred (Hastie et al., 2005). This information is used to find further coreferences that no single pair of documents would allow. Thus, missing a coreference pair may result in losing the possibility of realizing further coreferences. However, for two mentions matching a misspelling pattern which is highly accurate, the threshold for contextual evidence is lowered. Thus, correcting a misspelling is not beneficial for a single mention only, but for the accuracy of the whole.

The strategy we adopt for finding patterns is to work in a bootstrapping manner, enlarging the valid patterns list while maintaining a high accuracy of the coreference, over 90%. Initially, we start with an empty base of patterns. Considering only the very high precision threshold for coreference, above 98% certainty, we obtain a set of misspelling pairs. This set is used to extract patterns of misspellings via a parameter estimation found using the EM-algorithm. The pattern is considered valid only if it also has more than a given number of occurrences. The recursion of the previous steps is carried out by lowering with an ε the threshold for accuracy of coreference for pat-

tern candidates. The details and the pseudo code are given below.

Algorithm 2 Misspelling Pattern Extraction

Require: $thCoref$, ε, $minO$, $thAcc$
Require: $thCDC$
Ensure: $pattList$
1: $pattList, candPattList \leftarrow \emptyset$
2: **while** there is a pair (t, t') to test for coreference **do**
3: **if** (t, t') *matches* p, p **in** $pattList$ **then**
4: prob ← corefProb(p)
5: **else**
6: use PM algorithm on pair (t, t')
7: prob ← $thCoref$
8: **end if**
9: **if** pair (t, t') coref with prob **then**
10: $candPattList \leftarrow candPattList$ + (t, t')
11: **end if**
12: extractPatterns from $candPattList$
13: **for** cp **in** $new\ extracted\ patterns$ **do**
14: **if** #cp>$minO$ **and** corefProb(cp)>$thAcc$ **then**
15: $pattList \leftarrow pattList$ + (t, t')
16: **end if**
17: **end for**
18: **if** prob>$thCDC$ **then**
19: corefer (t, t')
20: **end if**
21: **end while**
22: $thCoref \leftarrow thCoref$ - ε
23: **goto** line 2

1. Compile a list of misspelling candidates For each source string, t, try to match t against the list of patterns (initially empty). If there is a pattern matching (t, t') then their prior probability of coreference is the probability associated with that pattern (line 4).

2. CDC coreference evidence For each pair (t ,t') in the canonical candidates list use the CDC system to compute the probability of coreference between t and t'. If the probability of coreference of t and t' is higher than $thCoref$, the default value is 98%, then consider t as a misspelling of t' and put (t, t') in a candidate pattern list (line 10).

3. Extract misspelling patterns Find patterns in the candidate pattern list. Consider only patterns with more than $minO$ occurrences, whose default value is 10, and which have the probability of coreference higher than $thAcc$, whose default value is 90% (line 15).

4. CDC and pattern evidence For each (t,t') pair matching a pattern and the CDC probability of coreference more then $thCDC$, whose default value is 80%, then corefer t and t' (line 21). The fact that the pair (t,t') matches a pattern of misspelling is considered supporting evidence for coreference and in this way it plays a direct role in enhancing the system coverage. Decrease $thCoref$ by ε,whose default is value 0.5, and repeat the process of finding patterns (goto line 2).

To extract the pattern from a given list of pairs, procedure extractPatterns at line 12 above, we generate all the suffixes and prefixes of the strings. We compute the probability that a group of characters represents a spelling error, given a certain suffix and/or prefix. We use the EM algorithm to compute these probabilities. For a pair (P, S) of a prefix and a suffix, the tuples (p(P)=p, p(S)=s, π) are the quantities to be estimated via EM, with π being the coreference probability. A coreference event is directly observable, without knowing, however, which prefix or suffix contribute to the coreference. The EM equations are given below, where X is the observed data; Z are the hidden variable, p and s respectively; θ the parameters (p,s, π); Q(θ,$\theta^{(t)}$) the expected log likelihood at iteration t.

$\mathbf{E-step}$ $\mu_i^{(t)}$

$$\begin{aligned}\mu_i^{(t)} &= E[z_i|x_i,\theta^{(t)}] \\ &= \frac{p(x_i|z_i,\theta^{(t)})\ p(z_i=P|\theta^{(t)})}{p(x_i|\theta^{(t)})} \\ &= \frac{\pi^{(t)}[p^{(t)}]^{x_i}[(1-p^{(t)}]^{(1-x_i)}}{\pi^{(t)}[p^{(t)}]^{x_i}[(1-p^{(t)}]^{(1-x_i)}+(1-\pi^{(t)})[s^{(t)}]^{x_i}[(1-s^{(t)}]^{(1-x_i)}}\end{aligned} \quad (1)$$

$\mathbf{M-step}$ $\theta^{(t+1)}$

$$\begin{aligned}&\frac{\partial Q(\theta|\theta^t)}{\partial \pi}=0 \quad \pi^{(t+1)}=\frac{\sum_i \mu_i^{(t)}}{n} \\ &\frac{\partial Q(\theta|\theta^t)}{\partial p}=0 \quad p^{(t+1)}=\frac{\sum_i \mu_i^{(t)} x_i}{\sum_i \mu_i^{(t)}} \\ &\frac{\partial Q(\theta|\theta^t)}{\partial s}=0 \quad s^{(t+1)}=\frac{\sum_i (1-\mu_i^{(t)}) x_i}{\sum_i (1-\mu_i^{(t)})}\end{aligned} \quad (2)$$

5 Experiments

We performed a set of experiments on different corpora in order to evaluate: (1) the performances of the PM algorithm for misspelling detection, (2) the accuracy of proper name misspelling pattern acquisition from large corpora, and (3) the improvements of a CDC system, employing a correction module for proper name misspellings.

In Section 5.1 the accuracy of the PM algorithm is tested on various corpora containing annotated misspellings of English words. In particular, we were interested to see the results when the edit distance between the misspelled pair is bigger than 3, because handling bigger values for k is crucial for finding misspelling errors produced by non-native speakers. The evaluation is directly relevant for the correction of the spelling of foreign names.

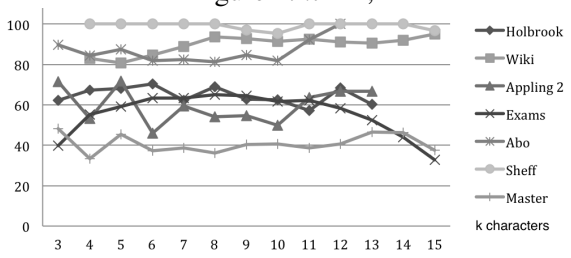
In Section 5.2 the proper name misspelling patterns were extracted from two large news corpora. One corpus is part of the English Gigawords, LDC2009T13 (Parker et al., 2009) and the second corpus is Adige500k in Italian (Magnini et al., 2006). We use a Named Entity Recognizer which has an accuracy above 90% for proper names. We evaluated the accuracy of the patterns by random sampling.

In Section 5.3 the accuracy of the CDC system with the correction module for proper name misspellings was tested against a gold standard.

5.1 PM Evaluation

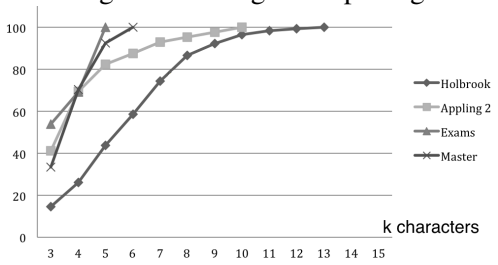
We consider the following publicly available English corpora containing the annotation of the misspelled words: Birkbeck, Aspell, Holbrook, Wikipidia. Birkbeck is a heterogeneous collection of documents, so in the experiments below we refer to each document separately. In particular we distinguish between misspellings of native speakers vs. misspelling of non-native speakers. Figure 2 shows that there are two types of corpora. For the first type, the misspellings found within two characters are between 80% and 100% of the whole number of misspellings. For the second type, less than 50% of the misspellings are within two characters.The second category is represented by the misspellings of non native speakers. The misspellings are far from the correct forms and they represent chunks of phonetically similar phonemes, like *boiz* vs. *boys*. The situation of the foreign name misspellings is likely to be similar to the misspellings found in the second type of corpora. For those cases, handling a k value bigger than 2 is crucial. Not only the

Figure 2: k = 1, 2



non-indexing methods, but also indexing ones are rather inefficient for k values bigger than 2 for large corpora. The PM algorithm does not have this drawback, and we tested the coverage of the errors we found for values of k ranging from 3 to 10. In Figure 3 we plot the distributions for the

Figure 3: Foreign Misspellings



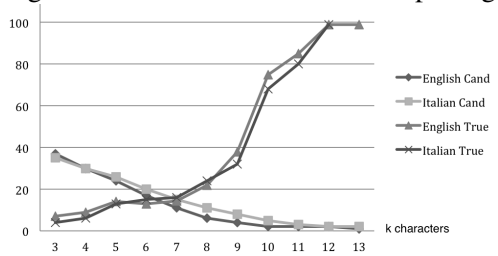
corpora which are problematic for k=2. Values of k are plotted on the OX axis, and the percentage of the misspellings within the respective k on the OY axis. The results showed PM is also able to find the phonemically similar misspellings. We can see that for k bigger than 9 the number of misspellings is not significant.

The PM algorithm performed very well, being able to find the misspellings even for large k values. There were 47, 837 words in Aspell, Holbrrok and Wikipedia, and 30, 671 in Birkbeck, and PM found all the misspelling pairs in a running time of 25 minutes. This is a very competitive time, even for indexing methods. For k above 8 the access to the hash table containing the prime combinations was slower, but not significantly so.

5.2 Pattern Extraction Evaluation

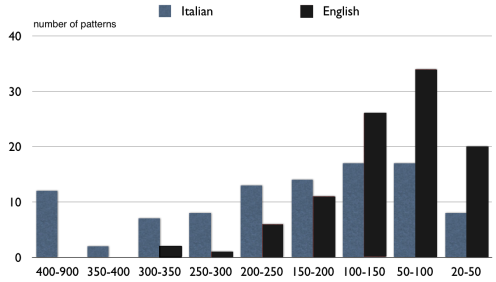
We extracted the set of names using a NER from the two corpora, LDC2009T13 and Adige500k. The set of proper names is rather large in both corpora - 160, 869 names from the English corpus and 185, 508 from the Italian corpus. Apparently, the quasi-similar names, which are considered as misspelled name candidates, is very high. In Figure 4 we plot this data. The *English Cand* and *Italian Cand* are absolute values, while the *English True* and *Italian True* represent percentages. For example, a name of length 5 is likely to have around 23 misspelling candidates, but only 17% of them are likely to be true misspellings, the rest being different names.

Figure 4: Candidates vs. True Misspellings



The numbers are estimated considering samples having the size between 30 and 50, for each name length. The percentages change rapidly with the length of the string. For names with the length bigger than 11, the probability that a misspelling candidate is a true misspelling is more than 98%. This fact suggests a strategy for pattern extraction: start from the higher name length towards the lower length names. The patterns found by the algorithm described in Section 4 have between 900 and 20 occurrences. There are 12 patterns having more than 400 occurrences, 20 having between 20 and 50 occurrences, see Fig. 5.

Figure 5: Distribution of the patterns:

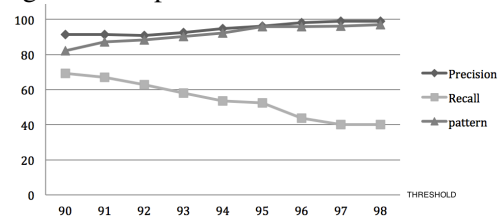


5.3 CDC and Misspelling correction

The CRIPCO corpus (Bentivogli et al., 2008) is a gold standard for CDC in Italian, containing pieces of news extracted from Adige500k. There are 107 names, the majority being Italian names. We scrambled the names to create misspelling candidates. For example the name $leonardo$ was scrambled like $teonardo$, $lionaldo$, $loenarod$ etc. We considered the top 15 frequency letters and maximum 4 letters for each scrambling. We randomly selected 70% of the original CRIPCO making no modifications, and called this corpus CRwCR. 30% of the original documents were assigned to the invented pseudo-names, and we called this corpus CRwSC (correct documents with scrambled names). From Adige500k we randomly chose 20, 000 documents and assigned them to the scrambled names as well, calling this corpus NCRwSC. From these pieces we created a new corpus: 70% of the initial CRIPCO documents with the original names, 30% of the CRIPCO documents with scrambled names and 20, 000 documents with the same scrambled names. For the names occurring in CRwCR, the scrambled names are valid name misspellings in the CRwSC corpus, and invalid in NCRwSC.

As expected, the PM algorithm found all the misspelling candidates and some others as well. We let the threshold confidence of coreference to vary from 90% to 98%. The number in Figure 6 refers to the precision and recall for the name misspellings in the CRIPCO corpus created via random scrambling. We were also interested to see how the pattern finding procedure works, but scrambling randomly produced too many contexts. Therefore, we chose to modify the names in a non random way, by replacing the final o to ino, ex. $paolo$ goes to $paolino$, and modifying one letter in the word for half of the occurrences, ex. $paorino$. The idea is that ino is a very common suffix for names in Italian. The system was able to learn the pseudo alternatives created in the context ino. The noise introduced was relatively low, see Fig. 6.

Figure 6: Proper Names CRIPCO Evaluation



6 Conclusion and Further Research

In this paper we described a system able to correct misspellings, including proper name misspellings, fast and accurately. The algorithm introduced, PM, overcomes the time/memory limitations of the approaches based on the edit distance.

The system is built on a novel string compare algorithm which runs in constant time independently of the length of the names or the number of different letters allowed, with no auxiliary memory request. As such, the algorithm is much faster than any other non-indexing algorithms. Because it is independent of k, it can be used even for large k, where even the indexing methods have limitations. We also used an EM based technique to find misspelling patterns. The results obtained are very accurate.

The system makes a first selection of the documents, drastically reducing the human work load. Another line of future research is to use the PM algorithm in other NLP tasks, where finding the pairs having some particular elements in common is necessary: for example, comparing parsing trees or dependency trees. We think that PM can be used in other NLP tasks as well and we hope the community can take advantage of it.

References

James Allan and Hema Raghavan. 2002. Using Part-of-Speech Patterns to Reduce Query Ambiguity. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 307–314. ACM.

Youssef Bassil and Mohammad Alwani. 2012. OCR Post-Processing Error Correction Algorithm Using Google's Online Spelling Suggestion. *Journal of Emerging Trends in Computing and Information Sciences, ISSN 2079-8407, Vol. 3, No. 1*.

Luisa Bentivogli, Christian Girardi, and Emanuele Pianta. 2008. Creating a Gold Standard for Person Cross-Document Coreference Resolution in Italian News. In *The Workshop Programme*, page 19.

Leonid Boytsov. 2011. Indexing Methods for Approximate Dictionary Searching: Comparative Analysis. *Journal of Experimental Algorithmics (JEA)*, 16:1–1.

Martin Chodorow and Claudia Leacock. 2000. An Unsupervised Method for Detecting Grammatical Errors. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 140–147. Association for Computational Linguistics.

Fred J Damerau. 1964. A Technique for Computer Detection and Correction of Spelling Errors. *Communications of the ACM*, 7(3):171–176.

Ralph Grishman. 1994. Whither Written Language Evaluation? In *Proceedings of the workshop on Human Language Technology*, pages 120–125. Association for Computational Linguistics.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. 2005. The Elements of Statistical Learning: Data Mining, Inference and Prediction. *The Mathematical Intelligencer*, 27(2):83–85.

Måns Huldén. 2009. Fast Approximate String Matching with Finite Automata. *Procesamiento del lenguaje natural*, 43:57–64.

Bernardo Magnini, Emanuele Pianta, Christian Girardi, Matteo Negri, Lorenza Romano, Manuela Speranza, Valentina Bartalesi Lenzi, and Rachele Sprugnoli. 2006. I-CAB: The Italian Content Annotation Bank. In *Proceedings of LREC*, pages 963–968.

Stoyan Mihov and Klaus U Schulz. 2004. Fast Approximate Search in Large Dictionaries. *Computational Linguistics*, 30(4):451–477.

Ryo Nagata, Koichiro Morihiro, Atsuo Kawai, and Naoki Isu. 2006. A Feedback-Augmented Method for Detecting Errors in The Writing of Learners of English. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 241–248. Association for Computational Linguistics.

Gonzalo Navarro. 2001. A Guided Tour to Approximate String Matching. *ACM computing surveys (CSUR)*, 33(1):31–88.

Kemal Oflazer. 1996. Error-tolerant Finite-state Recognition with Applications to Morphological Analysis and Spelling Correction. *Computational Linguistics*, 22(1):73–89.

Robert Parker, Linguistic Data Consortium, et al. 2009. *English Gigaword Fourth Edition*. Linguistic Data Consortium.

Octavian Popescu. 2009. Person Cross Document Coreference with Name Perplexity Estimates. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 997–1006. Association for Computational Linguistics.

Martin Reynaert. 2004. Text Induced Spelling Correction. In *Proceedings of the 20th international conference on Computational Linguistics*, page 834. Association for Computational Linguistics.

Rubén San Segundo, Javier Macías Guarasa, Javier Ferreiros, P Martin, and José Manuel Pardo. 2001. Detection of Recognition Errors and Out of the Spelling Dictionary Names in a Spelled Name Recognizer for Spanish. In *INTERSPEECH*, pages 2553–2556.

Jean Veronis. 1988. Morphosyntactic Correction in Natural Language Interfaces. In *Proceedings of the 12th conference on Computational linguistics-Volume 2*, pages 708–713. Association for Computational Linguistics.

Assessing the Impact of Translation Errors on Machine Translation Quality with Mixed-effects Models

Marcello Federico, Matteo Negri, Luisa Bentivogli, Marco Turchi
FBK - Fondazione Bruno Kessler
Via Sommarive 18, 38123 Trento, Italy
{federico,negri,bentivogli,turchi}@fbk.eu

Abstract

Learning from errors is a crucial aspect of improving expertise. Based on this notion, we discuss a robust statistical framework for analysing the impact of different error types on machine translation (MT) output quality. Our approach is based on linear mixed-effects models, which allow the analysis of error-annotated MT output taking into account the variability inherent to the specific experimental setting from which the empirical observations are drawn. Our experiments are carried out on different language pairs involving Chinese, Arabic and Russian as target languages. Interesting findings are reported, concerning the impact of different error types both at the level of human perception of quality and with respect to performance results measured with automatic metrics.

1 Introduction

The dominant statistical approach to machine translation (MT) is based on learning from large amounts of parallel data and tuning the resulting models on reference-based metrics that can be computed automatically, such as BLEU (Papineni et al., 2001), METEOR (Banerjee and Lavie, 2005), TER (Snover et al., 2006), GTM (Turian et al., 2003). Despite the steady progress in the last two decades, especially for few well resourced translation directions having English as target language, this way to approach the problem is quickly reaching a performance plateau. One reason is that parallel data are a source of reliable information but, alone, limit systems knowledge to observed positive examples (*i.e.* how a sentence should be translated) without explicitly modelling any notion of error (*i.e.* how a sentence should *not* be translated). Another reason is that, as a development and evaluation criterion, automatic metrics provide a holistic view of systems' behaviour without identifying the specific issues of a translation. Indeed, the global scores returned by MT evaluation metrics depend on comparisons between translation hypotheses and reference translations, where the causes and the nature of the differences between them are not identified.

To cope with these issues and define system improvement priorities, the focus of MT evaluation research is gradually shifting towards profiling systems' behaviour with respect to various typologies of errors (Vilar et al., 2006; Popović and Ney, 2011; Farrús et al., 2012, *inter alia*). This shift has enriched the traditional MT evaluation framework with a new element, that is the actual errors done by a system. Until now, most of the research has focused on the relationship (*i.e.* the correlation) between two elements of the framework: humans and automatic evaluation metrics. As a new element of the framework, which becomes a sort of "evaluation triangle", the analysis of error annotations opens interesting research problems related to the relationships between: *i)* error types and human perception of MT quality and *ii)* error types and the sensitivity of automatic metrics.

Besides motivating further investigation on metrics featuring high correlation with human judgements (a well-established MT research sub-field, which is out of the scope of this paper), connecting the vertices of this triangle raises new challenging questions such as:

(1) Which types of MT errors have the highest impact on human perception of translation quality? Surprisingly, little prior work focused on this side of the triangle. Error annotations have been considered to highlight strengths and weaknesses of MT engines or to investigate the influence of different error types on post-editors' work. However, the direct connection between er-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1643–1653,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

rors and users' preferences has been only partially understood, mainly from a descriptive standpoint and through rudimentary techniques unsuitable to draw clear-cut conclusions or reliable inferences.
(2) To which types of errors are different MT evaluation metrics more sensitive? This side of the triangle has been even less explored. For instance, little has been done to understand which automatic metric is more suitable to assess system improvements with respect to a specific issue (*e.g.* word order or morphology) or to shed light on the joint impact of different error types on performance results calculated with different metrics.

To answer these questions, **we propose a robust statistical framework to analyse the impact of different error types**, alone and in combination, both on human perception of quality and on MT evaluation metrics' results. Our analysis is carried out **by employing linear mixed-effects models**, a generalization of linear regression models suited to model responses with fixed and random effects. Experiments are performed on data covering three translation directions (English to Chinese, Arabic and Russian). For each direction, two automatic translations were collected for around 400 sentences and were manually evaluated by expert translators through absolute quality judgements and error annotation.

Building on the advantages offered by linear mixed-effects models, our main contributions include:

- A rigorous method, novel to MT error analysis research, to relate MT issues to human preferences and MT metrics' results;
- The application of such method to three translation directions having English as source and different languages as target;
- A number of findings, specific to each language direction, which are out of the reach of the few simpler methods proposed so far.

Overall, our study has clear practical implications for MT systems' development and evaluation. Indeed, the proposed statistical analysis framework represents an ideal instrument to: *i)* identify translation issues having the highest impact on human perception of quality and *ii)* choose the most appropriate evaluation metric to measure progress towards their solution.

2 Related Work

Error analysis, as a way to identify systems' weaknesses and define priorities for their improvement, is gaining increasing interest in the MT community (Popović and Ney, 2011; Popovic et al., 2013). Along this direction, the initial efforts to develop error taxonomies covering different levels of granularity (Flanagan, 1994; Vilar et al., 2006; Farrús Cabeceran et al., 2010; Stymne and Ahrenberg, 2012; Lommel et al., 2014) have been recently complemented by investigations on how to exploit error annotations for diagnostic purposes. Error annotations of sentences produced by different MT systems, in different target languages and domains, have been used to determine the quality of translations according to the amount of errors encountered (Popovic et al., 2013), to design new automatic metrics that take into consideration human annotations (Popovic, 2012; Bojar et al., 2013), and to train classifiers that can automatic identify fine-grained errors in the MT output (Popović and Ney, 2011). The impact of edit operations on post-editors' productivity, which implicitly connects the severity of different errors to human activity, has also been studied (Temnikova, 2010; O'Brien, 2011; Blain et al., 2011), but few attempts have been made to explicitly model how fine-grained errors impact on human quality judgements and automatic metrics.

Recently, the relation between different error types, their frequency, and human quality judgements has been investigated from a descriptive standpoint in (Lommel et al., 2014; Popović et al., 2014). In both works, however, the underlying assumption that the most frequent error has also the largest impact on quality perception is not verified (in general and, least of all, across language pairs, domains, MT systems and post-editors). Another limitation of the proposed (univariate) analysis lies in the fact that it exclusively focuses on error types taken in isolation. This simplification excludes the possibility that humans, when assigning a global quality score to a translation, may be influenced not only by the error types but also by their interaction. The implications of such possibility call for a multivariate analysis capable to model also error interactions.

In (Kirchhoff et al., 2013), a statistically-grounded approach based on conjoint analysis has been used to investigate users' reactions to different types of translation errors. According to

their results, word order is the most dispreferred error type, and the count of the errors in a sentence is not a good predictor of users' preferences. Though more sophisticated than methods based on rough error counts, the conjoint model is bound to several constraints that limit its usability. In particular, the application of conjoint analysis in this context requires to: *i)* operate with semi-automatically created (hence artificial) data instead of real MT output, *ii)* manually define different levels of severity for each error type (*e.g.* high/medium/low), and *iii)* limit the number of error types considered to avoid the explosion of all possible combinations. Finally, the conjoint analysis framework is not able to explicitly model variance in the translated sentences, the human annotators, and the SMT systems used to translate the source sentences. Our claim is that avoiding any possible bias introduced by these factors should be a priority in the analysis of empirical observations in a given experimental setting.

So far, the relation between errors and automatic metrics has been analysed by measuring the correlation between single or total error frequencies and automatic scores (Popović and Ney, 2011; Farrús et al., 2012). Using two different error taxonomies, both works show that the sum of the errors has a high correlation with BLEU and TER scores. Similar to the aforementioned works addressing the impact of MT errors on human perception, these studies disregard error interactions, and their possible impact on automatic scores.

To overcome these issues, we propose a robust statistic analysis framework based on mixed-effects models, which have been successfully applied to several NLP problems such as sentiment analysis (Greene and Resnik, 2009), automatic speech recognition (Goldwater et al., 2010), and spoken language translation (Ruiz and Federico, 2014). Despite their effectiveness, the use of mixed-effects models in the MT field is rather recent and limited to the analysis of human post-editions (Green et al., 2013; Läubli et al., 2013). In both studies, the goal was to evaluate the impact of post-editing on the quality and productivity of human translation assuming an ANOVA mixed model for a between-subject design, in which human translators either post-edited or translated the same texts. Our scenario is rather different as we employ mixed models to measure the influence of different MT error types - expressed as continuous fixed effects - on quality judgements and automatic quality metrics. Mixed models, having the capability to absorb random variability due to the specific experimental set-up, provide a robust multivariate method to efficiently analyse the importance of error types.

Finally, differently from all previous works, our analysis is run on language pairs having English as source and languages distant from English (in term of morphology and word-order) as target.

3 Mixed-effects Models

Mixed-effects models - or simply mixed models - like any regression model, express the relationship between a *response variable* and some *covariates* and/or *contrast factors*. They enhance conventional models by complementing *fixed effects* with so-called *random effects*. Random effects are introduced to absorb random variability inherent to the specific experimental setting from which the observations are drawn. In general, random effects correspond to covariates that are not - or cannot be - exhaustively observed in an experiment, *e.g.* the human annotators and the evaluated systems. Hence, mixed models permit to elegantly cope with experimental design aspects that hinder the applicability of conventional regression models. These are, in particular, the use of repeated and/or clustered observations that introduce correlations in the response variable that clearly violate the independence and homoscedasticity assumptions of conventional linear, ANOVA, and logistic regression models. Significance testing with mixed models is in general more powerful, *i.e.* less prone to Type II Errors, and also permits to reduce the chance of Type I Errors in within-subject designs, which are prone to the "fallacy of language-as-a-fixed-effect" (Clark, 1973).

Random effects can be directly associated to the regression model parameters, as *random intercepts* and *random slopes*, and have the same form of the generic error component of the model, *i.e.* normally distributed with zero mean and unknown variance. As random effects introduce hidden variables, mixed models are trained with Expectation Maximization, while significance testing is performed via likelihood-ratio (LR) tests.

In this work we employ mixed *linear* models to measure the influence of different MT error types, expressed as continuous fixed effects, on quality

judgements or on automatic quality metrics.[1]

We illustrate mixed linear models (Baayen et al., 2008) by referring to our analysis, which addresses the relationships between a quality metric (y) and different types of errors (*e.g.* A, B, and C)[2] observed at the sentence level. For the sake of simplicity, we assume to have balanced repeated observations for one single crossed effect. That is, we have $i \in \{1, \ldots, I\}$ MT systems (our groups) each of which translated the same $j \in \{1, \ldots, J\}$ test sentences. Our response variable y_{ij} - a numeric quality score - is computed on each (sentence, system) pair, and we aim to investigate its relationship with error statistics available for each MT output, namely A_{ij}, B_{ij} and C_{ij}. A (possible) linear mixed model for our study would be:

$$\begin{aligned} y_{ij} &= \beta_0 + \beta_1\, A_{ij} + \beta_2\, B_{ij} + \beta_3\, C_{ij} + \\ &\quad b_{0,i} + b_{1,i} A_{ij} + b_{2,i} B_{ij} + b_{3,i} C_i + \epsilon_{ij} \end{aligned} \quad (1)$$

The model is split into two lines on purpose. The first line shows the fixed effect component, that is intercept (β_0) and slopes ($\beta_1, \beta_2, \beta_3$) for each error type. The second line specifies the random structure of the model, which includes random intercept and slopes for each MT system and the residual error. Borrowing the notation from (Green et al., 2013), we conveniently rewrite (1) in the group-wise arranged matrix notation:

$$y_i = x_i^T \beta + z_i^T b_i + \epsilon_i \quad (2)$$

where y_i is the $J \times 1$ vector of responses, x_i is the $J \times p$ design matrix of covariates (including the intercept) with fixed coefficients $\beta \in \mathcal{R}^{p \times 1}$, z is the random structure matrix defined by $J \times q$ covariates with random coefficients $b_i \in \mathcal{R}^{q \times 1}$, and ϵ_i is the vector of residuals (in our example, $p = 4$ and $q = 4$). By packing together vectors and matrices indexed over groups i, we can rewrite the model in a general form (Baayen et al., 2008), which can represent any possible crossed-effects and random structures defined over them allowing, at the same time, for a compact model specification:

$$\begin{aligned} y &= X^T \beta + Z^T b + \epsilon \\ &\quad \epsilon \sim \mathcal{N}(0, \sigma^2 I),\ b \sim \mathcal{N}(0, \sigma^2 \Sigma), b \perp \epsilon \end{aligned} \quad (3)$$

where Σ is the relative variance-covariance $q \times q$ matrix of the random effects (now $q = 4I$), σ^2 is the variance of the per-observation term ϵ, the symbol $\perp$ denotes independence of random variables, and $\mathcal{N}$ indicates the multivariate normal distribution. While b, σ, and Σ are estimated via maximum likelihood, the single random intercept and slope values for each group are calculated subsequently. They are referred to as Best Linear Unbiased Predictors (BLUPS) and, formally, are *not* parameters of the model.

The significance of the contribution of each single parameter (*e.g.* single entries of Σ) to the goodness of fit can be tested via likelihood ratio. In this way, both the fixed and random effect structure of the model can be investigated with respect to its actual necessity to the model.

4 Dataset

For our analysis we used a dataset that covers three translation directions, corresponding to English to Chinese, Arabic, and Russian. An international organization provided us a set of English sentences together with their translation produced by two anonymous MT systems. For each evaluation item (source sentence and two MT outputs) three experts were asked to assign quality scores to the MT outputs, and a fourth expert was asked to annotate translation errors. The four experts, who were all professional translators native in the examined target languages, were carefully trained to get acquainted with the evaluation guidelines and the annotation tool specifically developed for these evaluation tasks (Girardi et al., 2014). The annotation process was carried out in parallel by all annotators over one week, resulting in a final dataset composed of 312 evaluation items for the ENZH direction, 393 for ENAR, and 437 for ENRU.

4.1 Quality Judgements

Quality judgements were collected by asking the three experts to rate each automatic translation according to a 1-5 Likert scale, where 1 means "incomprehensible translation" and 5 means "perfect translation". The distribution of the collected annotations with respect to each quality score is shown in Figure 1. As we can see, this distribution reflects different levels of perceived quality across languages. ENZH, for instance, has the highest number of low quality scores (1 and 2), while ENRU has the highest number of high qual-

[1] Although mixed *ordinal* models (Tutz and Hennevogl, 1996) are in principle more appropriate to target quality judgements, in our preliminary investigations mixed linear models showed a significantly higher predictive power.

[2] Here, A, B and C represent three generic error classes. Their actual number in a given experimental setting will depend on the granularity of the reference error taxonomy.

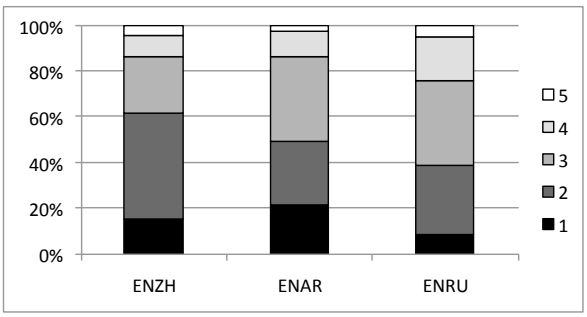


Figure 1: Distribution of quality scores.

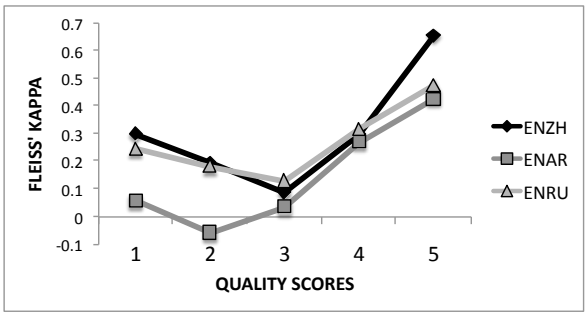


Figure 2: Class specific inter-annotator agreement.

ity scores (4 and 5).

Table 1 shows the average of all the quality scores assigned by each annototator as well as the average score obtained for each MT system. These values demonstrate the variability of annotators and systems. A particularly high variability among human judges is observed for the ENAR language direction (also reflected by the inter-annotator agreement scores discussed below), while ENZH shows the highest variability between systems. As we will see in §5.1, we successfully cope with this variability by considering systems and annotators as random effects, which allow the regression models to abstract from these differences.

	Ann1	Ann2	Ann3	Sys1	Sys2
ENZH	2.38	2.69	2.21	2.29	2.56
ENAR	2.76	2.77	1.84	2.39	2.53
ENRU	2.82	2.72	2.96	2.87	2.79

Table 1: Average quality scores per annotator and per system.

Inter-annotator agreement was computed using the *Fleiss' kappa coefficient* (Fleiss, 1971), and resulted in 22.70% for ENZH, 5.24% for ENAR, and 21.80% for ENRU. While for ENZH and ENRU the results fall in the range of "fair" agreement (Landis and Koch, 1977), for ENAR only "slight" agreement is reached, reflecting the higher annotators' variability evidenced in Table 1.

A more fine-grained agreement analysis is presented in Figure 2, where the *kappa* values are given for each score class. In general we notice a lower agreement on the intermediate quality scores, while annotators tend to agree on very bad and, even more, on good translations. In particular, we see that the agreement for ENAR is systematically lower than the values measured for the other languages on all the score classes.

4.2 Error Annotation

This evaluation task was carried out by one expert for each language direction, who was asked to identify the type of errors present in the MT output and to mark their position in the text. Since the focus of our work is the analysis method rather than the definition of an ideal error taxonomy, for the difficult language directions addressed we opted for the following general error classes, partially overlapping with (Vilar et al., 2006): *i)* reordering errors, *ii)* lexicon errors (including wrong lexical choices and extra words), *iii)* missing words, *iv)* morphology errors.

Figure 3 shows the distribution of the errors in terms of affected tokens (words) for each error type. Since token counts for Chinese are not word-based but character-based, for readability purposes the number of errors counted for Chinese translations have been divided by 2.5. Note also that morphological errors annotated for ENZH involve only 13 characters and thus are not visible in the plot. The total number of errors amounts to 16,320 characters for ENZH, 4,926 words for ENAR, and 5,965 words for ENRU.

This distribution highlights some differences between languages directions. For example, translations into Arabic and Russian present several morphology errors, while word reordering is the most frequent issue for translations into Chinese. As we will see in §5.1, error frequency does not give a direct indication of their impact on traslation quality judgements.

4.3 Automatic Metrics

In our investigation we consider three popular automatic metrics: sentence-level BLEU (Lin and Och, 2004), TER (Snover et al., 2006), and GTM (Turian et al., 2003). We compute all automatic scores by relying on a single reference and by

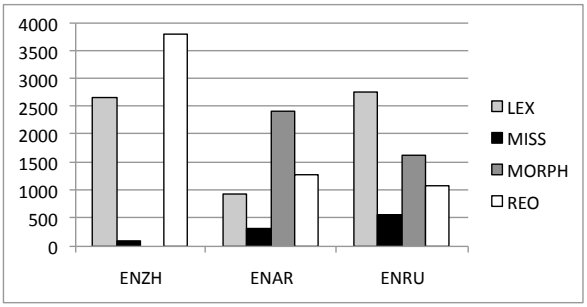


Figure 3: Distribution of error types.

means of standard packages. In particular, automatic scores on Chinese are computed at the character level. Moreover, as we use metrics as response variables for our regression models, we compute all metrics at the sentence level. The overall mean scores for all systems and languages are reported in Table 2. Differences in systems' performance can be observed for all language pairs; as we will observe in §5.2 such variability explains the effectiveness of considering the MT systems as a random effect.

	BLEU		TER		GTM	
	Sys1	Sys2	Sys1	Sys2	Sy1	Sys2
ENZH	27.95	44.11	64.52	48.13	62.15	72.30
ENAR	19.63	25.25	68.83	63.99	47.20	52.33
ENRU	27.10	31.07	60.89	54.41	53.74	56.41

Table 2: Overall automatic scores per system.

5 Experiments

To assess the impact of translation errors on MT quality we perform two sets of experiments. The first set (§5.1) addresses the relation between errors and human quality judgements. The second set (§5.2) focuses on the relation between errors and automatic metrics. In both cases, before measuring the impact of different errors on the response variable (respectively quality judgements and metrics), we validate the effectiveness of mixed linear models by comparing their prediction capability with other methods.

In all experiments, error counts of each category were normalized into percentages with respect to the sentence length and mapped in a logarithmic scale. In this way, we basically assume that the impact of errors tends to saturate above a given threshold, hypothesis that also results in better fits by our models.[3] Notice that while the chosen log-10 base is easy to interpret, linear models can implicitly adjust it. Our analysis makes use of mixed linear models incorporating, as fixed effects, the four types of errors (*lex*, *miss*, *morph* and *reo*) and their pairwise interactions (the product of the single error log counts), while their random structure depends on each specific experiment. For the experiments we rely on the R language (R Core Team, 2013) implementation of linear mixed model in the *lme4* library (Bates et al., 2014).

We assess the quality of our mixed linear models (*MLM*) by comparing their prediction capability with a sequence of simpler linear models including only fixed effects. In particular, we built five univariate models and two multivariate models. The univariate models use as covariates, respectively, the sum of all error types (*baseline*), and each of the four types of errors (*lex*, *miss*, *morph* and *reo*). The two multivariate models include all the four error types, considering them without interactions (*FLM w/o Interact.*) and with interactions (*FLM)*.

Prediction performance is computed in terms of Mean Absolute Error (MAE),[4] which we estimate by averaging over 1,000 random splits of the data in 90% training and 10% test. In particular, for the human quality classes we pick the integer between 1-5 that is closest to the predicted value.

5.1 Errors vs. Quality Judgements

The response variable we target in this experiment is the quality score produced by human annotators. Our measurements follow a typical within-subject design in which all the 3 annotators are exposed to the same conditions (levels of the independent variables), corresponding in our case to perfectly balanced observations from 2 MT systems and N sentences. This setting results in repeated or clustered observations (thus violating independence) corresponding to groups which naturally identify possible random effects,[5] namely the annotators (3 levels with 2xN observations each), the systems (2 levels and 3xN observations each), and the sen-

[3] In other words, we assume that human sensitivity to errors follows a log-scale law: *e.g.* more sensitive to variations in the interval [1-10] that in the interval [30-40].

[4] MAE is calculated as the average of the absolute errors $|f_i - y_i|$, where f_i is the prediction of the model and y_i the true value for the i^{th} instance. As it is a measure of error, lower MAE scores indicate that our predictions are closer to the true values of each test instance.

[5] In all our experiments, random effects are limited to random shifts since preliminary experiments also including random slopes did not provide consistent results.

Model	ENZH	ENAR	ENRU
baseline	0.58	0.73	0.67
lex	0.67	0.78	0.72
miss	0.72	0.89	0.74
morph	0.72	0.89	0.74
reo	0.70	0.82	0.76
FLM w/o Interact.	0.59	0.77	0.65
FLM	0.57	0.72	0.63
MLM	0.53	0.61	0.61

Table 3: Prediction capability of human judgements (MAE).

Error	ENZH	ENAR	ENRU
Intercept	4.29	3.79$_\bullet$	4.21
lex	-1.27	-0.96	-1.12
miss	-1.76	-0.90	-1.30
morph	-0.48$_\circ$	-0.83	-0.51
reo	-1.01	-0.75	-0.18
lex:miss	1.00	0.39	0.68
lex:morph	-	0.29	0.32
lex:reo	0.50	0.21	-
miss:morph	-	0.35	-
miss:reo	0.54	0.33	-
morph:reo	-	0.37	-

Table 4: Effect of translation errors on MT quality perception on all judged sentences. Reported coefficients (β) are all statistically significant with $p \leq 10^{-4}$, except those marked with $_\bullet$ ($p \leq 10^{-3}$), and $_\circ$ ($p \leq 10^{-2}$).

tences (N levels with 6 observations each). In principle, such random effects permit to remove systematic biases of individual annotators, single systems and even single sentences, which are modelled as random variables sampled from distinct populations.

Table 3 shows a comparison of the **prediction capability** of the mixed model[6] with simpler approaches. While the good performance achieved by our strong baseline cannot be outperformed by separately counting the number of errors of a single type, lower MAE results are obtained by methods based on multivariate analysis. Among them, FLM brings the first consistent improvements over the baseline by considering error interactions, while MLM leads to the lowest MAE due to the addition of random effects. The importance of random effects is particularly evidenced by ENAR (12 points below the baseline). Indeed, as discussed in §4.1, for this language combination human annotators show the lowest agreement score. This variability, which hides the smaller differences in systems' behaviour, demonstrates the importance of accounting for the erratic factors that might influence empirical observations in a given setting. The good performance achieved by MLM, combined with their high descriptive power,[7] motivates their adoption in our study.

Concerning the analysis of **error impact**, Table 4 shows the statistically significant coefficients for the full-fledged MLM models for each translation direction. By default, all reported coefficients have p-values $\leq 10^{-4}$, while those marked with $\bullet$ and $\circ$ have respectively p-values $\leq 10^{-3}$ and $\leq 10^{-2}$. Slope coefficients basically show the impact of different error types (alone and in combination) on human quality scores. Those that are not statistically significant are omitted as they do not increase the fitting capability of our model. As can be seen from the table, such impact varies across the different language combinations. While for ENZH and ENRU *miss* is the error having the highest impact (highest decrement with respect to the intercept), the most problematic error for ENAR is *lex*. It is interesting to observe that positive values for error combinations indicate that their combined impact is lower that the sum of the impact of the single errors. For instance, while for ENZH a one-step increment in *lex* and *miss* errors would respectively cause a reduction in the human judgement of 1.27 and 1.76, their occurrence in the same sentence would be discounted by 1.00. This would result in a global judgement of 2.26 (4.29 -1.27 -1.76 +1.00) instead of 1.26. While for ENAR this phenomenon can be observed for all error combinations, such discount effects are not always significant for the other two language pairs. The existence of discount effects of various magnitude associated to the different error combinations is a novel finding made possible by the adoption of mixed-effect models.

Another interesting observation is that, in contrast with the common belief that the most frequent errors have the highest impact on human quality judgements, our experiments do not reveal such strict correlation (at least for the examined language pairs). For instance, for ENZH and ENRU the impact of *miss* errors is higher than the impact of other more frequent issues.

[6]Note that the mixed model used in prediction does not include the random effect on sentences since the training samples do not guarantee sufficient observations for each test sentence.

[7]Note that the strong baseline used for comparison is not capable to describe the contribution of the different error types.

Model	BLEU score			TER			GTM		
	ENZH	ENAR	ENRU	ENZH	ENAR	ENRU	ENZH	ENAR	ENRU
baseline	12.4	9.8	12.2	15.7	13.4	14.4	9.8	10.6	11.5
lex	12.9	10.4	13.0	16.3	13.8	14.9	9.7	10.9	12.1
miss	13.8	10.5	14.1	17.3	14.2	16.4	10.5	11.1	13.2
morph	13.9	10.3	13.6	17.5	13.8	16.3	10.5	10.9	13.1
reo	13.7	10.5	14.0	17.4	14.1	16.3	10.4	11.1	13.1
FLM w/o Interact.	12.9	9.9	12.2	16.3	13.5	14.4	9.7	10.7	11.7
FLM	12.3	9.7	12.1	15.6	13.4	14.3	9.4	10.6	11.6
MLM	10.8	9.5	12.0	14.7	13.0	14.2	8.9	10.5	11.6

Table 5: Prediction capability of BLEU score, TER and GTM (MAE).

5.2 Errors vs. Automatic Metrics

In this experiment, the response variable is an automatic metric which is computed on a sample of MT outputs (which are again perfectly balanced over systems and sentences) and a set of reference translations. As no subjects are involved in the experiment, random variability is assumed to come from the involved systems, the tested sentences, and the unknown missing link between the covariates (error types) and the response variable which is modelled by the residual noise. Notice that, in this case, the random effect on the sentences also incorporates in some sense the randomness of the corresponding reference translations, which are themselves representatives of larger samples.

The **prediction capability** of the mixed model, in comparison with the simpler ones, is reported in Table 5. Also in this case, the low MAE achieved by the baseline is out of the reach of univariate methods. Again, small improvements are brought by FLM when considering error interactions, whereas the most visible gains are achieved by MLM due to their control of random effects. This is more evident for some language combinations and can be explained by the differences in systems' performance, a variability factor easily absorbed by random effects. Indeed, the largest MAE decrements over the baseline are always observed for ENZH (for which the overall mean results reported in Table 2 show the largest differences) and the smallest decrements relate to language/metric combinations where systems' behaviour is more similar (*e.g.* ENRU/GTM).

Concerning the analysis of **error impact**, Table 6 shows how different error types (alone and in combination) influence performance results measured with automatic metrics. To ease interpretation of the reported figures we also show Pearson and Spearman correlations of each set of coefficients (excluding intercept estimates) with their corresponding coefficients reported in Table 4. In fact, our primary interest in this experiment is to see which metrics show a sensitivity to specific error types similar to human perception. As we can see, the coefficients for each metric significantly vary depending on the language, for the simple reason that also the distribution and co-occurrence of errors vary significantly across the different languages and MT systems. Remarkably, for some translation directions, some of the metrics show a sensitivity to errors that is very similar to that of human judges. In particular, BLEU for ENZH and ENAR, and GTM for ENZH show a very high correlation with the human sensitivity to translation errors, with Pearson correlation coefficient $\geq$ 0.97. For ENRU, the best Pearson correlation is instead achieved by TER (-0.78).

Besides these general observations, a closer look at the reported scores brings additional findings. In three cases (BLEU for ENZH, GTM for ENZH and ENAR) the analysed metrics are most sensitive to the same error type that has the highest influence on human judgements (according to Table 4, these are *miss* for ENZH and ENRU, *lex* for ENAR). On the contrary, in one case (TER for ENZH) the analysed metric is insensitive to the error type (*miss*) which has the highest impact on human quality scores. From a practical point of view, these remarks provide useful indications about the appropriateness of each metric to highlight the deficiencies of a specific system and to measure improvements targeting specific issues. As a rule of thumb, for instance, to measure improvements of an ENZH system with respect to *miss*ing words, it would be more advisable to use BLEU or GTM instead of TER.[8]

[8]Note that this conclusion holds for our data sample, in which different types of errors co-occur and only one reference translation is available. In such conditions, our regression model shows that TER is not influenced by *miss* errors in a statistically significant way. This does not mean that TER is insensitive to missing words when occurring in isolation,

Error	BLEU score ENZH	BLEU score ENAR	BLEU score ENRU	TER ENZH	TER ENAR	TER ENRU	GTM ENZH	GTM ENAR	GTM ENRU
Intercept	60.55$_{\square}$	38.45$_{\circ}$	51.73	32.41$_{\square}$	52.25$_{\bullet}$	33.4$_{\bullet}$	83.57$_{\circ}$	60.11$_{\bullet}$	75.38
lex	-18.78	-9.25	-16.57	16.87	9.66	18.45	-13.63	-7.60	-16.13
miss	-23.20	-10.41	-6.75	-	-	8.24	-14.87	-	-5.98
morph	-	-9.97	-12.65	-	8.90	11.41	-	-6.60	-10.42
reo	-13.27	-7.62	-10.57	14.44	9.81	6.39	-7.29	-5.50	-7.03
lex:miss	14.37	4.97$_{\circ}$	-	-	-	-	8.24$_{\bullet}$	-	-
lex:morph	-	-	5.27$_{\bullet}$	-	-	-5.22$_{\circ}$	-	-	4.92
lex:reo	8.57	3.57$_{\circ}$	5.40$_{\bullet}$	-7.24$_{\circ}$	-4.35$_{\circ}$	-	5.46	3.22$_{\circ}$	3.65$_{\square}$
miss:morph	-	4.44$_{\circ}$	-	-	-	-	-	-	-
miss:reo	6.74$_{\circ}$	-	4.30	-	-	-6.38$_{\circ}$	5.07$_{\circ}$	-	4.71$_{\circ}$
morph:reo	-	3.81$_{\bullet}$	-	-	-4.97$_{\bullet}$	-	-	2.57$_{\circ}$	-
Pearson	**0.98**	**0.97**	0.70	-0.58	-0.78	**-0.78**	**0.98**	0.78	0.74
Spearman	**0.97**	**0.91**	0.73	-0.57	-0.59	**-0.80**	**0.97**	0.59	0.76

Table 6: Effect of translation errors on BLEU score, TER and GTM on all judged sentences and correlation with their corresponding effects on human quality scores (from Table 4). Reported coefficients (β) are statistically significant with $p \leq 10^{-4}$, except those marked with $_{\bullet}$ ($p \leq 10^{-3}$), $_{\circ}$ ($p \leq 10^{-2}$) and $_{\square}$($p \leq 10^{-1}$).

Similar considerations also apply to the analysis of the impact of error combinations. The same discount effects that we noticed when analysing the impact of errors' co-occurrence on human perception (§5.1) are evidenced, with different degrees of sensitivity, by the automatic metrics. While some of them substantially reflect human response (*e.g.* BLEU and GTM for ENZH), in some cases we observe either the insensitivity to specific combinations (mostly for ENAR), or a higher sensitivity compared to the values measured for human assessors (mostly for ENRU, where the impact of *miss:reo* combinations is discounted - hence underestimated - by all the metrics).

Despite such small differences, the coherence of our results with previous findings (§5.1) suggests the reliability of the applied method. Completing the picture along the side of the MT evaluation triangle which connects error annotations and automatic metrics, our findings contribute to shed light on the existing relationships between translation errors, their interaction, and the sensitivity of widely used automatic metrics.

6 Conclusion

We investigated the MT evaluation triangle (having as corners *automatic metrics*, *human quality judgements* and *error annotations*) along the two less explored sides, namely: *i)* the relation between MT errors and human quality judgements and *ii)* the relation between MT errors and automatic metrics. To this aim we employed a robust statistical analysis framework based on linear mixed-effects models (the first contribution of the paper), which have a higher descriptive power than simpler methods based on the raw count of translation errors and are less artificial compared to previous statistically-grounded approaches.

Working on three translation directions having Chinese, Arabic and Russian as target (our second contribution), we analysed error-annotated translations considering the impact of specific errors (alone and in combination) and accounting for the variability of the experimental set-up that originated our empirical observations. This led us to interesting findings specific to each language pair (third contribution). Concerning the relation between MT errors and quality judgements, we have shown that: *i)* the frequency of errors of a given type does not correlate with human preferences, *ii)* errors having the highest impact can be precisely isolated and *iii)* the impact of error interactions is often subject to measurable and previously unknown "discount" effects. Concerning the relation between MT errors and automatic metrics (BLEU, TER and GTM), our analysis evidenced significant differences in the sensitivity of each metric to different error types. Such differences provide useful indications about the most appropriate metric to assess system improvements with respect to specific weaknesses. If learning from errors is a crucial aspect of improving expertise, our method and the resulting empirical findings represent a significant contribution towards a

but that TER becomes less sensitive to such errors when they co-occur with other types of errors. Overall, our experiments show that when MT outputs contain more than one error type, automatic metrics show different levels of sensitivity to each specific error type.

more informed approach to system development, improvement and evaluation.

Acknowledgements

This work has been partially supported by the EC-funded project MateCat (ICT-2011.4.2-287688).

References

Harald R. Baayen, Douglas J. Davidson, and Douglas M. Bates. 2008. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of memory and language*, 59(4):390–412.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Douglas Bates, Martin Maechler, Ben Bolker, and Steven Walker, 2014. *lme4: Linear mixed-effects models using Eigen and S4*. R package version 1.1-6.

Frédéric Blain, Jean Senellart, Holger Schwenk, Mirko Plitt, and Johann Roturier. 2011. Qualitative analysis of post-editing for high quality machine translation. In Asia-Pacific Association for Machine Translation (AAMT), editor, *Machine Translation Summit XIII*, Xiamen (China), 19-23 sept.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August. Association for Computational Linguistics.

Herbert H. Clark. 1973. The language-as-fixed-effect fallacy: A critique of language statistics in psychological research. *Journal of verbal learning and verbal behavior*, 12(4):335–359.

Mireia Farrús, Marta R. Costa-jussà, and Maja Popović. 2012. Study and correlation analysis of linguistic, perceptual, and automatic machine translation evaluations. *J. Am. Soc. Inf. Sci. Technol.*, 63(1):174–184, January.

Mireia Farrús Cabeceran, Marta Ruiz Costa-Jussà, José Bernardo Mariño Acebal, José Adrián Rodríguez Fonollosa, et al. 2010. Linguistic-based evaluation criteria to identify statistical machine translation errors. In *Proceedings of the 14th Annual Conference of the European Association for Machine Translation (EAMT)*.

Mary Flanagan. 1994. Error classification for mt evaluation. In *Technology Partnerships for Crossing the Language Barrier: Proceedings of the First Conference of the Association for Machine Translation in the Americas*, pages 65–72.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5).

Christian Girardi, Luisa Bentivogli, Mohammad Amin Farajian, and Marcello Federico. 2014. Mt-equal: a toolkit for human assessment of machine translation output. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 120–123, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Sharon Goldwater, Daniel Jurafsky, and Christopher D. Manning. 2010. Which words are hard to recognize? prosodic, lexical, and disfluency factors that increase speech recognition error rates. *Speech Communication*, 52(3):181–200.

Spence Green, Jeffrey Heer, and Christopher D. Manning. 2013. The efficacy of human post-editing for language translation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 439–448. ACM.

Stephan Greene and Philip Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 503–511, Stroudsburg, PA, USA. Association for Computational Linguistics.

Katrin Kirchhoff, Daniel Capurro, and Anne M. Turner. 2013. A conjoint analysis framework for evaluating user preferences in machine translation. *Machine Translation*, pages 1–17.

Richard J. Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33 (1):159–174.

Samuel Läubli, Mark Fishel, Gary Massey, Maureen Ehrensberger-Dow, and Martin Volk. 2013. Assessing Post-Editing Efficiency in a Realistic Translation Environment. In Michel Simard Sharon O'Brien and Lucia Specia (eds.), editors, *Proceedings of MT Summit XIV Workshop on Post-editing Technology and Practice*, pages 83–91, Nice, France.

Chin-Yew Lin and Franz Josef Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of Coling 2004*, pages 501–507, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Arle Lommel, Aljoscha Burchardt, Maja Popović, Kim Harris, Eleftherios Avramidis, and Hans Uszkoreit.

2014. Using a new analytic measure for the annotation and analysis of mt errors on real data. In *Proceedings of the 17th Conference of the European Association for Machine Translation (EAMT)*, Dubrovnik, Croatia, June.

Sharon O'Brien. 2011. *Cognitive Explorations of Translation*. Bloomsbury Studies in Translation. Bloomsbury Academic.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. Research Report RC22176, IBM Research Division, Thomas J. Watson Research Center.

Maja Popović and Hermann Ney. 2011. Towards automatic error analysis of machine translation output. *Comput. Linguist.*, 37(4):657–688, December.

Maja Popovic, Eleftherios Avramidis, Aljoscha Burchardt, Sabine Hunsicker, Sven Schmeier, Cindy Tscherwinka, David Vilar, and Hans Uszkoreit. 2013. Learning from human judgments of machine translation output. In *Proceedings of the MT Summit XIV*. Proceedings of MT Summit XIV.

Maja Popović, Arle Lommel, Aljoscha Burchardt, Eleftherios Avramidis, and Hans Uszkoreit. 2014. Relations between different types of post-editing operations, cognitive effort and temporal effort. In *Proceedings of the 17th Conference of the European Association for Machine Translation (EAMT)*, Dubrovnik, Croatia, June.

Maja Popovic. 2012. Class error rates for evaluation of machine translation output. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 71–75, Montréal, Canada, June. Association for Computational Linguistics.

R Core Team, 2013. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Nick Ruiz and Marcello Federico. 2014. Assessing the Impact of Speech Recognition Errors on Machine Translation Quality. In *11th Conference of the Association for Machine Translation in the Americas (AMTA)*, Vancouver, BC, Canada.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *5th Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, Massachusetts, August.

Sara Stymne and Lars Ahrenberg. 2012. On the practice of error analysis for machine translation evaluation. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uur Doan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Irina Temnikova. 2010. Cognitive evaluation approach for a controlled language post-editing experiment. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

Joseph P. Turian, I. Dan Melamed, and Luke Shen. 2003. Evaluation of machine translation and its evaluation. In *Proceedings of the MT Summit IX*.

Gerhard Tutz and Wolfgang Hennevogl. 1996. Random effects in ordinal regression models. *Computational Statistics & Data Analysis*, 22(5):537–557.

David Vilar, Jia Xu, Luis Fernando dHaro, and Hermann Ney. 2006. Error analysis of statistical machine translation output. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, pages 697–702.

Refining Word Segmentation Using a Manually Aligned Corpus for Statistical Machine Translation

Xiaolin Wang **Masao Utiyama** **Andrew Finch** **Eiichiro Sumita**
National Institute of Information and Communications Technology
{xiaolin.wang,mutiyama,andrew.finch,eiichiro.sumita}@nict.go.jp

Abstract

Languages that have no explicit word delimiters often have to be segmented for statistical machine translation (SMT). This is commonly performed by automated segmenters trained on manually annotated corpora. However, the word segmentation (WS) schemes of these annotated corpora are handcrafted for general usage, and may not be suitable for SMT. An analysis was performed to test this hypothesis using a manually annotated word alignment (WA) corpus for Chinese-English SMT. An analysis revealed that 74.60% of the sentences in the WA corpus if segmented using an automated segmenter trained on the Penn Chinese Treebank (CTB) will contain conflicts with the gold WA annotations. We formulated an approach based on word splitting with reference to the annotated WA to alleviate these conflicts. Experimental results show that the refined WS reduced word alignment error rate by 6.82% and achieved the highest BLEU improvement (0.63 on average) on the Chinese-English open machine translation (OpenMT) corpora compared to related work.

1 Introduction

Word segmentation is a prerequisite for many natural language processing (NLP) applications on those languages that have no explicit space between words, such as Arabic, Chinese and Japanese. As the first processing step, WS affects all successive steps, thus it has a large potential impact on the final performance. For SMT, the unsupervised WA, building translation models and reordering models, and decoding are all based on segmented words.

Automated word segmenters built through supervised-learning methods, after decades of intensive research, have emerged as effective solutions to WS tasks and become widely used in many NLP applications. For example, the Stanford word segmenter (Xue et al., 2002)[1] which is based on conditional random field (CRF) is employed to prepare the official corpus for NTCIR-9 Chinese-English patent translation task (Goto et al., 2011).

However, one problem with applying these supervised-learning word segmenters to SMT is that the WS scheme of annotating the training corpus may not be optimal for SMT. (Chang et al., 2008) noticed that the words in CTB are often too long for SMT. For example, a full Chinese personal name which consists of a family name and a given name is always taken as a single word, but its counterpart in English is usually two words.

Manually WA corpora are precious resources for SMT research, but they used to be only available in small volumes due to the production cost. For example, (Och and Ney, 2000) initially annotated 447 English-French sentence pairs, which later became the test data set in ACL 2003 shared task on word alignment (Mihalcea and Pedersen, 2003), and was used frequently thereafter (Liang et al., 2006; DeNero and Klein, 2007; Haghighi et al., 2009)

For Chinese and English, the shortage of manually WA corpora has recently been relieved by the linguistic data consortium (LDC) [2] GALE Chinese-English word alignment and tagging training corpus (the GALE WA corpus)[3]. The corpus is considerably large, containing 4,735 documents, 18,507 sentence pairs, 620,189 Chinese tokens, 518,137 English words, and 421,763

[1] http://nlp.stanford.edu/software/segmenter.shtml

[2] http://catalog.ldc.upenn.edu

[3] Catalog numbers: LDC2012T16, LDC2012T20, LDC2012T24 and LDC2013T05.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1654–1664,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

alignment annotations. The corpus carries no Chinese WS annotation, and the WA annotation was performed between Chinese characters and English words. The alignment identifies minimum translation units and relations [4], referred as atomic blocks and atomic edges, respectively, in this paper. Figure 1 shows an example that contains six atomic edges.

Visual inspection of the segmentation of an automatic segmenter with reference to a WA corpus revealed a number of inconsistencies. For example, consider the word "bao fa" in Figure 1. Empirically we observed that this word is segmented as a single token by an automatic segmenter trained on the CTB, however, this segmentation differs with the alignment in the WA corpus, since its two components are aligned to two different English words. Our hypothesis was that the removal of these inconsistencies would benefit machine translation performance (this is explained further in Section 2.3), and we explored this idea in this work.

This paper focuses on optimizing Chinese WS for Chinese-English SMT, but both the research method and the proposed solution are language-independent. They can be applied to other language pairs.

The major contributions of this paper include,

- analyze the CTB WS scheme for Chinese-English SMT;
- propose a lexical word splitter to refine the WS;
- achieve a BLEU improvement over a baseline Stanford word segmenter, and a state-of-the-art extension, on Chinese-English OpenMT corpora.

The rest of this paper is organized as follows: first, Section 2 analyzes WS using a WA corpus; next, Section 3 proposes a lexical word splitter to refine WS; then, Section 4 evaluates the proposed method on end-to-end SMT as well as word segmentation and alignment; after that, Section 5 compares this work to related work; finally, Section 6 concludes this paper.

[4]Guidelines for Chinese-English Word Alignment(Version 4.0)

2 Analysis of a General-purpose Automatic Word Segmenter

This section first briefly describes the GALE WA corpus, then presents an analysis of the WS arising from a CTB-standard word segmenter with reference to the segmentation of the atomic blocks in the GALE WA corpus, finally the impact of the findings on SMT is discussed.

2.1 GALE WA corpus

The GALE WA corpus was developed by the LDC, and was used as training data in the DARPA GALE global autonomous language exploitation program [5]. The corpus incorporates linguistic knowledge into word aligned text to help improve automatic WA and translation quality. It employs two annotation schemes: alignment and tagging (Li et al., 2010). Alignment identifies minimum translation units and translation relations; tagging adds contextual, syntactic and language-specific features to the alignment annotation. For example, the sample shown in Figure 1 carries tags on both alignment edges and tokens.

The GALE WA corpus contains 18,057 manually word aligned Chinese and English parallel sentences which are extracted from newswire and web blogs. Table 1 presents the statistics on the corpus. One third of the sentences are approximately newswire text, and the remainder consists of web blogs.

2.2 Analysis of WS

In order to produce a Chinese word segmentation consistent with the CTB standard we used the Stanford Chinese word segmenter with a model trained on the CTB corpus. We will refer to this as the 'CTB segmenter' in the rest of this paper.

The Chinese sentences in the GALE WA corpus were first segmented by the CTB segmenter, and the predicted words were compared against the atomic blocks with respect to the granularity of segmentation. The analysis falls into the following three categories, two of which may be potentially harmful to SMT:

- Fully consistent: the word locates within the block of one atomic alignment edge. For example, in Figure 2(a), the Chinese text has

[5]https://catalog.ldc.upenn.edu/LDC2012T16

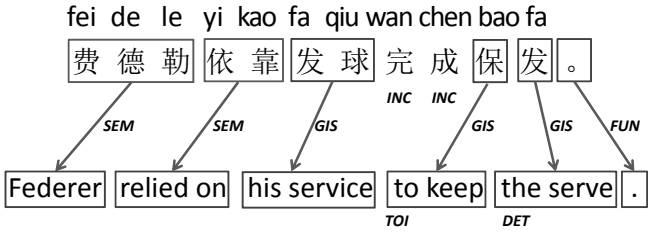


Figure 1: Example from the GALE WA corpus. Each line arrow represents an atomic edge, and each box represents an atomic block. SEM (semantic), GIS (grammatically inferred semantic) and FUN (function) are tags of edges. INC (not translated), TOI (to-infinitive) and DET (determiner) are tags of tokens.

Genre	# Files	# Sentences†	# CN tokens	# EN tokens	# Alignment edges
Newswire	2,175	6,218	246,371	205,281	164,033
Web blog	2,560	11,839	373,818	312,856	257,730
Total	4,735	18,057	620,189	518,137	421,763

Table 1: GALE WA corpus. † Sentences rejected by the annotators are excluded.

four atomic blocks; the CTB segmenter produces five words which all locate within the blocks, so they are all small enough.

- Alignment inconsistent: the word aligns to more than one atomic block, but the target expression is contiguous, allowing for correct phrase pair extraction (Zens et al., 2002). For example, in Figure 2(b), the characters in the word "shuang fang", which is produced by the CTB segmenter, contains two atomic blocks, but the span of the target "to both side" is continuous, therefore the phrase pair "shuang fang ||| to both sides" can be extracted.

- Alignment inconsistent and extraction hindered: the word aligned to more than one atomic block, and the target expression is not contiguous, which hinders correct phrase pair extractions. For example, in Figure 2(c), the word "zeng chan" has to be split in order to match the target language.

Table 2 shows the statistics of the three categories of CTB WS on the GALE WA corpus. 90.74% of the words are fully consistent, while the remaining 9.26% of the words have inconsistent alignments. 74.60% of the sentences contain this problem. The category with inconsistent alignment and extraction hindered only accounts for 0.46% of the words, affecting 9.06% of the sentences.

2.3 Impact of WS on SMT

The word alignment has a direct impact on the nature of both the translation model, and lexical reordering model in a phrase-base SMT system. The words in last two categories are all longer than an atomic block, which might lead to problems in the word alignment in two ways:

- First, longer words tend to be more sparse in the training corpus, thus the estimated distribution of their target phrases are less accurate.

- Second, the alignment from them to target sides are one-to-many, which is much more complicated and requires fertilized alignment models such as IBM model 4 – 6 (Och and Ney, 2000).

The words in the category of "fully consistent" can be aligned using simple models, because the alignment from them to the target side are one-to-one or many-to-one, and simple alignment models such as IBM model 1, IBM model 2 and HMM model are sufficient (Och and Ney, 2000).

3 Refining the Word Segmentation

In the last subsection, it was shown that 74.60% of parallel sentences were affected by issues related to under-segmentation of the corpus. Our hypothesis is that if these words are split into pieces that match English words, the accuracy of the unsupervised WA as well as the translation quality will be improved. To achieve this, we adopt a splitting

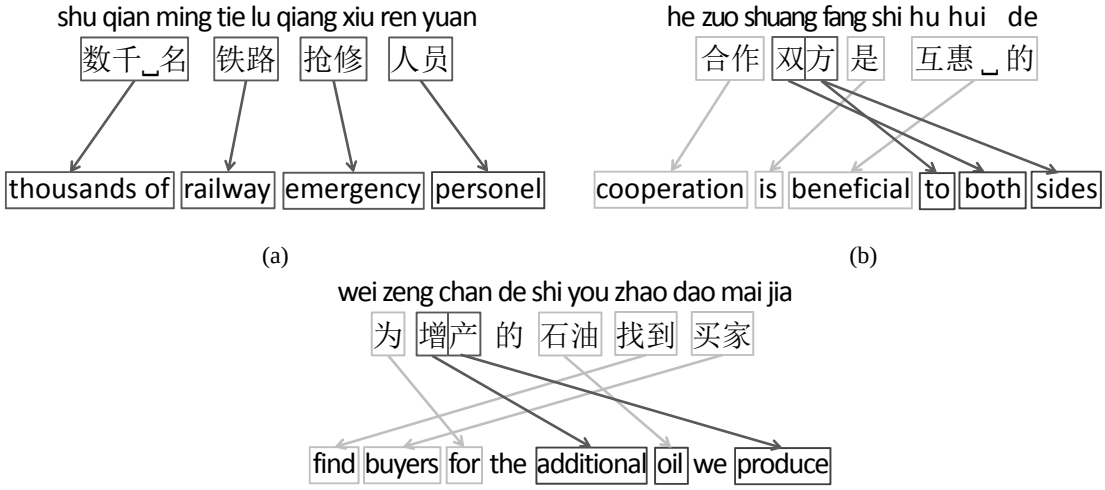


Figure 2: Examples of automated WS on manually WA corpus: (a) Fully consistent; (b) Alignment inconsistent; (c) Alignment inconsistent and extraction hindered. The Chinese words separated by white space are the output of the CTB segmenter. Arrows represent the alignment of atomic blocks. Note that "shuang fang" and "zeng chan" are words produced by the CTB segmenter, but consist of two atomic blocks.

Category	Count	Word Ratio	Sentence Ratio
Fully consistent	355,702	90.74%	25.40%†
Alignment inconsistent	34,464	8.81%	65.54%
Alignment inconsistent & extraction hindered	1,830	0.46%	9.06%
Sum of conflict ‡	36,294	9.26%	74.60%

Table 2: CTB WS on GALE WA corpus: † All words are fully consistent; ‡ Alignment inconsistent plus alignment inconsistent & extraction hindered

strategy, based on a supervised learning approach, to re-segment the corpus. This subsection first formalizes the task, and then presents the approach.

3.1 Word splitting task

The word splitting task is formalized as a sequence labeling task as follows: each word (represented by a sequence of characters $\mathbf{x} = x_1 \dots x_T$ where T is the length of sample) produced by the CTB segmenter is a sample, and a corresponding sequence of binary boundary labels $\mathbf{y} = y_1 \dots y_T$ is the learning target,

$$y_t = \begin{cases} 1 & \text{if there is a split point} \\ & \quad \text{between } c_t \text{ and } c_{t-1}; \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The sequence of boundary labels is derived from the gold WA annotation as follows: for a sequence of two atomic blocks, where the first character of the second block is x_t, then the label $y_t = 1$. Figure 3 presents several samples extracted from the examples in Figure 2.

Input	*Target*	***y***
铁路	铁路	0 0
双方	双␣方	0 1
出版业	出版␣业	0 0 1
增 产	增␣产	0 1

Figure 3: Samples of word splitting task

Each word sample may have no split point, one split point or multiple split points, depending on the gold WA annotation. Table 3 shows the statistics of the word splitting data set which is built from the GALE manual WA corpus and the CTB segmenter's output, where 2000 randomly sampled sentences are taken as a held-out test set.

Set	# Sentences	# Samples	# Split points	# Split points per sample
Train.	16,057	348,086	32,337	0.0929
Test	2,000	43,910	3,929	0.0895

Table 3: Data set for learning the word splitting

3.2 CRF approach

This paper employs a condition random field (CRF) to solve this sequence labeling task (Lafferty et al., 2001). A linear-chain CRF defines the conditional probability of $\mathbf{y}$ given $\mathbf{x}$ as,

$$P_\Lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\mathbf{x}}(\sum_{t=1}^{T}\sum_k \lambda_k f_k(y_{t-1}, y_t, \mathbf{x}, t)), \tag{2}$$

where $\Lambda = \{\lambda_1, \ldots\}$ are parameters, $Z_\mathbf{x}$ is a per-input normalization that makes the probability of all state sequences sum to one; $f_k(y_{t-1}, y_t, \mathbf{x}, t)$ is a feature function which is often a binary-valued sparse feature. The training of CRF model is to maximize the likelihood of training data together with a regularization penalty to avoid over-fitting as (Peng et al., 2004; Peng and McCallum, 2006),

$$\Lambda^* = \underset{\Lambda}{argmax}(\sum_i logP_\Lambda(\mathbf{y}_i|\mathbf{x}_i) - \sum_k \frac{\lambda_k^2}{2\delta_k^2}), \tag{3}$$

where ($\mathbf{x}$,$\mathbf{y}$) are training samples; the hyperparameter δ_k can be understood as the variance of the prior distribution of λ_k. When predicting the labels of test samples, the CRF decoder searches for the optimal label sequence y^* that maximizes the conditional probability,

$$\mathbf{y}^* = \underset{\mathbf{y}}{argmax}\, P_\Lambda(\mathbf{y}|\mathbf{x}). \tag{4}$$

In (Chang et al., 2008) a method is proposed to select an appropriate level of segmentation granularity (in practical terms, to encourage smaller segments). We call their method "length tuner". The following artificial feature is introduced into the learned CRF model:

$$f_0(\mathbf{x}, y_{t-1}, y_t, 1) = \begin{cases} 1 & \text{if } y_t = +1 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The weight λ_0 of this feature is set by hand to bias the output of CRF model. By way of explanation, a very large positive λ_0 will cause every character to be segmented, or conversely a very large negative λ_0 will inhibit the output of segmentation boundaries. In their experiments, $\lambda_0 = 2$ was used to force a CRF segmenter to adopt an intermediate granularity between character and the CTB WS scheme. Compared to the length tuner, our proposed method exploits lexical knowledge about word splitting, and we will therefore refer to it as the "lexical word splitter" or "lexical splitter" for short.

3.3 Feature Set

The features $f_k(y_{t-1}, y_t, \mathbf{x}, t)$ we used include the WS features from the Chinese Stanford word segmenter and a set of extended features described below. The WS features are included because the target split points may share some common characteristics with the boundaries in the CTB WS scheme.

The extended features consists of four types – named entities, word frequency, word length and character-level unsupervised WA. For each type of the feature, the value and value concatenated with previous or current character are taken as sparse features (see Table 4 for details). The real values of word frequency, word length and character-level unsupervised WA are converted into sparse features due to the routine of CRF model.

The character-level unsupervised alignment feature is inspired by the related works of unsupervised bilingual WS (Xu et al., 2008; Chung and Gildea, 2009; Nguyen et al., 2010; Michael et al., 2011). The idea is that the character-level WA can approximately capture the counterpart English expression of each Chinese token, and source tokens aligned to different target expressions should be split into different words (see Figure 4 for an illustration).

The values of the character-level alignment features are obtained through building a dictionary. First, unsupervised WA is performed on the SMT training corpus where the Chinese sentences are treated as sequences of characters; then, the Chinese sentences are segmented by CTB segmenter and a dictionary of segmented words are built; finally, for each word in the dictionary, the relative frequency of being split at a certain position is cal-

Feature	Definition	Example
NE	NE tag of current word	Geography:NE
NE-C_{-1}	NE concatenated with previous character	Geo.-ding:NE-C_{-1}
NE-C_0	NE concatenated with current character	Geo.-mei:NE-C_0
Frequency	Nearest integer of negative logarithm of word frequency	5†:Freq
Freq.-C_{-1}	Frequency concatenated with previous character	5-ding:Freq-C_{-1}
Freq.-C_0	Frequency concatenated with current character	5-mei:Freq-C_0
Length	Length of current word (1,2,3,4,5,6,7 or $\geqslant$7)	4:Len
Len.-Position	Length concatenated with the position	4-2:Len-Pos
Len.-C_{-1}	Length concatenated with previous character	4-ding:Len-C_{-1}
Len.-C_0	Length concatenated with current character	4-mei:Len-C_0
Char. Align.	Five-level relative frequency of being split	0.4‡:CA
C.A.-C_{-1}	C.A. concatenated with previous character	0.4-ding:CA-C_{-1}
C.A.-C_0	C.A. concatenated with current character	0.4-mei:CA-C_0

Table 4: Extended features used in the CRF model for word splitting. The example shows the features used in the decision whether to split the Chinese word "la ding mei zhou" (Latin America, the first four Chinese characters in Figure 4) after the second Chinese character. † Round(-$\log_{10}$(0.00019)); ‡ Round(0.43 $\times$ 5) / 5

Figure 4: Illustration of character-level unsupervised alignment features. The dotted lines are word boundaries suggested by the alignment.

culated as,

$$f_{CA}(w,i) = \frac{n_i}{n_w} \quad (6)$$

where w is a word, i is a splitting position (from 1 to the length of w minus 1); n_i is the number of times the words as split at position i according to the character-level alignment, that is, the character before and after i are aligned to different English expressions; n_w is occurrence count of word w in the training corpus.

4 Experiments

In the last section we found that 9.26% of words produced by the CTB segmenter have the potential to cause problems for SMT, and propose a lexical word splitter to address this issue through segmentation refinement. This section contains experiments designed to empirically evaluate the proposed lexical word splitter in three aspects: first, whether the WS accuracy is improved; second, whether the accuracy of the unsupervised WA during training SMT systems is improved; third, whether the end-to-end translation quality is improved.

This section first describes the experimental methodology, then presents the experimental results, and finally illustrates the operation of our proposed method using a real example.

4.1 Experimental Methodology

4.1.1 Experimental Corpora

The GALE manual WA corpus and the Chinese to English corpus from the shared task of the NIST open machine translation (OpenMT) 2006 evaluation [6] were employed as the experimental corpus (Table 5).

The experimental corpus for WS was constructed by first segmenting 2000 held out sentences from the GALE manual WA corpus with the Stanford segmenter, and then refining the segmentation with the gold alignment annotation. For example, the gold segmentation for the examples in Figure 2 is presented in Figure 5. Note that this test corpus is intended to represent an oracle segmentation for our proposed method, and serves primarily to gauge the improvement of our method over the baseline Stanford segmenter, relative to an upper bound.

[6] http://www.itl.nist.gov/iad/mig/tests/mt/2006/

数千 ␣ 名 ␣ 铁路 ␣ 抢修 ␣ 人员

合作 ␣ 双 ␣ 方 ␣ 是 ␣ 互惠 ␣ 的

中国 ␣ 出版 ␣ 业 ␣ 发展

为 ␣ 增 ␣ 产 ␣ 的 ␣ 石油 ␣ 找到 ␣ 买家

Figure 5: Examples of gold WS for evaluation

Set	# sent. pairs	# CN tokens	# EN tokens
Train.	442,967	19,755,573	13,444,927
Eval02	878†	38,204	105,944
Eval03	919†	40,900	113,970
Eval04	1,597†	71,890	207,279
Eval05	1,082†	50,461	138,952
Eval06	1,664†	62,422	189,059

Table 5: NIST Open machine translation 2006 Corpora. † Number of sentence samples which contain one Chinese sentence and four English reference sentences.

The experimental corpus for unsupervised WA was the union set of the NIST OpenMT training set and the 2000 test sentence pairs from GALE WA corpus. We removed the United Nations corpus from the NIST OpenMT constraint training resources because it is out of domain.

The main result of this paper is the evaluation of the end-to-end performance of an SMT system. The experimental corpus for this task was the NIST OpenMT corpus. The data set of the NIST evaluation 2002 was used as a development set for MERT tuning (Och, 2003), and the remaining data sets of the NIST evaluation from 2003 to 2006 were used as test sets. The English sentences were tokenized by Stanford toolkit [7] and converted to lowercase.

4.1.2 Evaluation

The performance of WS was measured by precision, recall and F_1 of gold words (Sproat and Emerson, 2003),

The performance of unsupervised WA in the SMT training procedure was measured through alignment error rate (AER)(Och and Ney, 2000; Liang et al., 2006). Sure alignment edges and possible alignment edges were not distinguished in this paper as no such tags are found in GALE manual WA corpus.

The performance of SMT was measured using BLEU (Papineni et al., 2002).

[7] http://nlp.stanford.edu/software/corenlp.shtml

4.1.3 Baseline Methods

Two Chinese WS methods were taken as the baseline methods in this paper. One method was the CTB segmenter, that is, Stanford Chinese word segmenter with the model trained on CTB corpus. The other method was the length tuner in (Chang et al., 2008), which added a constant into the confidence scores of a trained CRF word segmenter to encourage it to output more word boundaries (see Section 3.2 for details).

4.1.4 Implementation and Parameter settings

The proposed lexical word splitter was implemented on the CRF model toolkit released with the Stanford segmenter (Tseng et al., 2005). The regularity parameters δ_k are set to be 3, the same as the Stanford segmenter, because no significant performance improvements were observed by tuning that parameter.

To extract features for the word splitter, the Stanford named entity recognizer (Finkel et al., 2005)[8] was employed to obtain the tags of named entities. Word frequencies were caculated from the source side of SMT training corpus. The character-level unsupervised alignment was conducted using GIZA++ (Och and Ney, 2003)[9].

The length tuner reused the CRF model of CTB segmenter. The parameter λ_0 was tuned through the grid search in (Chang et al., 2008), that is, observing the BLEU score on the SMT development set varing from $\lambda_0 = 0$ to $\lambda_0 = 32$. The grid search showed that $\lambda_0 = 2$ was optimal, agreeing with the value in (Chang et al., 2008).

Moses (Koehn et al., 2007)[10], a state-of-the-art phrase-based SMT system, was employed to perform end-to-end SMT experiments. GIZA++ was employed to perform unsupervised WA.

4.2 Experimental Results

4.2.1 Word Segmentation

The WS performance of CTB segmenter, length tuner and the proposed lexical splitter are presented in Table 6. The proposed method achieves the highest scores on all the criterion of F_1, precision and recall. The length tuner outperforms the CTB segmenter in terms of recall, but with lower precision.

[8] http://nlp.stanford.edu/software/CRF-NER.shtml

[9] http://www.statmt.org/moses/giza/GIZA++.html

[10] http://www.statmt.org/moses/

WS	F_1	Prec.	Recall
CTB segmenter	0.878	0.917	0.842
Length tuner	0.873	0.894	0.852
Lexical splitter	**0.915**	**0.922**	**0.908**

Table 6: Performance of WS

WS	AER	Prec.	Recall
CTB segmenter	0.425	0.622	0.534
Length tuner	0.417	0.642	0.535
Lexical splitter	**0.396**	**0.674**	**0.547**

Table 7: Performance of unsupervised WA using different WS strategies

4.2.2 Word Alignment

The WA performance of the CTB segmenter, length tuner and the proposed lexical spliter is presented in Table 7. Both lexical splitter and length tuner outperform the CTB segmenter, indicating the splitting words into smaller pieces can improve the accuracy of unsupervised WA. This result supports the finding in (Chang et al., 2008) that the segment size from CTB WS is too large for SMT. In addition, the proposed lexical splitter significantly outperforms the length tuner.

4.2.3 Machine Translation

The end-to-end SMT performance of CTB segmenter, length tuner and the proposed lexical spliter are presented in Table 8. Each experiment was performed three times, and the average BLEU and standard derivation were calculated, because there is randomness in the results from MERT. The proposed lexical splitter outperformed the two baselines on all the test sets, and achieves an average improvement of 0.63 BLEU percentage points, indicating that the proposed method can effectively improve the translation quality. The length tuner also outperforms the CTB segmenter, but the average improvement is 0.15 BLEU percentage points, much less than the proposed methods.

4.3 Analysis

Figure 6 presents an example from the test corpus, which demonstrates how the proposed lexical splitter splits words more accurately than the baseline length tuner method. Two words in the segmentation result of the CTB segmenter are worthy of attention. The first one is “yang nian”(the year of goat), the lexical splitter split this word and got the right translation, while the length tuner did not split it. The second is “rong jing”(booming or prosperity), the length tuner split this word, which resulted in wrong translations, while the lexical splitter avoided this mistake.

5 Comparison to Related Work

The most similar work in the literature to the proposed method is the the length tuner method proposed by (Chang et al., 2008). This method also encourages the generation of more words during segmentation by using a single parameter that can be use to control segment length. Our method differs from theirs in that it is able to acquire vocabulary knowledge from word alignments that can be used to more accurately split words into segments suitable for machine translation.

There is large volume of research using bilingual unsupervised and semi-supervised WS to address the problem of optimizing WS for SMT (Xu et al., 2008; Chung and Gildea, 2009; Nguyen et al., 2010; Michael et al., 2011). The main difference with our approach is that they use automatic WA results, most often obtained using the same tools as are used in training SMT systems. One of the main problems of using unsupervised WA is that it is noisy, and therefore, employing iterative optimization methods to refine the results of unsupervised WA is a key issue in their research, for example boosting (Ma and Way, 2009; Michael et al., 2011), expectation maximization (Chung and Gildea, 2009), Bayesian sampling (Xu et al., 2008; Nguyen et al., 2010), or heuristic search (Zhao et al., 2013). Nevertheless, noisy WA makes both analyzing WS and improving SMT quality quite hard. In contrast, by using manual WA, we can clearly analyze the segmentation problems (Section 2), and train supervised models to solve the problem (Section 3).

As far as we are aware, among related work on WS, our method achieves the highest BLEU improvement relative to the start-of-the-art WS – the Stanford Chinese word segmenter – on the Chinese-English OpenMT corpora. The methods proposed in (Ma and Way, 2009; Chung and Gildea, 2009) fail to outperform the Stanford Chinese word segmenter on Chinese-English OpenMT corpora. The length tuner method proposed in (Chang et al., 2008) is less effective to ours according to the experimental results in this paper.

WS	eval03	eval04	eval05	eval06	improve
CTB segmenter	31.89 ± 0.09	32.73 ± 0.19	31.03 ± 0.16	31.38 ± 0.23	
Length tuner	32.06 ± 0.07	32.74 ± 0.10	31.34 ± 0.11	31.50 ± 0.11	0.15 ± 0.12
Lexical splitter	**32.55** ± **0.18**	**32.94** ± **0.11**	**31.87** ± **0.15**	**32.17** ± **0.35**	**0.63** ± **0.29**

Table 8: Performance (BLEU) of SMT

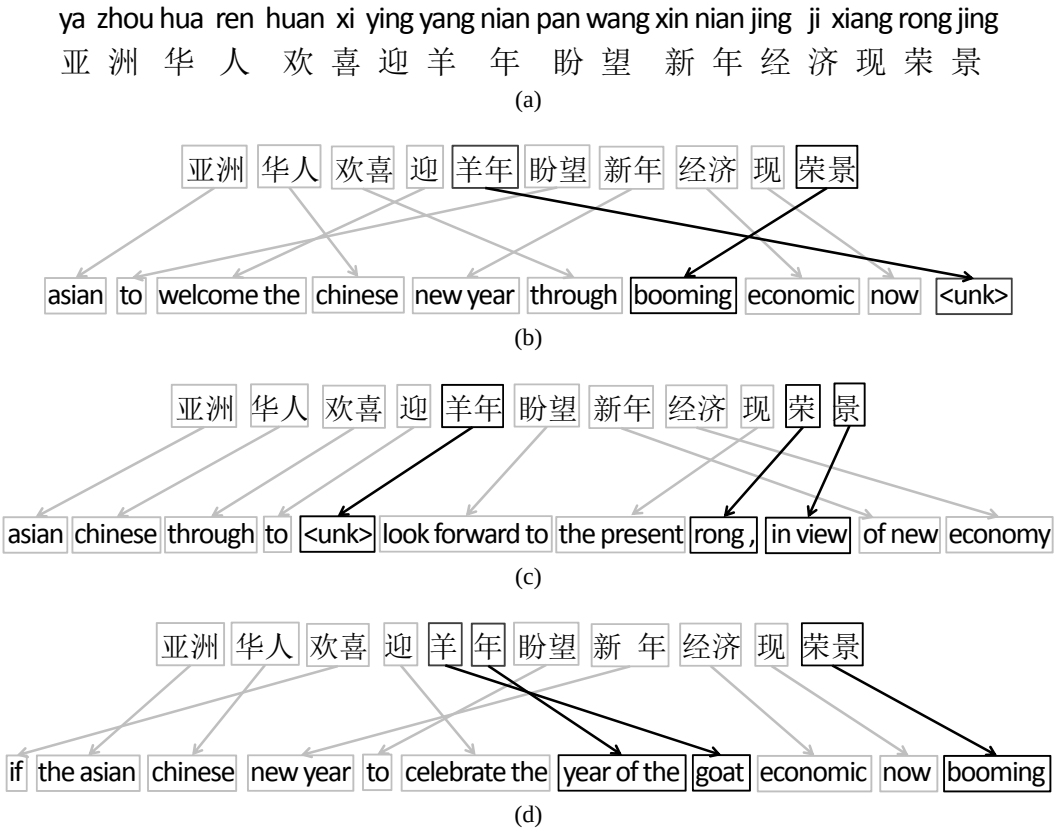


Figure 6: Example of SMT from test sets. (a) source; (b) CTB segmenter; (c) length tuner; (d) lexical splitter. The four gold references are: “ethnic chinese in asia celebrate year of goat and hope for economic prosperity in new year”, “ asian chinese celebrate the arrival of the year of sheep and wish a prosperous new year”, “ asian chinese happily welcome the year of goat , expecting economic prosperity in new year”,“asian chinese happily welcomed year of the goat , praying for prosperity in the new year”

6 Conclusion

This paper is concerned with the role of word segmentation in Chinese-to-English SMT. We explored the use of a manually annotated word alignment corpus to refine word segmentation for machine translation. Based on an initial finding that 74.60% of running sentences in the WA corpus have segmentation inconsistent with a gold WA annotation, we proposed a supervised lexical resegmentation model to modify the WS in order to relieve these issues.

Our main experimental results show that the proposed approach is capable of improving both alignment quality and end-to-end translation quality. The proposed method achieved the highest BLEU score relative to a number of respectable baseline systems that included the Stanford word segmenter, and an improved Stanford word segmenter that could be tuned for segment length. No language-specific techniques other than a manually aligned corpus were employed in this paper, thus the approach can applied to other SMT language pairs that require WS.

In the future, we plan to explore combining multiple source words which are aligned to the same target words. This is the symmetric topic of the post word splitting which is studied in this paper. The effect of this word combination oper-

ation on SMT is non-trivial. On one hand, it can reduce the ambiguity in the source side. On the other hand, it may cause sparseness problems.

Acknowledgements

We thank the three reviewers for their valuable comments. We also thank the Stanford natural language processing group for releasing the source codes of their word segmenter.

References

Pi-Chuan Chang, Michel Galley, and Christopher D Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the 3rd Workshop on Statistical Machine Translation*, pages 224–232. Association for Computational Linguistics.

Tagyoung Chung and Daniel Gildea. 2009. Unsupervised tokenization for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 718–726. Association for Computational Linguistics.

John DeNero and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *ACL*, volume 45, page 17.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin K Tsou. 2011. Overview of the patent machine translation task at the NTCIR-9 workshop. In *Proceedings of NTCIR*, volume 9, pages 559–578.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised itg models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 923–931. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289. Association for Computing Machinery.

Xuansong Li, Niyu Ge, Stephen Grimes, Stephanie Strassel, and Kazuaki Maeda. 2010. Enriching word alignment with linguistic tags. In *LREC*, pages 2189–2195.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 104–111. Association for Computational Linguistics.

Yanjun Ma and Andy Way. 2009. Bilingually motivated domain-adapted word segmentation for statistical machine translation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 549–557. Association for Computational Linguistics.

Paul Michael, Andrew Finch, and Eiichiro Sumita. 2011. Integration of multiple bilingually-trained segmentation schemes into statistical machine translation. *IEICE transactions on information and systems*, 94(3):690–697.

Rada Mihalcea and Ted Pedersen. 2003. An evaluation exercise for word alignment. In Rada Mihalcea and Ted Pedersen, editors, *HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 1–10, Edmonton, Alberta, Canada, May 31. Association for Computational Linguistics.

ThuyLinh Nguyen, Stephan Vogel, and Noah A Smith. 2010. Nonparametric word segmentation for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 815–823. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2000. A comparison of alignment models for statistical machine translation. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 1086–1090. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Fuchun Peng and Andrew McCallum. 2006. Information extraction from research papers using conditional random fields. *Information Processing & Management*, 42(4):963–979.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. *Computer Science Department Faculty Publication Series*.

Richard Sproat and Thomas Emerson. 2003. The first international chinese word segmentation bakeoff. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*, pages 133–143. Association for Computational Linguistics.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for SIGHAN Bakeoff 2005. In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*, volume 171. Jeju Island, Korea.

Jia Xu, Jianfeng Gao, Kristina Toutanova, and Hermann Ney. 2008. Bayesian semi-supervised Chinese word segmentation for statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 1017–1024. Association for Computational Linguistics.

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated chinese corpus. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–8. Association for Computational Linguistics.

Richard Zens, Franz Josef Och, and Hermann Ney. 2002. Phrase-based statistical machine translation. In *Advances in Artificial Intelligence*, pages 18–32. Springer.

Hai Zhao, Masao Utiyama, Eiichro Sumita, and Bao-Liang Lu. 2013. An empirical study on word segmentation for chinese machine translation. *A. Gelbukh (Ed.): CICLing 2013, Part II, LNCS 7817*, pages 248–263.

Improving Pivot-Based Statistical Machine Translation by Pivoting the Co-occurrence Count of Phrase Pairs

Xiaoning Zhu[1*], Zhongjun He[2], Hua Wu[2], Conghui Zhu[1], Haifeng Wang[2], and Tiejun Zhao[1]
Harbin Institute of Technology, Harbin, China[1]
{xnzhu,chzhu,tjzhao}@mtlab.hit.edu.cn
Baidu Inc., Beijing, China[2]
{hezhongjun,wu_hua,wanghaifeng}@baidu.com

Abstract

To overcome the scarceness of bilingual corpora for some language pairs in machine translation, pivot-based SMT uses pivot language as a "bridge" to generate source-target translation from source-pivot and pivot-target translation. One of the key issues is to estimate the probabilities for the generated phrase pairs. In this paper, we present a novel approach to calculate the translation probability by pivoting the co-occurrence count of source-pivot and pivot-target phrase pairs. Experimental results on Europarl data and web data show that our method leads to significant improvements over the baseline systems.

1 Introduction

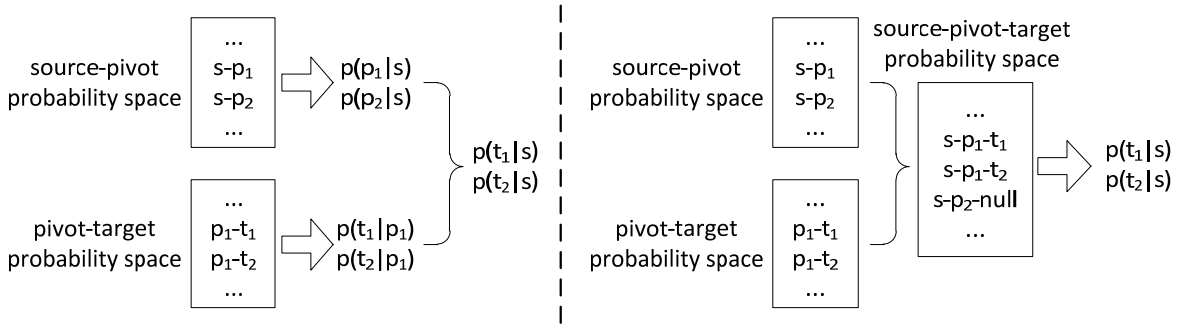
Statistical Machine Translation (SMT) relies on large bilingual parallel data to produce high quality translation results. Unfortunately, for some language pairs, large bilingual corpora are not readily available. To alleviate the parallel data scarceness, a conventional solution is to introduce a “bridge” language (named pivot language) to connect the source and target language (de Gispert and Marino, 2006; Utiyama and Isahara, 2007; Wu and Wang, 2007; Bertoldi et al., 2008; Paul et al., 2011; El Kholy et al., 2013; Zahabi et al., 2013), where there are large amounts of source-pivot and pivot-target parallel corpora.

Among various pivot-based approaches, the triangulation method (Cohn and Lapata, 2007; Wu and Wang, 2007) is a representative work in pivot-based machine translation. The approach proposes to build a source-target phrase table by merging the source-pivot and pivot-target phrase table. One of the key issues in this method is to estimate the translation probabilities for the generated source-target phrase pairs. Conventionally, the probabilities are estimated by multiplying the posterior probabilities of source-pivot and pivot-target phrase pairs. However, it has been shown that the generated probabilities are not accurate enough (Cui et al., 2013). One possible reason may lie in the non-uniformity of the probability space. Through Figure 1. (a), we can see that the probability distributions of source-pivot and pivot-target language are calculated separately, and the source-target probability distributions are induced from the source-pivot and pivot-target probability distributions. Because of the absence of the pivot language (e.g., p2 is in source-pivot probability space but not in pivot-target one), the induced source-target probability distribution is not complete, which will result in inaccurate probabilities.

To solve this problem, we propose a novel approach that utilizes the co-occurrence count of source-target phrase pairs to estimate phrase translation probabilities more precisely. Different from the triangulation method, which merges the source-pivot and pivot-target phrase pairs after training the translation model, we propose to merge the source-pivot and pivot-target phrase pairs immediately after the phrase extraction step, and estimate the co-occurrence count of the source-pivot-target phrase pairs. Finally, we compute the translation probabilities according to the estimated co-occurrence counts, using the standard training method in phrase-based SMT (Koehn et al., 2003). As Figure 1. (b) shows, the

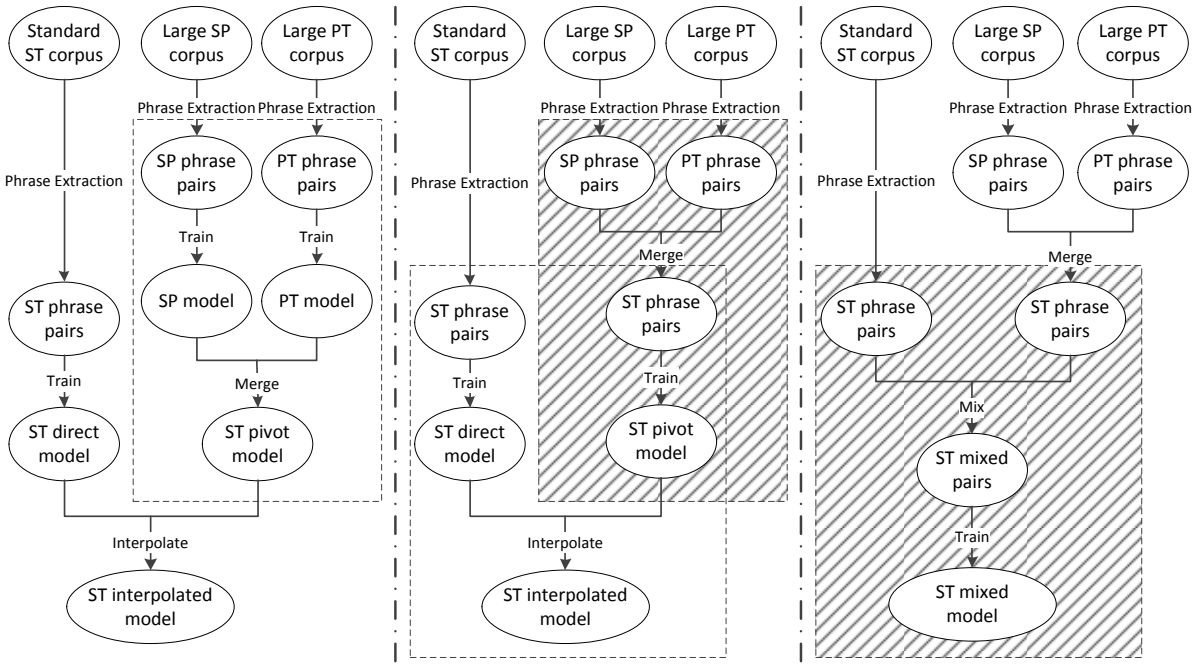
[*] This work was done when the first author was visiting Baidu.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1665–1675,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics



(a) the *triangulation* method (b) the *co-occurrence count* method

Figure 1: An example of probability space evolution in pivot translation.



(a) the *triangulation* method (b) the *co-occurrence count* method (c) the *mixed* model

Figure 2: Framework of the triangulation method, the co-occurrence count method and the mixed model. The shaded box in (b) denotes difference between the co-occurrence count method and the triangulation method. The shaded box in (c) denotes the difference between the interpolation model and the mixed model.

source-target probability distributions are calculated in a complete probability space. Thus, it will be more accurate than the traditional triangulation method. Figure 2. (a) and (b) show the difference between the triangulation method and our co-occurrence count method.

Furthermore, it is common that a small standard bilingual corpus can be available between the source and target language. The direct translation model trained with the standard bilingual corpus exceeds in translation performance, but its weakness lies in low phrase coverage. However, the pivot model has characteristics characters. Thus, it is important to combine the direct and pivot translation model to compensate mutually and further improve the translation performance. To deal with this problem, we propose a mixed model by merging the phrase pairs extracted by pivot-based method and the phrase pairs extracted from the standard bilingual corpus. Note that, this is different from the conventional interpolation method, which interpolates the direct and pivot translation model. See Figure 2. (b) and (c) for further illustration.

The remainder of this paper is organized as follows. In Section 2, we describe the related work. We introduce the co-occurrence count method in Section 3, and the mixed model in Section 4. In Section 5 and Section 6, we describe and analyze the experiments. Section 7 gives a conclusion of the paper.

2 Related Work

Several methods have been proposed for pivot-based translation. Typically, they can be classified into 3 kinds as follows:

Transfer Method: The transfer method (Utiyama and Isahara, 2007; Wang et al., 2008; Costa-jussà et al., 2011) connects two translation systems: a source-pivot MT system and a pivot-target MT system. Given a source sentence, (1) the source-pivot MT system translates it into the pivot language, (2) and the pivot-target MT system translates the pivot sentence into the target sentence. During each step (source to pivot and pivot to target), multiple translation outputs will be generated, thus a minimum Bayes-risk system combination method is often used to select the optimal sentence (González-Rubio et al., 2011; Duh et al., 2011). The problem with the transfer method is that it needs to decode twice. On one hand, the time cost is doubled; on the other hand, the translation error of the source-pivot translation system will be transferred to the pivot-target translation.

Synthetic Method: It aims to create a synthetic source-target corpus by: (1) translate the pivot part in source-pivot corpus into target language with a pivot-target model; (2) translate the pivot part in pivot-target corpus into source language with a pivot-source model; (3) combine the source sentences with translated target sentences or/and combine the target sentences with translated source sentences (Utiyama et al., 2008; Wu and Wang, 2009). However, it is difficult to build a high quality translation system with a corpus created by a machine translation system.

Triangulation Method: The triangulation method obtains source-target phrase table by merging source-pivot and pivot-target phrase table entries with identical pivot language phrases and multiplying corresponding posterior probabilities (Wu and Wang, 2007; Cohn and Lapata, 2007), which has been shown to work better than the other pivot approaches (Utiyama and Isahara, 2007). A problem of this approach is that the probability space of the source-target phrase pairs is non-uniformity due to the mismatching of the pivot phrase.

3 Our Approach

In this section, we will introduce our method for learning a source-target phrase translation model with a pivot language as a bridge. We extract the co-occurrence count of phrase pairs for each language pair with a source-pivot and a pivot-target corpus. Then we generate the source-target phrase pairs with induced co-occurrence information. Finally, we compute translation probabilities using the standard phrase-based SMT training method.

3.1 Phrase Translation Probabilities

Following the standard phrase extraction method (Koehn et al., 2003), we can extract phrase pairs $(\bar{s}, \bar{p})$ and $(\bar{p}, \bar{t})$ from the corresponding word-aligned source-pivot and pivot-target training corpus, where $\bar{s}$, $\bar{p}$ and $\bar{t}$ denotes the phrase in source, pivot and target language respectively.

Formally, given the co-occurrence count $c(\bar{s}, \bar{p})$ and $c(\bar{p}, \bar{t})$, we can estimate $c(\bar{s}, \bar{t})$ by Equation 1:

$$c(\bar{s}, \bar{t}) = \sum_{\bar{p}} g(c(\bar{s}, \bar{p}), c(\bar{p}, \bar{t})) \quad (1)$$

where $g(\cdot)$ is a function to merge the co-occurrences count $c(\bar{s}, \bar{p})$ and $c(\bar{p}, \bar{t})$. We propose four calculation methods for function $g(\cdot)$.

Given the co-occurrence count $c(\bar{s}, \bar{p})$ and $c(\bar{p}, \bar{t})$, we first need to induce the co-occurrence count $c(\bar{s}, \bar{p}, \bar{t})$. The $c(\bar{s}, \bar{p}, \bar{t})$ is counted when the source phrase, pivot phrase and target phrase occurred together, thus we can infer that $c(\bar{s}, \bar{p}, \bar{t})$ is smaller than $c(\bar{s}, \bar{p})$ and $c(\bar{p}, \bar{t})$. In this circumstance, we consider that $c(\bar{s}, \bar{p}, \bar{t})$ is approximately equal to the minimum value of $c(\bar{s}, \bar{p})$ and $c(\bar{p}, \bar{t})$, as shown in Equation 2.

$$c(\bar{s}, \bar{p}, \bar{t}) \approx \sum_{\bar{p}} \min(c(\bar{s}, \bar{p}), c(\bar{p}, \bar{t})) \quad (2)$$

Because the co-occurrence count of source-target phrase pairs needs the existence of pivot phrase $\bar{p}$, we intuitively believe that the co-occurrence count $c(\bar{s}, \bar{t})$ is equal to the co-occurrence count $c(\bar{s}, \bar{p}, \bar{t})$. Under this assumption, we can obtain the co-occurrence count $c(\bar{s}, \bar{t})$ as shown in Equation 3. Furthermore, to testify our assumption, we also try the *maximum* value (Equation 4) to infer the co-occurrence count of $(\bar{s}, \bar{t})$ phrase pair.

$$c(\bar{s},\bar{t}) = \sum_{\bar{p}} \min(c(\bar{s},\bar{p}), c(\bar{p},\bar{t})) \quad (3)$$

$$c(\bar{s},\bar{t}) = \sum_{\bar{p}} \max(c(\bar{s},\bar{p}), c(\bar{p},\bar{t})) \quad (4)$$

In addition, if source-pivot and pivot-target parallel corpus greatly differ in quantities, then the *minimum* function would likely just take the counts from the smaller corpus. To deal with the problem of the imbalance of the parallel corpora, we also try the *arithmetic mean* (Equation 5) and *geometric mean* (Equation 6) function to infer the co-occurrence count of source-target phrase pairs.

$$c(\bar{s},\bar{t}) = \sum_{\bar{p}} (c(\bar{s},\bar{p}) + c(\bar{p},\bar{t}))/2 \quad (5)$$

$$c(\bar{s},\bar{t}) = \sum_{\bar{p}} \sqrt{c(\bar{s},\bar{p}) \times c(\bar{p},\bar{t})} \quad (6)$$

When the co-occurrence count of source-target language is calculated, we can estimate the phrase translation probabilities with the following Equation 7 and Equation 8.

$$\phi(\bar{s}|\bar{t}) = \frac{c(\bar{s},\bar{t})}{\sum_{\bar{s}} c(\bar{s},\bar{t})} \quad (7)$$

$$\varphi(\bar{t}|\bar{s}) = \frac{c(\bar{s},\bar{t})}{\sum_{\bar{t}} c(\bar{s},\bar{t})} \quad (8)$$

3.2 Lexical Weight

Given a phrase pair $(\bar{s},\bar{t})$ and a word alignment a between the source word positions $i = 1,\cdots,n$ and the target word positions $j = 0,\cdots,m$, the lexical weight of phrase pair $(\bar{s},\bar{t})$ can be calculated by the following Equation 9 (Koehn et al., 2003).

$$p_\omega(\bar{s}|\bar{t},a) = \prod_{i=1}^{n} \frac{1}{|\{j|(i,j)\in a\}|} \sum_{\forall(i,j)\in a} \omega(s_i|t_j) \quad (9)$$

The lexical translation probability distribution $\omega(s|t)$ between source word *s* and target word *t* can be estimated with Equation 10.

$$\omega(s|t) = \frac{c(s,t)}{\sum_{s'} c(s',t)} \quad (10)$$

To compute the lexical weight for a phrase pair $(\bar{s},\bar{t})$ generated by $(\bar{s},\bar{p})$ and $(\bar{p},\bar{t})$, we need the alignment information a, which can be obtained as Equation 11 shows.

$$a = \{(s,t)|\exists p\colon (s,p)\in a_1 \& (p,t)\in a_2\} \quad (11)$$

where a_1 and a_2 indicate the word alignment information in the phrase pair $(\bar{s},\bar{p})$ and $(\bar{p},\bar{t})$ respectively.

4 Integrate with Direct Translation

If a standard source-target bilingual corpus is available, we can train a direct translation model. Thus we can integrate the direct model and the pivot model to obtain further improvements. We propose a mixed model by merging the co-occurrence count in direct translation and pivot translation. Besides, we also employ an interpolated model (Wu and Wang, 2007) by merging the direct translation model and pivot translation model using a linear interpolation.

4.1 Mixed Model

Given n pivot languages, the co-occurrence count can be estimated using the method described in Section 3.1. Then the co-occurrence count and the lexical weight of the mixed model can be estimated with the following Equation 12 and 13.

$$c(s,t) = \sum_{i=0}^{n} c_i(s,t) \quad (12)$$

$$p_\omega(\bar{s}|\bar{t},a) = \sum_{i=0}^{n} \alpha_i\, p_{\omega,i}(\bar{s}|\bar{t},a) \quad (13)$$

where $c_0(s,t)$ and $p_{\omega,0}(\bar{s}|\bar{t},a)$ are the co-occurrence count and lexical weight in the direct translation model respectively. $c_i(s,t)$ and $p_{\omega,i}(\bar{s}|\bar{t},a)$ denote the co-occurrence count and lexical weight in the pivot translation model. α_i is the interpolation coefficient, requiring $\sum_{i=0}^{n} \alpha_i = 1$.

4.2 Interpolated Model

Following Wu and Wang (2007), the interpolated model can be modelled with Equation 14.

$$\phi(\bar{s}|\bar{t}) = \sum_{i=0}^{n} \beta_i \phi_i(\bar{s}|\bar{t}) \quad (14)$$

where $\phi_0(\bar{s}|\bar{t})$ is the phrase translation probability in direct translation model; $\phi_i(\bar{s}|\bar{t})$ is the phrase translation probability in pivot translation model. The lexical weight is obtained with Equation 13. β_i is the interpolation coefficient, requiring $\sum_{i=0}^{n} \beta_i = 1$.

Language Pairs	Sentence Pairs	Source Words	Target Words
de-en	1.9M	48.5M	50.9M
es-en	1.9M	54M	51.7M
fr-en	2M	58.1M	52.4M

Table 1: Training data of Europarl corpus

System	BLEU%					
	de-es	de-fr	es-de	es-fr	fr-de	fr-es
Baseline	27.04	23.01	20.65	33.84	20.87	38.31
Minimum	**27.93***	**23.94***	**21.52***	**35.38***	**21.48***	**39.62***
Maximum	25.70	21.59	20.26	32.58	20.50	37.30
Arithmetic mean	26.01	22.24	20.13	33.38	20.37	37.37
Geometric mean	27.31	**23.49***	**21.10***	**34.76***	**21.15***	**39.19***

Table 2: Comparison of different merging methods on in-domain test set. * indicates the results are significantly better than the baseline ($p<0.05$).

System	BLEU%					
	de-es	de-fr	es-de	es-fr	fr-de	fr-es
Baseline	15.34	13.52	11.47	21.99	12.19	25.00
Minimum	**15.77***	**14.08***	**11.99***	**23.90***	**12.55***	**27.05***
Maximum	13.41	11.83	10.17	20.48	10.83	22.75
Arithmetic mean	13.96	12.10	10.57	21.07	11.30	23.70
Geometric mean	15.09	13.30	11.52	**23.32***	**12.46***	**26.22***

Table 3: Comparison of different merging methods on out-of-domain test set.

5 Experiments on Europarl Corpus

Our first experiment is carried out on Europarl[1] corpus, which is a multi-lingual corpus including 21 European languages (Koehn, 2005). In our work, we perform translations among French (fr), German (de) and Spanish (es). Due to the richness of available language resources, we choose English (en) as the pivot language. Table 1 summarized the statistics of training data. For the language model, the same monolingual data extracted from the Europarl are used.

The word alignment is obtained by GIZA++ (Och and Ney, 2000) and the heuristics "grow-diag-final" refinement rule (Koehn et al., 2003). Our translation system is an in-house phrase-based system analogous to Moses (Koehn et al., 2007). The baseline system is the triangulation method (Wu and Wang, 2007), including an interpolated model which linearly interpolate the direct and pivot translation model.

We use WMT08[2] as our test data, which contains 2000 in-domain sentences and 2051 out-of-domain sentences with single reference. The translation results are evaluated by case-insensitive BLEU-4 metric (Papineni et al., 2002). The statistical significance tests using 95% confidence interval are measured with paired bootstrap resampling (Koehn, 2004).

5.1 Results

We compare 4 merging methods with the baseline system. The results are shown in Table 2 and Table 3. We find that the *minimum* method outperforms the others, achieving significant improvements over the baseline on all translation directions. The absolute improvements range from 0.61 (fr-de) to 1.54 (es-fr) in BLEU% score on in-domain test data, and range from 0.36 (fr-de) to 2.05 (fr-es) in BLEU% score on out-of-domain test data. This indicates that our method is effective and robust in general.

[1] http://www.statmt.org/europarl

[2] http://www.statmt.org/wmt08/shared-task.html

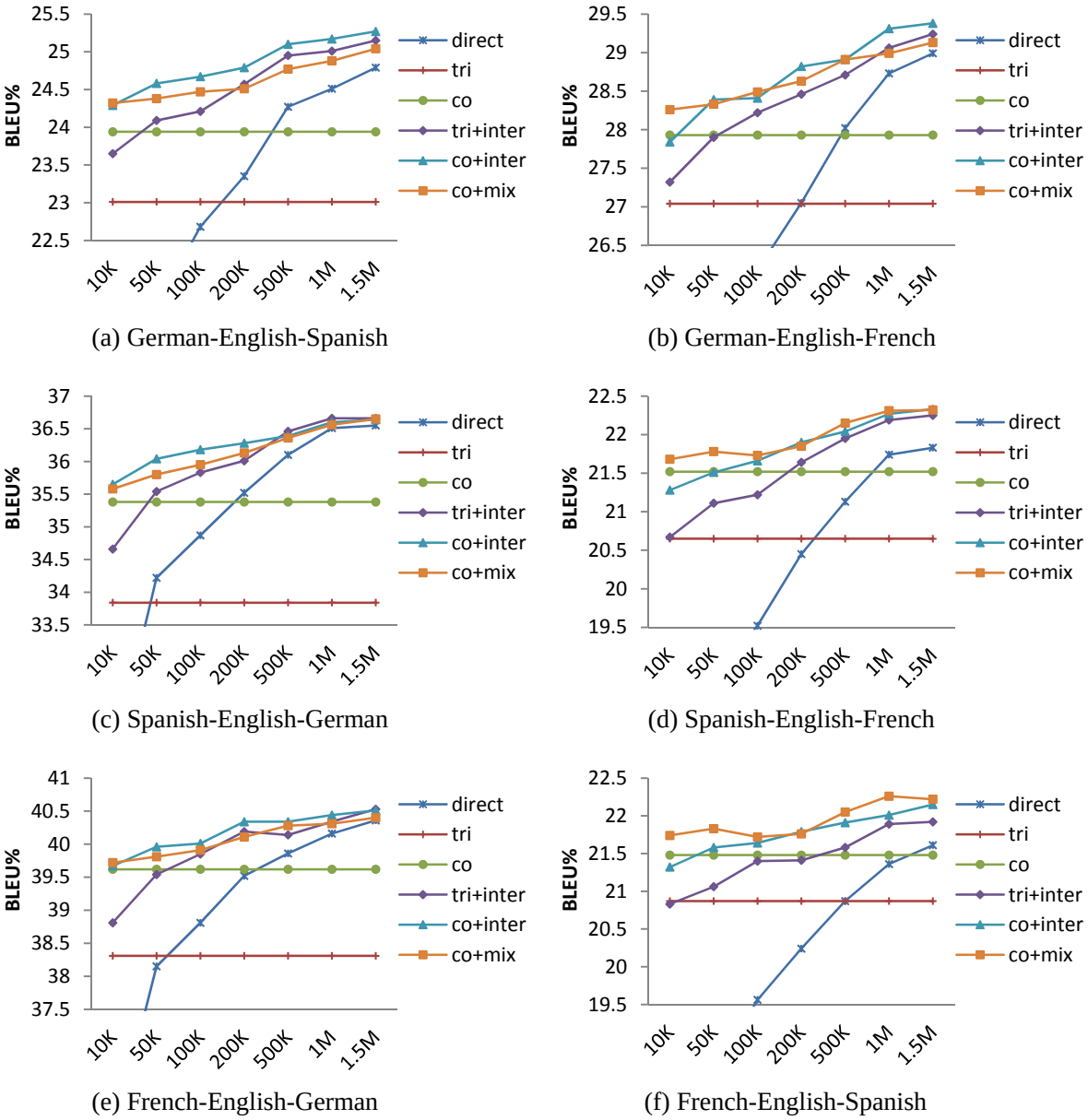


Figure 3: Comparisons of pivot-based methods on different scales of source-target standard corpora. (*direct*: direct model; *tri*: triangulation model; *co*: co-occurrence count model; *tri+inter*: triangulation model interpolated with direct model ; *co+inter*: co-occurrence count model interpolated with direct model; *co+mix*: mixed model). X-axis represents the scale of the standard training data.

The *geometric mean* method also achieves improvement, but not as significant as the *minimum* method. However, the *maximum* and the *arithmetic mean* methods show a decrement in BLEU scores. This reminds us that how to choose a proper merging function for the co-occurrence count is a key problem. In the future, we will explore more sophisticated method to merge co-occurrence count.

5.2 Analysis

The pivot-based translation is suitable for the scenario that there exists large amount of source-pivot and pivot-target bilingual corpora and only a little source-target bilingual data. Thus, we randomly select 10K, 50K, 100K, 200K, 500K, 1M, 1.5M sentence pairs from the source-target bilingual corpora to simulate the lack of source-target data. With these corpora, we train several direct translation models with different scales of bilingual data. We interpolate each direct translation model with the pivot model (both triangulation method and co-occurrence count method) to obtain the interpolated model respectively. We also mix the direct model and pivot model using the method described in Section 4.1. Following

Wu and Wang (2007), we set $\alpha_0 = 0.9$, $\alpha_1 = 0.1$, $\beta_0 = 0.9$ and $\beta_1 = 0.1$ empirically. The experiments are carried out on 6 translation directions: German-Spanish, German-French, Spanish-German, Spanish-French, French-German and French-Spanish. The results are shown in Figure 3. We only list the results on in-domain test sets. The trend of the results on out-of domain test sets is similar with in-domain test sets.

The results are explained as follows:

(1) Comparison of Pivot Translation and Direct Translation

The pivot translation models are better than the direct translation models trained on a small source-target bilingual corpus. With the increment of source-target corpus, the direct model first outperforms the triangulation model and then outperforms the co-occurrence count model consecutively.

Taking Spanish-English-French translation as an example, the co-occurrence count model achieves BLEU% scores of 35.38, which is close to the direct translation model trained with 200K source-target bilingual data. Compared with the co-occurrence count model, the triangulation model only achieves BLEU% scores of 33.84, which is close to the direct translation model trained with 50K source-target bilingual data.

(2) Comparison of Different Interpolated Models

For the pivot model trained by triangulation method and co-occurrence count method, we interpolate them with the direct translation model trained with different scales of bilingual data. Figure 3 shows the translation results of the different interpolated models. For all the translation directions, our co-occurrence count method interpolated with the direct model is better than the triangulation model interpolated with the direct model.

The two interpolated model are all better than the direct translation model. With the increment of the source-target training corpus, the gap becomes smaller. This indicates that the pivot model and its affiliated interpolated model are suitable for language pairs with small bilingual data. Even if the scale of source-pivot and pivot-target corpora is close to the scale of source-target bilingual corpora, the pivot translation model can help the direct translation model to improve the translation performance. Take Spanish-English-French translation as an issue, when the scale of Spanish-French parallel data is 1.5M sentences pairs, which is close to the Spanish-English and English-French parallel data, the performance of *co+mix* model is still outperforms the *direct* translation model.

(3) Comparison of Interpolated Model and Mixed Model

When only a small source-target bilingual corpus is available, the mix model outperforms the interpolated model. With the increasing of source-target corpus, the mix model is close to the interpolated model or worse than the interpolated model. This indicates that the mix model has a better performance when the source-target corpus is small which is close to the realistic scenario.

5.3 Integrate the Co-occurrence Count Model and Triangulation Model

Experimental results in the previous section show that, our co-occurrence count models generally outperform the baseline system. In this section, we carry out experiments that integrates co-occurrence count model into the triangulation model.

For French-English-German translation, we apply a linear interpolation method to integrate the co-occurrence count model into triangulation model following the method described in Section 4.2. We set α as the interpolation coefficient of triangulation model and $1 - \alpha$ as the interpolation coefficient of co-occurrence count model respectively. The experiments take 9 values for interpolation coefficient, from 0.1 to 0.9. The results are shown in Figure 4.

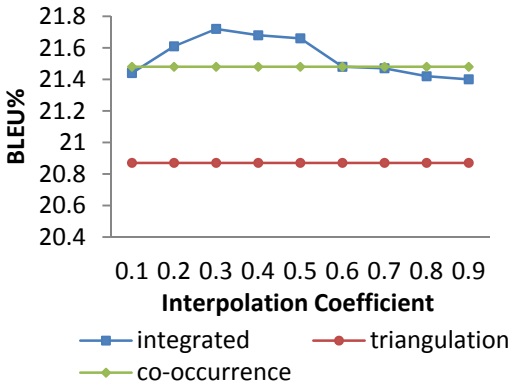


Figure 4: Results of integrating the co-occurrence count model and the triangulation model.

When using interpolation coefficient ranging from 0.2 to 0.7, the integrated models outperform the triangulation and the co-occurrence count model. However, for the other intervals, the inte-

Language Pairs	Sentence Pairs	Source Words	Target Words
zh-en-1	1M	18.1M	17.7M
zh-en-2	2M	36.2M	35.5M
zh-en-3	3M	54.2M	53.2M
zh-en-4	4M	72.3M	70.9M
zh-en-5	5M	90.4M	88.6M
en-jp	1M	9.2M	11.1M

Table 4: Training data of web corpus

System	BLEU%				
	zh-en-jp-1*	zh-en-jp-2	zh-en-jp-3	zh-en-jp-4	zh-en-jp-5
Baseline	29.07	29.39	29.44	29.67	29.80
Minimum	**31.13***	**31.28***	**31.43***	**31.62***	**32.02***
Maximum	28.88	29.01	29.12	29.37	29.59
Arithmetic mean	29.08	29.36	29.51	29.79	30.01
Geometric mean	**30.77***	**31.30***	**31.75***	**32.07***	**32.34***

Table 5: Comparison of different merging methods on the imbalanced web data. (zh-en-jp-1 means the translation system is trained with zh-en-1 as source-pivot corpus and en-jp as pivot-target corpus, and so on.)

grated models perform slightly lower than the co-occurrence count model, but still show better results than the triangulation model. The trend of the curve infers that the integrated model synthesizes the contributions of co-occurrence count model and triangulation model. Additionally, it also indicates that, the choice of the interpolation coefficient affects the translation performances.

6 Experiments on Web Data

The experimental on Europarl is artificial, as the training data for directly translating between source and target language actually exists in the original data sets. Thus, we conducted several experiments on a more realistic scenario: translating Chinese (zh) to Japanese (jp) via English (en) with web crawled data.

As mentioned in Section 3.1, the source-pivot and pivot-target parallel corpora can be imbalanced in quantities. If one parallel corpus was much larger than another, then *minimum* heuristic function would likely just take the counts from the smaller corpus.

In order to analyze this issue, we manually set up imbalanced corpora. For source-pivot parallel corpora, we randomly select 1M, 2M, 3M, 4M and 5M Chinese-English sentence pairs. On the other hand, we randomly select 1M English-Japanese sentence pairs as pivot-target parallel corpora. The training data of Chinese-English and English-Japanese language pairs are summarized in Table 4. For the Chinese-Japanese direct corpus, we randomly select 5K, 10K, 20K, 30K, 40K, 50K, 60K, 70K, 80K, 90K and 100K sentence pairs to simulate the lack of bilingual data. We built a 1K in-house test set with four references. For Japanese language model training, we used the monolingual part of English-Japanese corpus.

Table 5 shows the results of different co-occurrence count merging methods. First, the *minimum* method and the *geometric mean* method outperform the other two merging methods and the baseline system with different training corpus. When the scale of source-pivot and pivot-target corpus is roughly balanced (zh-en-jp-1), the *minimum* method achieves an absolute improvement of 2.06 percentages points on BLEU over the baseline, which is also better than the other merging methods. While, with the growth of source-pivot corpus, the gap between source-pivot corpus and pivot-target corpus becomes bigger. In this circumstance, the *geometric mean* method becomes better than the *minimum* method. Compared to the *minimum* method, the *geometric mean* method considers both the source-pivot and the pivot-target corpus, which may lead to a better result in the case of imbalanced training corpus.

Furthermore, with the imbalanced corpus zh-en-jp-5, we compared the translation performance of our co-occurrence count model (with *geometric mean* merging method), triangulation model, interpolated model, mixed model and the direct translation models. Figure 5 summarized the results.

The co-occurrence count model can achieve an absolute improvement of 2.54 percentages points on BLEU over the baseline. The triangulation method outperforms the direct translation when only 5K sentence pairs are available. Meanwhile, the number is 10K when using the co-occurrence count method. The co-occurrence count models interpolated with the direct model significantly outperform the other models.

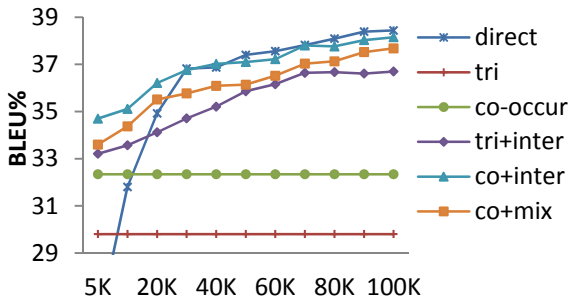


Figure 5: Results on Chinese-Japanese Web Data. X-axis represents the scale of the standard training data.

In this experiment, the training data contains parallel sentences on various domains. And the training corpora (Chinese-English and English-Japanese) are typically very different, since they are obtained on the web. It indicates that our co-occurrence count method is robust in the realistic scenario.

7 Conclusion

This paper proposed a novel approach for pivot-based SMT by pivoting the co-occurrence count of phrase pairs. Different from the triangulation method merging the source-pivot and pivot-target language after training the translation model, our method merges the source-pivot and pivot-target language after extracting the phrase pairs, thus the computing for phrase translation probabilities is under the uniform probability space. The experimental results on Europarl data and web data show significant improvements over the baseline systems. We also proposed a mixed model to combine the direct translation and pivot translation, and the experimental results show that the mixed model has a better performance when the source-target corpus is small which is close to the realistic scenario.

A key problem in the approach is how to learn the co-occurrence count. In this paper, we use the *minimum* function on balanced corpora and the *geometric mean* function on imbalanced corpora to estimate the co-occurrence count intuitively. In the future, we plan to explore more effective approaches.

Acknowledgments

We would like to thank Yiming Cui for insightful discussions, and three anonymous reviewers for many invaluable comments and suggestions to improve our paper. This work is supported by National Natural Science Foundation of China (61100093), and the State Key Development Program for Basic Research of China (973 Program, 2014CB340505).

Reference

Nicola Bertoldi, Madalina Barbaiani, Marcello Federico, and Roldano Cattoni. 2008. Phrase-Based statistical machine translation with Pivot Languages. In *Proceedings of the 5th International Workshop on Spoken Language Translation (IWSLT)*, pages 143-149.

Trevor Cohn and Mirella Lapata. 2007. Machine Translation by Triangulation: Make Effective Use of Multi-Parallel Corpora. In *Proceedings of 45th Annual Meeting of the Association for Computational Linguistics*, pages 828-735.

Marta R. Costa-jussà, Carlos Henríquez, and Rafael E. Banchs. 2011. Enhancing Scarce-Resource Language Translation through Pivot Combinations. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 1361-1365.

Yiming Cui, Conghui Zhu, Xiaoning Zhu, Tiejun Zhao and Dequan Zheng. 2013. Phrase Table Combination Deficiency Analyses in Pivot-based SMT. In *Proceedings of 18th International Conference on Application of Natural Language to Information Systems*, pages 355-358.

Adria de Gispert and Jose B. Marino. 2006. Catalan-English statistical machine translation without parallel corpus: bridging through Spanish. In *Proceedings of 5th International Conference on Language Resources and Evaluation (LREC)*, pages 65-68.

Kevin Duh, Katsuhito Sudoh, Xianchao Wu, Hajime Tsukada and Masaaki Nagata. 2011. Generalized Minimum Bayes Risk System Combination. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 1356-1360.

Ahmed El Kholy, Nizar Habash, Gregor Leusch, Evgeny Matusov and Hassan Sawaf. 2013. Language Independent Connectivity Strength Features for Phrase Pivot Statistical Machine Translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 412-418.

Ahmed El Kholy, Nizar Habash, Gregor Leusch, Evgeny Matusov and Hassan Sawaf. 2013. Selective Combination of Pivot and Direct Statistical Machine Translation Models. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pages 1174-1180.

Jesús González-Rubio, Alfons Juan and Francisco Casacuberta. 2011. Minimum Bayes-risk System Combination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 1268-1277.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *HLT-NAACL: Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127-133.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388-395.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of MT Summit X*, pages 79-86.

Philipp Koehn, Hieu Hoang, Alexanda Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, demonstration session*, pages 177-180.

Philipp Koehn, Alexandra Birch, and Ralf Steinberger. 2009. 462 Machine Translation Systems for Europe. In *Proceedings of the MT Summit XII*.

Gregor Leusch, Aurélien Max, Josep Maria Crego and Hermann Ney. 2010. Multi-Pivot Translation by System Combination. In *Proceedings of the 7th International Workshop on Spoken Language Translation*, pages 299-306.

Franz Josef Och and Hermann Ney. 2000. A comparison of alignment models for statistical machine translation. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 1086-1090.

Michael Paul, Andrew Finch, Paul R. Dixon and Eiichiro Sumita. 2011. Dialect Translation: Integrating Bayesian Co-segmentation Models with Pivot-based SMT. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1-9.

Michael Paul and Eiichiro Sumita. 2011. Translation Quality Indicators for Pivot-based Statistical MT. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 811-818.

Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computation Linguistics*, pages 311-319.

Rie Tanaka, Yohei Murakami and Toru Ishida. 2009. Context-Based Approach for Pivot Translation Services. In the Twenty-first International Conference on Artificial Intelligence, pages 1555-1561.

Jörg Tiedemann. 2012. Character-Based Pivot Translation for Under-Resourced Languages and Domains. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 141-151.

Masatoshi Tsuchiya, Ayu Purwarianti, Toshiyuki Wakita and Seiichi Nakagawa. 2007. Expanding Indonesian-Japanese Small Translation Dictionary Using a Pivot Language. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 197-200.

Takashi Tsunakawa, Naoaki Okazaki and Jun'ichi Tsujii. 2010. Building a Bilingual Lexicon Using Phrase-based Statistical Machine Translation via a Pivot Language. In

Proceedings of the 22th International Conference on Computational Linguistics (Coling), pages 127-130.

Masao Utiyama and Hitoshi Isahara. 2007. A Comparison of Pivot Methods for Phrase-Based Statistical Machine Translation. In *Proceedings of Human Language Technology: the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 484-491.

Masao Utiyama, Andrew Finch, Hideo Okuma, Michael Paul, Hailong Cao, Hirofumi Yamamoto, Keiji Yasuda,and Eiichiro Sumita. 2008. The NICT/ATR speech Translation System for IWSLT 2008. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 77-84.

Haifeng Wang, Hua Wu, Xiaoguang Hu, Zhanyi Liu, Jianfeng Li, Dengjun Ren, and Zhengyu Niu. 2008. The TCH Machine Translation System for IWSLT 2008. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 124-131.

Hua Wu and Haifeng Wang. 2007. Pivot Language Approach for Phrase-Based Statistical Machine Translation. In *Proceedings of 45th Annual Meeting of the Association for Computational Linguistics*, pages 856-863.

Hua Wu and Haifeng Wang. 2009. Revisiting Pivot Language Approach for Machine Translation. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th IJCNLP of the AFNLP*, pages 154-162.

Samira Tofighi Zahabi, Somayeh Bakhshaei and Shahram Khadivi. Using Context Vectors in Improving a Machine Translation System with Bridge Language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 318-322.

Word Translation Prediction for Morphologically Rich Languages with Bilingual Neural Networks

Ke Tran Arianna Bisazza Christof Monz
Informatics Institute, University of Amsterdam
Science Park 904, 1098 XH Amsterdam, The Netherlands
{m.k.tran,a.bisazza,c.monz}@uva.nl

Abstract

Translating into morphologically rich languages is a particularly difficult problem in machine translation due to the high degree of inflectional ambiguity in the target language, often only poorly captured by existing word translation models. We present a general approach that exploits source-side contexts of foreign words to improve translation prediction accuracy. Our approach is based on a probabilistic neural network which does not require linguistic annotation nor manual feature engineering. We report significant improvements in word translation prediction accuracy for three morphologically rich target languages. In addition, preliminary results for integrating our approach into a large-scale English-Russian statistical machine translation system show small but statistically significant improvements in translation quality.

1 Introduction

The ability to make context-sensitive translation decisions is one of the major strengths of phrase-based SMT (PSMT). However, the way PSMT exploits source-language context has several limitations as pointed out, for instance, by Quirk and Menezes (2006) and Durrani et al. (2013). First, the amount of context used to translate a given input word depends on the phrase segmentation, with hypotheses resulting from different segmentations competing with one another. Another issue is that, given a phrase segmentation, each source phrase is translated independently from the others, which can be problematic especially for short phrases. As a result, the predictive translation of a source phrase does not access useful linguistic clues in the source sentence that are outside of the scope of the phrase.

Lexical weighting tackles the problem of unreliable phrase probabilities, typically associated with long phrases, but does not alleviate the problem of context segmentation. An important share of the translation selection task is then left to the language model (LM), which is certainly very effective but can only leverage *target* language context. Moreover, decisions that are taken at early decoding stages—such as the common practice of retaining only top n translation options for each source span—depend only on the translation models and on the target context available in the phrase.

Source context based translation models (Gimpel and Smith, 2008; Mauser et al., 2009; Jeong et al., 2010; Haque et al., 2011) naturally address these limitations. These models can exploit a boundless context of the input text, but they assume that target words can be predicted independently from each other, which makes them easy to integrate into state-of-the-art PSMT systems. Even though the independence assumption is made on the target side, these models have shown the benefits of utilizing source context, especially in translating into morphologically rich languages. One drawback of previous research on this topic, though, is that it relied on rich sets of manually designed features, which in turn required the availability of linguistic annotation tools like POS taggers and syntactic parsers.

In this paper, we specifically focus on improving the prediction accuracy for word translations. Achieving high levels of word translation accuracy is particularly challenging for language

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1676–1688,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

pairs where the source language is morphologically poor, such as English, and the target language is morphologically rich, such as Russian, i.e., language pairs with a high degree of surface realization ambiguity (Minkov et al., 2007). To address this problem we propose a general approach based on bilingual neural networks (BNN) exploiting source-side contextual information.

This paper makes a number of contributions: Unlike previous approaches our models do not require any form of linguistic annotation (Minkov et al., 2007; Kholy and Habash, 2012; Chahuneau et al., 2013), nor do they require any feature engineering (Gimpel and Smith, 2008). Moreover, besides directly predicting fully inflected forms as Jeong et al. (2010), our approach can also model stem and suffix prediction explicitly. Prediction accuracy is evaluated with respect to three morphologically rich target languages (Bulgarian, Czech, and Russian) showing that our approach consistently yields substantial improvements over a competitive baseline. We also show that these improvements in prediction accuracy can be beneficial in an end-to-end machine translation scenario by integrating into a large-scale English-Russian PSMT system. Finally, a detailed analysis shows that our approach induces a positive bias on phrase translation probabilities leading to a better ranking of the translation options employed by the decoder.

2 Lexical coverage of SMT models

The first question we ask is whether translation can be improved by a more accurate selection of the translation options already existing in the SMT models, as opposed to generating new options. To answer this question we measure the lexical coverage of a baseline PSMT system trained on English-Russian.[1] We choose this language pair because of the morphological richness on the target side: Russian is characterized by a highly inflectional morphology with a particularly complex nominal declension (six core cases, three genders and two number categories). As suggested by Green and DeNero (2012), we compute the recall of reference tokens in the set of target tokens that the decoder could produce in a translation of the source, that is the target tokens of all phrase pairs that matched the input sentence and that were actually used for decoding.[2] We call this the decoder's *lexical search space*. Then, we compare the reference/space recall against the reference/MT-output recall: that is, the percentage of reference tokens that also appeared in the 1-best translation output by the SMT system. Results for the WMT12 benchmark are presented in Table 1. From the first two rows, we see that only a rather small part of the correct target tokens available to the decoder are actually produced in the 1-best MT output (50% against 86%). Although our word-level analysis does not directly estimate phrase-level coverage, these numbers suggest that a large potential for translation improvement lies in better lexical selection during decoding.

Token recall:	
reference/MT-search-space	86.0%
reference/MT-output	50.0%
stem-only reference/MT-output	12.3%
of which reachable	11.2%

Table 1: Lexical coverage analysis of the baseline SMT system (English-Russian wmt12).

To quantify the importance of morphology, we count how many reference tokens matched the MT output only at the stem level[3] and for how many of those the correct surface form existed in the search space (reachable matches). These two numbers represent the upper bound of the improvement achievable by a model only predicting suffixes given the target stems. As shown in Table 1, such a model could potentially increase the reference/MT-output recall by 12.3% with generation of new inflected forms, and by 11.2% without. Thus, also when it comes to morphology, generation seems to be of secondary importance compared to better selection in our experimental setup.

3 Predicting word translations in context

It is standard practice in PSMT to use word-to-word translation probabilities as an additional phrase score. More specifically, state-of-the-art PSMT systems employ the maximum-likelihood estimate of the context-independent probability of a target word given its aligned source word $P(t_j|s_i)$ for each word alignment link a_{ij}.

[1] Training data and SMT setup are described in Section 6.

[2] This corresponds to the top 30 phrases sorted by weighted phrase, lexical and LM probabilities, for each source span. Koehn (2004) and our own experience suggest that using more phrases has little or no impact on MT quality.

[3] Word segmentation for this analysis is performed by the Russian Snowball stemmer, see also Section 5.3.

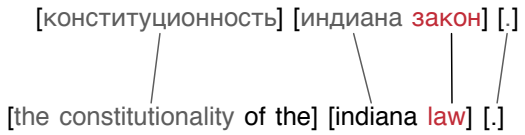


Figure 1: Fragment of English sentence and its incorrect Russian translation produced by the baseline SMT system. Square brackets indicate phrase boundaries.

The main goal of our work is to improve the estimation of such probabilities by exploiting the context of s_i, which in turn we expect will result in better phrase translation selection. Figure 1 illustrates this idea: the translation of "law" in this example has a wrong case—nominative instead of genitive. Due to the rare word "Indiana/индиана", the target LM must backoff to the bigram history and does not penalize this choice sufficiently. However, a model that has access to the word "of" in the near source context could bias the translation of "law" to the correct case.

We then model $P(t_j|\mathbf{c}_{s_i})$ with source context $\mathbf{c}_{s_i}$ defined as a fixed-length word sequence centered around s_i:

$$\mathbf{c}_{s_i} = s_{i-k}, ..., s_i, ..., s_{i+k}$$

Our definition of context is similar to the $n-1$ word history used in n-gram LMs. Similarly to previous work in source context-sensitive translation modeling (Jeong et al., 2010; Chahuneau et al., 2013), target words are predicted independently from each other, which allows for an efficient decoding integration. We are particularly interested in translating into morphologically rich languages where source context can provide useful information for the prediction of target translation, for example, the gender of the subject in a source sentence constrains the morphology of the translation of the source verb. Therefore, we integrate the notions of stem and suffix directly into the model. We assume the availability of a word segmentation function g that takes a target word t as input and returns its stem and suffix: $g(t) = (\sigma, \mu)$. Then, the conditional probability $p(t_j|\mathbf{c}_{s_i})$ can be decomposed into stem probability and suffix probability:

$$p(t_j|\mathbf{c}_{s_i}) = p(\sigma_j|\mathbf{c}_{s_i})p(\mu_j|\mathbf{c}_{s_i}, \sigma_j) \quad (1)$$

These two probabilities can be estimated separately, which yields the two subtasks:

1. predict target stem σ given source context $\mathbf{c}_s$;
2. predict target suffix μ given source context $\mathbf{c}_s$ and target stem σ.

Based on the results of our analysis, we focus on the selection of existing translation candidates. We then restrict our prediction on a set of possible target candidates depending on the task instead of considering all target words in the vocabulary. More specifically, for each source word s_i, our candidate generation function returns the set of target words $T_s = \{t_1, \ldots, t_m\}$ that were aligned to s_i in the parallel training corpus, which in turn corresponds to the set of target words that the SMT system can produce for a given source. In practice, we use a pruned version of T_s to speed up training and reduce noise (see details in Section 5).

As for the morphological models, given T_s and g, we can obtain $L_s = \{\sigma_1, \ldots, \sigma_k\}$, the set of possible target stem translations of s, and $M_\sigma = \{\mu_1, \ldots, \mu_l\}$, the set of possible suffixes for a target stem σ. The use of L_s, and M_σ is similar to stemming and inflection operations in (Toutanova et al., 2008) while the set T_s is similar to the GEN function in (Jeong et al., 2010).[4]

Our approach differs crucially from previous work (Minkov et al., 2007; Chahuneau et al., 2013) in that it does not require linguistic features such as part-of-speech and syntactic tree on the source side. The proposed models automatically learn features that are relevant for each of the modeled tasks, directly from word-aligned data. To make the approach completely language independent, the word segmentation function g can be trained with an unsupervised segmentation tool. The effects of using different word segmentation techniques are discussed in Section 5.

4 Bilingual neural networks for translation prediction

Probabilistic neural network (NN), or continuous space, language models have received increasing attention over the last few years and have been applied to several natural language processing tasks (Bengio et al., 2003; Collobert and Weston, 2008; Socher et al., 2011; Socher et al., 2012). Within statistical machine translation, they

[4]Note that our suffix generation function M_σ is restricted to the forms observed in the target monolingual data, but not to those aligned to a source word s, which opens the possibility of generating inflected forms that are missing from the translation models. We leave this possibility to future work.

have been used for monolingual target language modeling (Schwenk et al., 2006; Le et al., 2011; Duh et al., 2013; Vaswani et al., 2013), n-gram translation modeling (Son et al., 2012), phrase translation modeling (Schwenk, 2012; Zou et al., 2013; Gao et al., 2014) and minimal translation modeling (Hu et al., 2014). The recurrent neural network LMs of Auli et al. (2013) are primarily trained to predict target word sequences. However, they also experiment with an additional input layer representing source side context.

Our models differ from most previous work in neural language modeling in that we predict a target translation given a source context while previous models predict the next word given a target word history. Unlike previous work in phrase translation modeling with NNs, our models have the advantage of accessing source context that can fall outside the phrase boundaries.

We now describe our models in a general setting, predicting target translations given a source context, where target translations can be either words, stems or suffixes.[5]

4.1 Neural network architecture

Following a common approach in deep learning for NLP (Bengio et al., 2003; Collobert and Weston, 2008), we represent each source word s_i by a column vector $\mathbf{r}_{s_i} \in \mathbb{R}^d$. Given a source context $\mathbf{c}_{s_i} = s_{i-k}, ..., s_i, ..., s_{i+k}$ of k words on the left and k words on the right of s_i, the context representation $\mathbf{r}_{\mathbf{c}_{s_i}} \in \mathbb{R}^{(2k+1)\times d}$ is obtained by concatenating the vector representations of all words in $\mathbf{c}_{s_i}$:

$$\mathbf{r}_{\mathbf{c}_{s_i}} = \mathbf{r}_{s_{i-k}} \odot ... \odot \mathbf{r}_{s_{i+k}}$$

Our main BNN architecture for word or stem prediction (Figure 2a) is a feed-forward neural network (FFNN) with one hidden layer, a matrix $\mathbf{W}_1 \in \mathbb{R}^{n\times(2k+1)d}$ connecting the input layer to the hidden layer, a matrix $\mathbf{W}_2 \in \mathbb{R}^{|V_t|\times n}$ connecting the hidden layer to the output layer, and a bias vector $\mathbf{b}_2 \in \mathbb{R}^{|V_t|}$ where $|V_t|$ is the size of target translations vocabulary. The target translation distribution $P_{\text{BNN}}(t|\mathbf{c}_{s_i})$ for a given source context $\mathbf{c}_{s_i}$ is computed by a forward pass:

$$\text{softmax}\left(\mathbf{W}_2\, \phi(\mathbf{W}_1 \mathbf{r}_{\mathbf{c}_{s_i}}) + \mathbf{b}_2\right) \qquad (2)$$

where ϕ is a nonlinearity (tanh, sigmoid or rectified linear units). The parameters of the neural network are $\boldsymbol{\theta} = \{\mathbf{r}_{s_i}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_2\}$.

The suffix prediction BNN is obtained by adding the target stem representation $\mathbf{r}_\sigma$ to the input layer (see Figure 2b).

4.2 Model variants

We encounter two major issues with FFNNs: (i) They do not provide a natural mechanism to compute word surface conditional probability $p(t|\mathbf{c}_s)$ given individual stem probability $p(\sigma|\mathbf{c}_s)$ and suffix probability $p(\mu|\mathbf{c}_s, \sigma)$, and (ii) FFNNs do not provide the flexibility to capture long dependencies among words if they lie outside the source context window. Hence, we consider two BNN variants: a log-bilinear model (LBL) and a convolutional neural network model (ConvNet). LBL could potentially address (i) by factorizing target representations into target stem and suffix representations whereas ConvNets offer the advantage of modeling variable input length (ii) (Kalchbrenner et al., 2014).

Log-bilinear model. The FFNN models stem and suffix probabilities separately. A log-bilinear model instead could directly model word prediction through a factored representation of target words, similarly to Botha and Blunsom (2014). Thus, no probability mass would be wasted over stem-suffix combinations that are not in the candidate generation function. The LBL model specifies the conditional distribution for the word translation $t_j \in T_{s_i}$ given a source context $\mathbf{c}_{s_i}$:

$$P_{\boldsymbol{\theta}}(t_j|\mathbf{c}_{s_i}) = \frac{\exp(s_{\boldsymbol{\theta}}(t_j, \mathbf{c}_{s_i}))}{\sum_{t'_j \in T_{s_i}} \exp(s_{\boldsymbol{\theta}}(t'_j, \mathbf{c}_{s_i}))} \qquad (3)$$

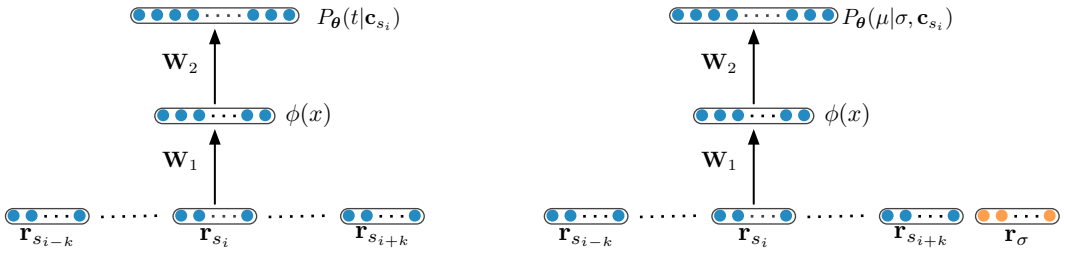
We use an additional set of word representations $\mathbf{q}_{t_j} \in \mathbb{R}^n$ for target translations t_j. The LBL model computes a predictive representation $\hat{\mathbf{q}}$ of a source context $\mathbf{c}_{s_i}$ by taking a linear combination of the source word representations $\mathbf{r}_{s_{i+m}}$ with the position-dependent weight matrices $\mathbf{C}_m \in \mathbb{R}^{n\times d}$:

$$\hat{\mathbf{q}} = \sum_{m=-k}^{k} \mathbf{C}_m \mathbf{r}_{s_{i+m}} \qquad (4)$$

The score function $s_{\boldsymbol{\theta}}(t_j, \mathbf{c}_{s_i})$ measures the similarity between the predictive representation $\hat{\mathbf{q}}$ and the target representation $\mathbf{q}_{t_j}$:

$$s_{\boldsymbol{\theta}}(t_j, \mathbf{c}_{s_i}) = \hat{\mathbf{q}}^\top \mathbf{q}_{t_j} + \mathbf{b}_h^\top \mathbf{q}_{t_j} + b_{t_j} \qquad (5)$$

[5] The source code of our models is available at `https://bitbucket.org/ketran`



(a) BNN for word prediction.

(b) BNN for suffix prediction.

Figure 2: Feed-forward BNN architectures for predicting target translations: (a) word model (similar to stem model), and (b) suffix model with an additional vector representation $\mathbf{r}_\sigma$ for target stems σ.

Here b_{t_j} is the bias term associated with target word t_j. $\mathbf{b}_h \in \mathbb{R}^n$ are the representation biases. $s_{\boldsymbol{\theta}}(t_j, \mathbf{c}_{s_i})$ can be seen as the negative energy function of the target translation t_j and its context $\mathbf{c}_{s_i}$. The parameters of the model thus are $\boldsymbol{\theta} = \{\mathbf{r}_{s_i}, \mathbf{C}_m, \mathbf{q}_{t_j}, \mathbf{b}_h, b_{t_j}\}$. Our log-bilinear model is a modification of the log-bilinear model proposed for n-gram language modeling in (Mnih and Hinton, 2007).

Convolutional neural network model. This model (Figure 3) computes the predictive representation $\hat{\mathbf{q}}$ by applying a sequence of $2k$ convolutional layers $\{\mathbf{L}_1, \ldots, \mathbf{L}_{2k}\}$. The source context $\mathbf{c}_{s_i}$ is represented as a matrix $\mathbf{m}_{c_{s_i}} \in \mathbb{R}^{d\times(2k+1)}$:

$$\mathbf{m}_{c_{s_i}} = \left[\mathbf{r}_{s_{i-k}}; \ldots; \mathbf{r}_{s_{i+k}}\right] \tag{6}$$

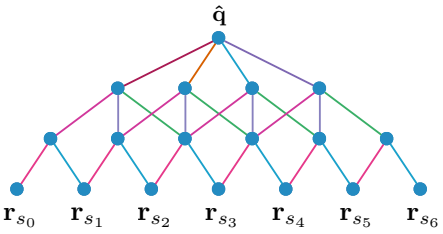


Figure 3: Convolutional neural network model. Edges with the same color indicate the same kernel weight matrix.

Each convolutional layer $\mathbf{L}_i$ consists of a one-dimensional filter $\mathbf{m}_i \in \mathbb{R}^{d\times 2}$. Each row of $\mathbf{m}_i$ is convolved with the corresponding row in the previous layer resulting in a weight matrix whose number of columns decreases by one. Thus after $2k$ convolutional layers, the network transforms the source context matrix $\mathbf{m}_{c_{s_i}}$ to a feature vector $\hat{\mathbf{q}} \in \mathbb{R}^d$. A fully connected layer with weight matrix $\mathbf{W}$ followed by a softmax layer are placed after the last convolutional layer $\mathbf{L}_{2k}$ to perform classification. The parameters of the convolutional neural network model are $\boldsymbol{\theta} = \{\mathbf{r}_{s_i}, \mathbf{m}_j, \mathbf{W}\}$. Here, we focus on a fixed length input, however convolutional neural networks may be used to model variable length input (Kalchbrenner et al., 2014; Kalchbrenner and Blunsom, 2013).

4.3 Training

In training, for each example $(t, \mathbf{c}_s)$, we maximize the conditional probability $P_{\boldsymbol{\theta}}(t|\mathbf{c}_s)$ of a correct target label t. The contribution of the training example $(t, \mathbf{c}_s)$ to the gradient of the log conditional probability is given by:

$$\frac{\partial}{\partial\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(t|\mathbf{c}_s) = \frac{\partial}{\partial\boldsymbol{\theta}} s_{\boldsymbol{\theta}}(t|\mathbf{c}_s) - \sum_{t'\in T_s} P_{\boldsymbol{\theta}}(t'|\mathbf{c}_s) \frac{\partial}{\partial\boldsymbol{\theta}} s_{\boldsymbol{\theta}}(t', \mathbf{c}_s)$$

Note that in the gradient, we do not sum over all target translations T but a set of possible candidates T_s of a source word s. In practice $|T_s| \leq 200$ with our pruning settings (see Section 5.1), thus training time for one example does not depend on the vocabulary size. Our training criterion can be seen as a form of contrastive estimation (Smith and Eisner, 2005), however we explicitly move the probability mass from *competing* candidates to the correct translation candidate, thus obtaining more reliable estimates of the conditional probabilities.

The BNN parameters are initialized randomly according to a zero-mean Gaussian. We regularize the models with L_2. As an alternative to the L_2 regularizer, we also experiment with dropout (Hinton et al., 2012), where the neurons are randomly zeroed out with dropout rate p. This technique is known to be useful in computer vision tasks but has been rarely used in NLP tasks. In FFNN, we use dropout after the hidden layer, while in ConvNet, dropout applies after the last convolutional layer. The dropout rate p is set to 0.3 in our exper-

iments. We use rectified nonlinearities[6] in FFNN and after each convolutional layer in ConvNet. We train our BNN models with the standard stochastic gradient descent.

5 Evaluating word translation prediction

In this section, we assess the ability of our BNN models to predict the correct translation of a word in context. In addition to English-Russian, we also consider translation prediction for Czech and Bulgarian. As members of the Slavic language family, Czech and Bulgarian are also characterized by highly inflectional morphology. Czech, like Russian, displays a very rich nominal inflection with as many as 14 declension paradigms. Bulgarian, unlike Russian, is not affected by case distinctions but is characterized by a definiteness suffix.

5.1 Experimental setup

The following parallel corpora are used to train the BNN models:

- English-Russian: WMT13 data (News Commentary and Yandex corpora);
- English-Czech: CzEng 1.0 corpus (Bojar et al., 2012) (Web Pages and News sections);
- English-Bulgarian: a mix of crawled news data, TED talks and Europarl proceedings.

Detailed corpus statistics are given in Table 2. For each language pair, accuracies are measured on a held-out set of 10K parallel sentences.

To prepare the candidate generation function, each dataset is first word-aligned with GIZA++, then a bilingual lexicon with maximum-likelihood probabilities (P_{mle}) is built from the symmetrized alignment. After some frequency and significance pruning,[7] the top 200 translations sorted by $P_{\text{mle}}(t|s) \cdot P_{\text{mle}}(s|t)$ are kept as candidate word translations for each source word in the vocabulary. Word alignments are also used to train the BNN models: each alignment link constitutes a training sample, with no special treatment of unaligned words and 1-to-many alignments.

The context window size k is set to 3 (corresponding to 7-gram) and the dimensionality of source word representations to 100 in all experiments. The number of hidden units in our feedforward neural networks and the target translation embedding size in LBL models are set to 200. All models are trained for 10 iterations with learning rate set to 0.001.

	En-Ru	**En-Cs**	**En-Bg**
Sentences	1M	1M	0.8M
Src. tokens	26.5M	19.2M	19.3M
Trg. tokens	24.7M	16.7M	18.9M
Src. T/T	.0109	.0105	.0051
Trg. T/T	.0247	.0163	.0104

Table 2: BNN training corpora statistics: number of sentences, tokens, and type/token ratio (T/T).

5.2 Word, stem and suffix prediction accuracy

We measure accuracy at top-n, i. e. the number of times the correct translation was in the top n candidates sorted by a model. For each subtask—word, stem and suffix prediction—the BNN model is compared to the context-independent maximum-likelihood baseline $P_{\text{mle}}(t|s)$ on which the PSMT lexical weighting score is based. Note that this is a more realistic baseline than the uniform models sometimes reported in the literature. The oracle corresponds to the percentage of aligned source-target word pairs in the held-out set that are covered by the candidate generation function. Out of the missing links, about 4% is due to lexicon pruning. Results for all three language pairs are presented in Table 3. In this series of experiments, the morphological BNNs utilize unsupervised segmentation models trained on each target language following Lee et al. (2011).[8]

As shown in Table 3, the BNN models outperform the baseline by a large margin in all tasks and languages. In particular, word prediction accuracy at top-1 increases by +6.4%, +24.6% and +9.0% absolute in English-Russian, English-Czech and English-Bulgarian respectively, without the use of any features based on linguistic annotation. While the baseline and oracle differences among languages can be explained by different levels of overlap between training and held-out set, we cannot easily explain why the Czech BNN performance is so much higher. When comparing the

[6] We find that using rectified linear units gives better results than sigmoid and tanh.

[7] Each lexicon is pruned with minimum word frequency 5, minimum source-target word pair frequency 2, minimum log odds ratio 10.

[8] We use the C++ implementation available at `http://groups.csail.mit.edu/rbg/code/morphsyn`

Model	En-Ru	En-Cs	En-Bg
Word prediction (%):			
Baseline	33.0 / 50.1	42.0 / 59.9	47.9 / 66.0
Word BNN	39.4 / 56.6	66.6 / 81.4	56.9 / 74.0
	+6.4 / +6.5	+24.6/+21.5	+9.0 / +8.0
Oracle	79.5	90.2	86.9
Stem prediction (%):			
Baseline	40.7 / 58.2	46.1 / 64.3	51.9 / 70.1
Stem BNN	45.1 / 62.5	66.1 / 81.6	56.7 / 74.4
	+4.4 / +4.3	+20.0/+17.3	+4.8 / +4.3
Suffix prediction (%):			
Baseline	71.2 / 85.6	78.8 / 93.2	81.5 / 92.4
Suffix BNN	77.0 / 89.7	91.9 / 97.4	87.7 / 94.9
	+5.8 / +4.1	+13.1 /+4.2	+6.2 / +2.5

Table 3: BNN prediction accuracy (top-1/top-3) compared to a context-independent maximum-likelihood baseline.

three prediction subtasks, we find that word prediction is the hardest task as expected. Stem prediction accuracies are considerably higher than word prediction accuracies in Russian, but almost equal in the other two languages. Finally, baseline accuracies for suffix prediction are by far the highest, ranging between 71.2% and 81.5%, which is primarily explained by a smaller number of candidates to choose from. Also on this task, the BNN model achieves considerable gains of +5.8%, +13.1% and +6.2% at top-1, without the need of manual feature engineering.

From these figures, it is hard to predict whether word BNNs or morphological BNNs will have a better effect on SMT performance. On one hand, the word-level BNN achieves the highest gain over the MLE baseline. On the other, the stem- and suffix-level BNNs provide two separate scoring functions, whose weights can be directly tuned for translation quality. A preliminary answer to this question is given by the SMT experiments presented in Section 6.

5.3 Effect of word segmentation

This section analyzes the effect of using different segmentation techniques. We consider two supervised tagging methods that produce lemma and inflection tag for each token in a context-sensitive manner: TreeTagger (Sharoff et al., 2008) for Russian and the Morce tagger (Spoustová et al., 2007) for Czech.[9] Finally, we employ the Russian Snowball rule-based stemmer as a light-weight context-insensitive segmentation technique.[10]

[9] Annotation included in the CzEng 1.0 corpus release.

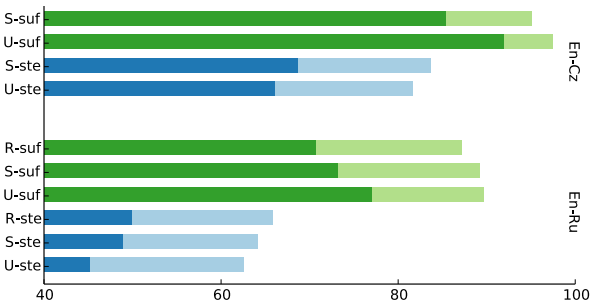


Figure 4: Effect of different word segmentation techniques (U: unsupervised, S: supervised, R: rule-based stemmer) on stem and suffix prediction accuracy. The dark part of each bar stands for top-1, the light one for top-3 accuracy.

As shown in Figure 4, accuracies for both stem and suffix prediction vary noticeably with the segmentation used. However, higher stem accuracies corresponds to lower suffix accuracies and vice versa, which can be mainly due to a general preference of a tool to segment more or less than another. In summary, the unsupervised segmentation methods and the light-weight stemmer appear to perform comparably to the supervised methods.

5.4 Effect of training data size

We examine the predictive power of our models with respect to the size of training data. Table 4 shows the accuracies of stem and suffix models trained on 200K and 1M English-Russian sentence pairs with unsupervised word segmentation. Surprisingly, we observe only a minor loss when we decrease the training data size, which suggests that our models are robust even on a small data set.

# Train sent.	Stem Acc.	Suffix Acc.
1M	45.1 / 62.5	77.0 / 89.7
200K	44.6 / 61.8	75.7 / 88.6

Table 4: Accuracy at top-1/top-3 (%) of stem and suffix BNNs with different training data sizes.

5.5 Fine-grained evaluation

We evaluate the suffix BNN model at the part-of-speech (POS) level. Table 5 provides suffix prediction accuracy per POS for En-Ru. For this analysis, Russian data is segmented by TreeTag-

[10] http://snowball.tartarus.org/algorithms/russian/stemmer.html

ger. Additionally, we report the average number of suffixes per stem given the part-of-speech.

Our results are consistent with the findings of Chahuneau et al. (2013):[11] the prediction of adjectives is more difficult than that of other POS while Russian verb prediction is relatively easier in spite of the higher number of suffixes per stem. These differences reflect the importance of source versus target context features in the prediction of the target inflection: For instance, adjectives agree in gender with the nouns they modify, but this may be only inferred from the target context.

POS	A	V	N	M	P
Acc. (%)	49.6	61.9	62.8	84.5	64.4
$\lvert M_\sigma \rvert$	18.2	18.4	9.2	7.1	13.3

Table 5: Suffix prediction accuracy at top-1 (%), breakdown by category (A: adjectives, V: verbs, N: nouns, M: numerals and P: pronouns). $|M_\sigma|$ denotes the average number of suffixes per stem.

5.6 Neural Network variants

Table 6 shows the stem and suffix accuracies of BNN variants on English-Czech. Although none of the variants outperform our main FFNN architecture, we observe similar performances by the LBL on stem prediction, and by the ConvNet on suffix prediction. This suggests that future work could exploit their additional flexibilities (see Section 4.2) to improve the BNN predictive power. As for the low suffix accuracy by the LBL, it can be explained by the absence of nonlinearity transformation. Nonlinearity is important for the suffix model where the prediction of target suffix μ_j often does not depend linearly on s_i and σ_j. The predictive representation of target stem in the LBL stem model, however, mainly depends on the source representation $\mathbf{r}_{s_i}$ through a position dependent weight matrix $\mathbf{C}_0$. Thus, we observe a smaller accuracy drop in the stem model than in the suffix model. Conversely, the ConvNet performs poorly on stem prediction because it captures the meaning of the whole source context instead of emphasizing the importance of the source word s_i as the main predictor of the target translation t_j.

[11] Chahuneau et al. (2013) report an average accuracy of 63.1% for the prediction of A, V, N, M suffixes. When we train our model on the same dataset (news-commentary) we obtain a comparable result (64.7% vs 63.1%).

Unexpectedly, no improvement is obtained by the use of dropout regularizer (see Section 4.3).

Model	Stem Acc	Suffix Acc
FFNN	66.1 / 81.6	91.9 / 97.4
FFNN+do	64.6 / 81.1	91.5 / 97.5
LBL	63.6 / 79.6	86.4 / 96.4
ConvNet+do	58.6 / 75.6	90.3 / 96.9

Table 6: Accuracies at top-1/top-3 (%) of stem and suffix models. +do indicates dropout instead of L_2 regularizer. FFNN is our main architecture.

6 SMT experiments

While the main objective of this paper is to improve prediction accuracy of word translations, see Section 5, we are also interested in knowing to which extent these improvements carry over within an end-to-end machine translation task. To this end, we integrate our translation prediction models described in Section 4 into our existing English-Russian SMT system.

For each phrase pair matching the input, the phrase BNN score $P_{\text{BNN-p}}$ is computed as follows:

$$P_{\text{BNN-p}}(\tilde{s}, \tilde{t}, a) = \prod_{i=1}^{|\tilde{s}|} \begin{cases} \frac{1}{|\{a_i\}|} \sum_{j \in \{a_i\}} P_{\text{BNN}}(t_j | \mathbf{c}_{s_i}) & \text{if } |\{a_i\}| > 0 \\ P_{\text{mle}}(\text{NULL} | s_i) & \text{otherwise} \end{cases}$$

where a is the word-level alignment of the phrase pair $(\tilde{s}, \tilde{t})$ and $\{a_i\}$ is the set of target positions aligned to s_i. If a source-target link cannot be scored by the BNN model, we give it a P_{BNN} probability of 1 and increment a separate count feature ε. Note that the same phrase pair can get different BNN scores if used in different source side contexts.

Our baseline is an in-house phrase-based (Koehn et al., 2003) statistical machine translation system very similar to Moses (Koehn et al., 2007). All system runs use hierarchical lexicalized reordering (Galley and Manning, 2008; Cherry et al., 2012), distinguishing between monotone, swap, and discontinuous reordering, all with respect to left-to-right and right-to-left decoding. Other features include linear distortion, bidirectional lexical weighting (Koehn et al., 2003), word and phrase penalties, and finally a word-level 5-gram target LM trained on all available monolingual data with modified Kneser-Ney smoothing (Chen and Goodman, 1999). The distortion

Corpus	Lang.	#Sent.	#Tok.
paral.train	EN	1.9M	48.9M
	RU		45.9M
Wiki dict.	EN/RU	508K	–
mono.train	RU	21.0M	390M
WMT2012	EN	3K	64K
WMT2013		3K	56K

Table 7: SMT training and test data statistics. All numbers refer to tokenized, lowercased data.

limit is set to 6 and for each source phrase the top 30 translation candidates are considered. When translating into a morphologically rich language, data sparsity issues in the target language become particularly apparent. To compensate for this we also experiment with a 5-gram suffix-based LM in addition to the surface-based LM (Müller et al., 2012; Bisazza and Monz, 2014).

The BNN models are integrated as additional log-probability feature functions ($\log P_{\text{BNN-p}}$): one feature for the word prediction model or two features for the stem and suffix models respectively, plus the penalty feature ε.

Table 7 shows the data used to train our English-Russian SMT system. The feature weights for all approaches were tuned by using pairwise ranking optimization (Hopkins and May, 2011) on the wmt12 benchmark (Callison-Burch et al., 2012). During tuning, 14 PRO parameter estimation runs are performed in parallel on different samples of the n-best list after each decoder iteration. The weights of the individual PRO runs are then averaged and passed on to the next decoding iteration. Performing weight estimation independently for a number of samples corrects for some of the instability that can be caused by individual samples. The wmt13 set (Bojar et al., 2013) was used for testing. We use approximate randomization (Noreen, 1989) to test for statistically significant differences between runs (Riezler and Maxwell, 2005).

Translation quality is measured with case-insensitive BLEU[%] using one reference translation. As shown in Table 8, statistically significant improvements over the respective baseline (Baseline and Base+suffLM) are marked ▲ at the $p < .01$ level. Integrating our bilingual neural network approach into our SMT system yields small but statistically significant improvements of 0.4 BLEU over a competitive baseline. We can also

SMT system	wmt12 (dev)	wmt13 (test)
Baseline	24.7	18.9
+ stem/suff. BNN	25.1	19.3▲
Base+suffLM	24.5	19.2
+ word BNN	24.5	19.3
+ stem/suff. BNN	24.7	19.6▲

Table 8: Effect of our BNN models on English-Russian translation quality (BLEU[%]).

see that it is beneficial to add a suffix-based language model to the baseline system. The biggest improvement is obtained by combining the suffix-based language model and our BNN approach, yielding 0.7 BLEU over a competitive, state-of-the-art baseline, of which 0.4 BLEU are due to our BNNs. Finally, one can see that the BNNs modeling stems and suffixes separately perform better than a BNN directly predicting fully inflected forms.

To better understand the BNN effect on the SMT system, we analyze the set of phrase pairs that are employed by the decoder to translate each sentence. This set is ranked by the weighted combination of phrase translation and lexical weighting scores, target language model score and, if available, phrase BNN scores. As shown in Table 9, the morphological BNN models have a positive effect on the decoder's lexical search space increasing the recall of reference tokens among the top 1 and 3 phrase translation candidates. The mean reciprocal rank (MRR) also improves from 0.655 to 0.662. Looking at the 1-best SMT output, we observe a slight increase of reference/output recall (50.0% to 50.7%), which is less than the increase we observe for the top 1 translation candidates (57.6% to 59.0%). One possible explanation is that the new, more accurate translation distributions are overruled by other SMT model scores,

Token recall (wmt12):	Baseline	+BNN
reference/MT-search-space [top-1]	57.6%	59.0%
reference/MT-search-space [top-3]	70.7%	70.9%
reference/MT-search-space [top-30]	86.0%	85.0%
reference/MT-search-space [MRR]	0.655	0.662
reference/MT-output	50.0%	50.7%
stem-only reference/MT-output	12.3%	11.5%
of which reachable	11.2%	10.3%

Table 9: Target word coverage analysis of the English-Russian SMT system before and after adding the morphological BNN models.

like the target LM, that are based on traditional maximum-likelihood estimates. While the suffix-based LMs proved beneficial in our experiments, we speculate that higher gains could be obtained by coupling our approach with a morphology-aware neural LM like the one recently presented by Botha and Blunsom (2014).

7 Related work

While most relevant literature has been discussed in earlier sections, the following approaches are particularly related to ours: Minkov et al. (2007) and Toutanova et al. (2008) address target inflection prediction with a log-linear model based on rich morphological and syntactic features. Their model exploits target context and is applied to inflect the output of a stem-based SMT system, whereas our models predict target words (or pairs of stem-suffix) independently and are integrated into decoding. Chahuneau et al. (2013) address the same problem with another feature-rich discriminative model that can be integrated in decoding, like ours, but they also use it to inflect on-the-fly stemmed phrases. It is not clear what part of their SMT improvements is due to the generation of new phrases or to better scoring. Jeong et al. (2010) predict surface word forms in context, similarly to our word BNN, and integrate the scores into the SMT system. Unlike us, they rely on linguistic feature-rich log-linear models to do that. Gimpel and Smith (2008) propose a similar approach to directly predict phrases in context, instead of words.

All those approaches employed features that capture the global structure of source sentences, like dependency relations. By contrast, our models access only local context in the source sentence but they achieve accuracy gains comparably to models that also use global sentence structure.

8 Conclusions

We have proposed a general approach to predict word translations in context using bilingual neural network architectures. Unlike previous NN approaches, we model word, stem and suffix distributions in the target language given context in the source language. Instead of relying on manually engineered features, our models automatically learn abstract word representations and features that are relevant for the modeled task directly from word-aligned parallel data. Our preliminary results with LBL and ConvNet architectures suggest that potential improvement may be achieved by factorizing target representations or by dynamically modeling source context size. Evaluated on three morphologically rich languages, our approach achieves considerable gains in word, stem and suffix accuracy over a context-independent maximum-likelihood baseline. Finally, we have shown that the proposed BNN models can be tightly integrated into a phrase-based SMT system, resulting in small but statistically significant BLEU improvement over a competitive, large-scale English-Russian baseline.

Our analysis shows that the number of correct target words occurring in highly scored phrase translation candidates increases after integrating the morphological BNNs. However, only few of these end up in the 1-best translation output. Future work will investigate the benefits of coupling our BNN models with target language models that also exploit abstract word representations, such as Botha and Blunsom (2014) and Auli et al. (2013).

Acknowledgments

This research was funded in part by the Netherlands Organization for Scientific Research (NWO) under project numbers 639.022.213 and 612.001.218. We would like to thank Ekaterina Garmash for helping with the error analysis of the English-Russian translations.

References

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, Washington, USA, October.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Arianna Bisazza and Christof Monz. 2014. Class-based language modeling for translating into morphologically rich languages. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, pages 1918–1927, Dublin, Ireland.

Ondřej Bojar, Zdeněk Žabokrtský, Ondřej Dušek, Petra Galuščáková, Martin Majliš, David Mareček, Jiří Maršík, Michal Novák, Martin Popel, and Aleš Tamchyna. 2012. The joy of parallelism with czeng

1.0. In *Proceedings of LREC2012*, Istanbul, Turkey, May. ELRA, European Language Resources Association.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August. Association for Computational Linguistics.

Jan A. Botha and Phil Blunsom. 2014. Compositional Morphology for Word Representations and Language Modelling. In *Proceedings of the 31st International Conference on Machine Learning*, Beijing, China, June.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June. Association for Computational Linguistics.

Victor Chahuneau, Eva Schlinger, Noah A. Smith, and Chris Dyer. 2013. Translating into morphologically rich languages with synthetic phrases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1677–1687, Seattle, USA, October.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 4(13):359–393.

Colin Cherry, Robert C. Moore, and Chris Quirk. 2012. On hierarchical re-ordering and permutation parsing for phrase-based decoding. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 200–209, Montréal, Canada, June. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th Annual International Conference on Machine Learning*, volume 12, pages 2493–2537.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 678–683, Sofia, Bulgaria, August.

Nadir Durrani, Alexander Fraser, and Helmut Schmid. 2013. Model with minimal translation units, but decode with phrases. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–11, Atlanta, Georgia, USA, June.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Morristown, NJ, USA. Association for Computational Linguistics.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 699–709. Association for Computational Linguistics.

Kevin Gimpel and Noah A. Smith. 2008. Rich source-side context for statistical machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 9–17, Columbus, Ohio, USA.

Spence Green and John DeNero. 2012. A class-based agreement model for generating accurately inflected translations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, ACL '12, pages 146–155, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rejwanul Haque, Sudip Kumar Naskar, Antal Bosch, and Andy Way. 2011. Integrating source-language context into phrase-based statistical machine translation. *Machine Translation*, 25(3):239–285, September.

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

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Yuening Hu, Michael Auli, Qin Gao, and Jianfeng Gao. 2014. Minimum translation modeling with recurrent neural networks. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 20–29, Gothenburg, Sweden, April. Association for Computational Linguistics.

Minwoo Jeong, Kristina Toutanova, Hisami Suzuki, and Chris Quirk. 2010. A discriminative lexicon model for complex morphology. In *The Ninth Conference of the Association for Machine Translation in the Americas*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, USA, October.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665. Association for Computational Linguistics.

Ahmed El Kholy and Nizar Habash. 2012. Translate, predict or generate: Modeling rich morphology in statistical machine translation. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133, Edmonton, Canada.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In Robert E. Frederking and Kathryn B. Taylor, editors, *Proceedings of the 6th Conference of the Association for Machine Translations in the Americas (AMTA 2004)*, pages 115–124.

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, J Gauvain, and François Yvon. 2011. Structured output layer neural network language model. In *Proceedings of Proceedings of ICASSP*.

Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. 2011. Modeling syntactic context improves morphological segmentation. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 1–9, Portland, Oregon, USA, June. Association for Computational Linguistics.

Arne Mauser, Saša Hasan, and Hermann Ney. 2009. Extending statistical machine translation with discriminative and trigger-based lexicon models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 210–218, Stroudsburg, PA, USA. Association for Computational Linguistics.

Einat Minkov, Kristina Toutanova, and Hisami Suzuki. 2007. Generating complex morphology for machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 128–135.

Andriy Mnih and Geoffrey E. Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, pages 641–648, New York, NY, USA.

Thomas Müller, Hinrich Schütze, and Helmut Schmid. 2012. A comparative investigation of morphological language modeling for the languages of the European Union. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 386–395, Montréal, Canada, June. Association for Computational Linguistics.

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

Chris Quirk and Arul Menezes. 2006. Do we need phrases? challenging the conventional wisdom in statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 9–16, New York City, USA, June. Association for Computational Linguistics.

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

Holger Schwenk, Daniel Dechelotte, and Jean-Luc Gauvain. 2006. Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL 2006 Conference*, pages 723–730, Sydney, Australia, July. Association for Computational Linguistics.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In *Proceedings of COLING*.

Serge Sharoff, Mikhail Kopotev, Tomaz Erjavec, Anna Feldman, and Dagmar Divjak. 2008. Designing and evaluating a russian tagset. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the*

Conference on Empirical Methods in Natural Language Processing, pages 151–161, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Le Hai Son, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 39–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Drahomíra Spoustová, Jan Hajič, Jan Votrubec, Pavel Krbec, and Pavel Květoň. 2007. The best of two worlds: Cooperation of statistical and rule-based taggers for czech. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing*, pages 67–74, Prague, Czech Republic, June. Association for Computational Linguistics.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. 2008. Applying morphology generation models to machine translation. In *Proceedings of the Association for Computational Linguistics*.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, Seattle, October.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398, Seattle, USA, October.

Dependency-Based Bilingual Language Models for Reordering in Statistical Machine Translation

Ekaterina Garmash and **Christof Monz**
Informatics Institute, University of Amsterdam
Science Park 904, 1098 XH Amsterdam, The Netherlands
{e.garmash,c.monz}@uva.nl

Abstract

This paper presents a novel approach to improve reordering in phrase-based machine translation by using richer, syntactic representations of units of bilingual language models (BiLMs). Our method to include syntactic information is simple in implementation and requires minimal changes in the decoding algorithm. The approach is evaluated in a series of Arabic-English and Chinese-English translation experiments. The best models demonstrate significant improvements in BLEU and TER over the phrase-based baseline, as well as over the lexicalized BiLM by Niehues et al. (2011). Further improvements of up to 0.45 BLEU for Arabic-English and up to 0.59 BLEU for Chinese-English are obtained by combining our dependency BiLM with a lexicalized BiLM. An improvement of 0.98 BLEU is obtained for Chinese-English in the setting of an increased distortion limit.

1 Introduction

In statistical machine translation (SMT) reordering (also called distortion) refers to the order in which source words are translated to generate the translation in the target language. Word orders can differ significantly across languages. For instance, Arabic declarative sentences can be verb-initial, while the corresponding English translation should realize the verb after the subject, hence requiring a reordering. Determining the correct reordering during decoding is a major challenge for SMT. This problem has received a lot of attention in the literature (see, e.g., Tillmann (2004), Zens and Ney (2003), Al-Onaizan and Papineni (2006)), as choosing the correct reordering improves readability of the translation and can have a substantial impact on translation quality (Birch, 2011). In this paper, we only consider those approaches that include a reordering feature function into the log-linear interpolation used during decoding.

The simplest reordering model is linear distortion (Koehn et al., 2003) which scores the distance between phrases translated at steps t and $t+1$ of the derivation. This model ignores any contextual information, as the distance between translated phrases is its only parameter. Lexical distortion modeling (Tillmann, 2004) conditions reordering probabilities on the phrase pairs translated at the current and previous steps. Unlike linear distortion, it characterizes reordering not in terms of distance but type: monotone, swap, or discontinuous.

In this paper, we base our approach to reordering on bilingual language models (Marino et al., 2006; Niehues et al., 2011). Instead of directly characterizing reordering, they model sequences of elementary translation events as a Markov process.[1] Originally, Marino et al. (2006) used this kind of model as the translation model, while more recently it has been used as an additional model in PBSMT systems (Niehues et al., 2011). We adopt and generalize the approach of Niehues et al. (2011) to investigate several variations of bilingual language models. Our method consists of labeling elementary translation events (tokens of bilingual LMs) with their different contextual properties.

What kind of contextual information should be incorporated in a reordering model? Lexical information has been used by Tillmann (2004) but is known to suffer from data sparsity (Galley and Manning, 2008). Also previous contributions to bilingual language modeling (Marino et al., 2006; Niehues et al., 2011) have mostly used lexical information, although Crego and Yvon (2010a) and Crego and Yvon (2010b) label bilingual to-

[1] Note that the standard PBSMT translation model assumes that events of translating separate phrases in a sentence are independent.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1689–1700,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

kens with a rich set of POS tags. But in general, reordering is considered to be a syntactic phenomenon and thus the relevant features are syntactic (Fox, 2002; Cherry, 2008). Syntactic information is incorporated in tree-based approaches in SMT, allowing one to provide a more detailed definition of translation events and to redefine decoding as parsing of a source string (Liu et al., 2006; Huang et al., 2006; Marton and Resnik, 2008), of a target string (Shen et al., 2008), or both (Chiang, 2007; Chiang, 2010). Reordering is a result of a given derivation, and CYK-based decoding used in tree-based approaches is more syntax-aware than the simple PBSMT decoding algorithm. Although tree-based approaches potentially offer a more accurate model of translation, they are also a lot more complex and requiring more intricate optimization and estimation techniques (Huang and Mi, 2010).

Our idea is to keep the simplicity of PBSMT but move towards the expressiveness typical of tree-based models. We incrementally build up the syntactic representation of a translation during decoding by adding precomputed fragments from the source parse tree. The idea to combine the merits of the two SMT paradigms has been proposed before, where Huang and Mi (2010) introduce incremental decoding for a tree-based model. On a very general level, our approach is similar to theirs in that it keeps track of a sequence of source syntactic subtrees that are being translated at consecutive decoding steps. An important difference is that they keep track of whether the visited subtrees have been fully translated, while in our approach, once a syntactic structural unit has been added to the history, it is not updated anymore.

In this paper, we focus on source syntactic information. During decoding we have full access to the source sentence, which allows us to obtain a better syntactic analysis (than for a partial sentence) and to precompute the units that the model operates with. We investigate the following research questions: How well can we capture reordering regularities of a language pair by incorporating source syntactic parameters into the units of a bilingual language model? What kind of source syntactic parameters are necessary and sufficient?

Our contributions can be summarized as follows: We argue that the contextual information used in the original bilingual models (Niehues et al., 2011) is insufficient and introduce a simple model that exploits source-side syntax to improve reordering (Sections 2 and 3). We perform a thorough comparison between different variants of our general model and compare them to the original approach. We carry out translation experiments on multiple test sets, two language pairs (Arabic-English and Chinese-English), and with respect to two metrics (BLEU and TER). Finally, we present a preliminary analysis of the reorderings resulting from the proposed models (Section 4).

2 Motivation

In this section, we elaborate on our research questions and provide background for our approach. We also discuss existing bilingual n-gram models and argue that they are often not expressive enough to differentiate between alternative reorderings. We should first note that the most commonly used n-gram model to distinguish between reorderings is a target language model, which does not take translation correspondence into account and just models target-side fluency. Al-Onaizan and Papineni (2006) show that target language models by themselves are not sufficient to correctly characterize reordering. In what follows we only discuss bilingual models.

The word-aligned sentence pair in Figure 1.a[2] demonstrates a common Arabic-English reordering. As stated in the introduction, bilingual language models capture reordering regularities as a sequence of elementary translation events[3]. In the given example, one could decompose the sequential process of translation as follows: First translate the first word *Alwzyr* as *the minister*, then *ArjE* as *attributed*, then *ArtfAE* as *the increase* and so on. The sequence of elementary translation events is modeled as an n-gram model (Equation 1, where t_i is a translation event). There are numerous ways in which t_i can be defined. Below we first discuss how they have been defined within previous approaches, and then introduce our definition.

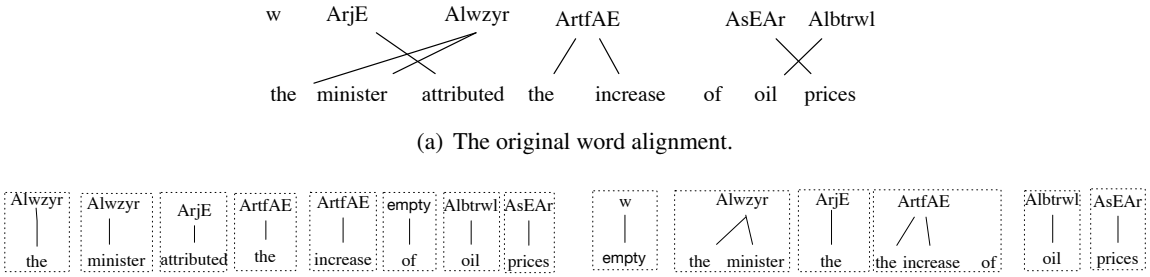
$$p(t_1, \ldots, t_m) = \prod_{i=1}^{m} p(t_i | t_{i-n+1} \ldots t_{i-1}) \quad (1)$$

2.1 Lexicalized bilingual LMs

By including both source and target information into the representation of translation events we ob-

[2]We used Buckwalter transliteration for Arabic words.

[3]By an *elementary* translation event we mean a translation of some substructure of a sentence.



(a) The original word alignment.

(b) BiLM tokens extracted from sentence (a).

(c) MTU tokens extracted from sentence (a).

Figure 1: Arabic-English parallel sentence, automatically word-aligned. The bilingual token sequences are produced according to two alternative definitions (BiLM and MTU).

tain a bilingual LM. The richer representation allows for a finer distinction between reorderings. For example, Arabic has a morphological marker of definiteness on both nouns and adjectives. If we first translate a definite adjective and then an indefinite noun, it will probably not be a likely sequence according to the translation model. This kind of intuition underlies the model of Niehues et al. (2011), a *bilingual LM* (BiLM), which defines elementary translation events $t_1, ..., t_n$ as follows:

$$t_i = \langle e_i, \{f | f \in A(e_i)\} \rangle, \quad (2)$$

where e_i is the i-th target word and $A : E \rightarrow \mathcal{P}(F)$ is an alignment function, E and F referring to target and source sentences, and $\mathcal{P}(\cdot)$ is the powerset function. In other words, the i-th translation event consists of the i-th target word and all source words aligned to it. Niehues et al. (2011) refer to the defined translation events t_i as *bilingual tokens* and we adopt this terminology.

There are alternative definitions of bilingual language models. Our choice of the above definition is supported by the fact that it produces an unambiguous segmentation of a parallel sentence into tokens. Ambiguous segmentation is undesirable because it increases the token vocabulary, and thus the model sparsity. Another disadvantage comes from the fact that we want to compare permutations of the same set of elements. For example, the two different segmentations of ba into $[ba]$ and $[b][a]$ still represent the same permutation of the sequence ab. In Figure 1 one can produce a segmentation of (*AsEAr Albtrwl*, *oil prices*) into (*Albtrwl*, *oil*) and (*AsEAr*, *prices*) or leave it as is. If we allow for both segmentations, the learnt probability parameters may be different for the sum of (*Albtrwl*, *oil*) and (*AsEAr*, *prices*) and for the unsegmented phrase.

Durrani et al. (2011) introduce an alternative method for unambiguous bilingual segmentation where tokens are defined as minimal phrases, called minimal translation units (MTUs). Figure 1 compares the BiLM and MTU tokenization for a specific example. Since Niehues et al. (2011) have shown their model to work successfully as an additional feature in combination with commonly used standard phrase-based features, we use their approach as the main point of reference and base our approach on their segmentation method. In the rest of the text we refer to Niehues et al. (2011) as the *original* BiLM.[4] At the same time, we do not see any specific obstacles for combining our work with MTUs.

2.2 Suitability of lexicalized BiLM to model reordering

As mentioned in the introduction, lexical information is not very well-suited to capture reordering regularities. Consider Figure 2.a. The extracted sequence of bilingual tokens is produced by aligning source words with respect to target words (so that they are in the same order), as demonstrated by the shaded part of the picture. If we substituted the Arabic translation of *Egyptian* for the Arabic translation of *Israeli*, the reordering should remain the same. What matters for reordering is the syntactic role or context of a word. By using unnecessarily fine-grained categories we risk running into sparsity issues.

Niehues et al. (2011) also described an alternative variant of the original BiLM, where words are substituted by their POS tags (Figure 2.a, shaded part). Also, however, POS information by itself may be insufficiently expressive to separate cor-

[4] Although, strictly speaking, it is not the original approach (see the references in Section 1).

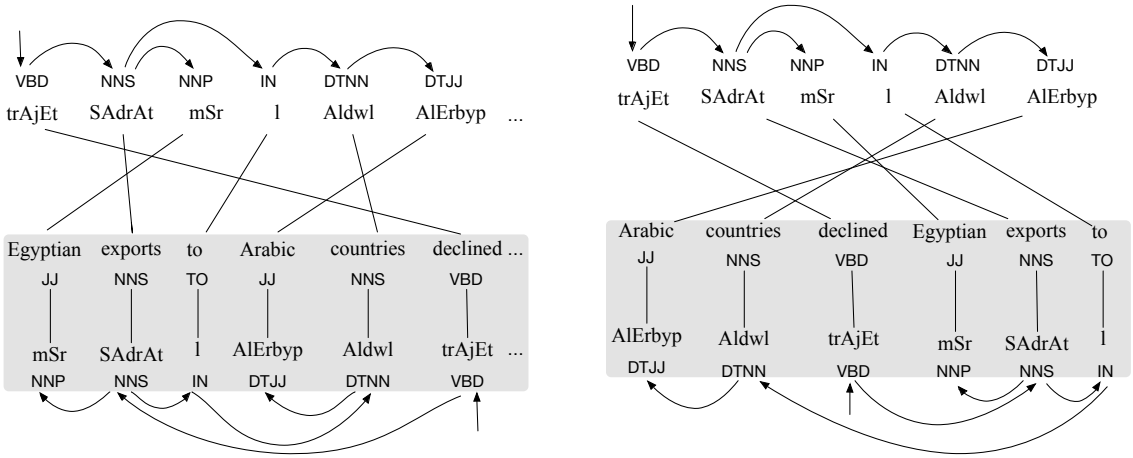


Figure 2: Arabic-English parallel sentence, automatically parsed and word-aligned, with corresponding sequences of bilingual tokens (in the shaded part). Comparison between translations produced via correct (a) and incorrect (b) reorderings.

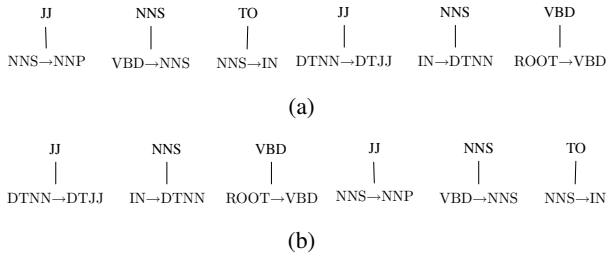


Figure 3: Sequences of bilingual tokens with source words substituted with their and their parents' POS tags: correct (a) and incorrect (b) reorderings.

rect and incorrect reorderings, see Figure 2.b. Although the corresponding sequence of POS-tag-substituted bilingual tokens is different from the correct sequence (Figure 2.b, shaded part), it still is a likely sequence. Indeed, the log-probabilities of the two sequences with respect to a 4-gram BiLM model[5] result in a higher probability of -10.25 for the incorrect reordering than for the correct one (-10.39).

Since fully lexicalized bilingual tokens suffer from data sparsity and POS-based bilingual tokens are insufficiently expressive, the question is which level of syntactic information strikes the right balance between expressiveness and generality.

[5]Section 4 contains details about data and software setup.

2.3 BiLM with dependency information

Dependency grammar is commonly used in NLP to formalize role-based relations between words. The intuitive notion of syntactic modification is captured by the primitive binary relation of dependence. Dependency relations do not change with the linear order of words (Figure 2) and therefore can provide a characterization of a word's syntactic class that invariant under reordering.

If we incorporate dependency relations into the representation of bilingual tokens, the incorrect reordering in Figure 2.b will produce a highly unlikely sequence. For example, we can substitute each source word with its POS tag and its parent's POS tag (Figure 3). Again, we computed 4-gram log-probabilities for the corresponding sequences: the correct reordering results in a substantially higher probability of -10.58 than the incorrect one (-13.48). We may consider situations where more fine-grained distinctions are required. In the next section, we explore different representations based on source dependency trees.

3 Dependency-based BiLM

In this section, we introduce our model which combines the BiLM from Niehues et al. (2011) with source dependency information. We further give details on how the proposed models are trained and integrated into a phrase-based decoder.

3.1 The general framework

In the previous section we outlined our framework as composed of two steps: First, a parallel sentence is tokenized according to the BiLM model (Niehues et al., 2011). Next, words in the bilingual tokens are substituted with their contextual properties. It is thus convenient to use the following generalized definition for a token sequence $t_1...t_n$ in our framework:

$$t_i = \langle ContE(e_i), \{ContF(f)|f \in A(e_i)\}\rangle, \quad (3)$$

where e_i is the i-th target word, $A : E \rightarrow \mathcal{P}(F)$ is an alignment function, F and E are source and target sentences, and $ContE$ and $ContF$ are target and source *contextual functions*, respectively. A contextual function returns a word's contextual property, based on its sentential context (source or target). See Figure 4 for an example of a sequence of BiLM tokens with a $ContF$ defined as returning the POS tag of the source word combined with the POS tags of its parent, grandparent and siblings, and $ContE$ defined as an identity function (see Section 3.2 for a detailed explanation of the functions and notation).

In this work we focus on source contextual functions ($ContF$). We also exploit some very simple target contextual functions, but do not go into an in-depth exploration.

3.2 Dependency-based contextual functions

In NLP approaches exploiting dependency structure, two kinds of relations are of special importance: the parent-child relation and the sibling relation. Shen et al. (2008) work with two well-formed dependency structures, both of which are defined in such a way that there is one common parent and a set of siblings. Li et al. (2012) characterize rules in hierarchical SMT by labeling them with the POS tags of the parents of the words inside the rule. Lerner and Petrov (2013) model reordering as a sequence of classification steps based on a dependency parse of a sentence. Their model first decides how a word is reordered with respect to its parent and then how it is reordered with respect to its siblings.

Based on these previous approaches, we propose to characterize contextual syntactic roles of a word in terms of POS tags of the words themselves and their relatives in a dependency tree. It is straightforward to incorporate parent information since each node has a unique parent. As for siblings information, we incorporate POS tags of the closest sibling to the left and the closest to the right. We do not include all of the siblings to avoid overfitting. In addition to these basic syntactic relations, we consider the grandparent relation.

The following list is a summary of the source contextual functions that we use. We describe a function with respect to the kind of contextual property of a word it returns: (i) the word itself (Lex); (ii) POS label of the word (Pos); (iii) POS label of the word's parent; (iv) POS of the word's closest sibling to the left, concatenated with the POS tag of the closest sibling to the right; (v) the POS label of the word's grandparent. We use target-side contextual functions returning: (i) an empty string, (ii) POS of the word, (iii) the word itself.

Notation. We do not use the above functions separately to define individual BiLM models, but use combinations of these functions. We use the following notation for function combinations: "•" horizontally connects source (on the left) and target (on the right) contextual functions for a given model. For example, Lex•Lex refers to the original (lexicalized) BiLM. We use arrows (→) to designate parental information (the arrow goes from parent to child). Pos→Pos refers to a combination of a function returning the POS of a word and the POS of its parent (as in Figure 3). Pos→Pos→Pos is a combination of the previous with the function returning the grandparent's POS. Finally, we use +sibl to indicate the use of the sibling function described above: For example, Pos→Pos+sibl is a source function that returns the word's POS, its parent's POS and the POS labels of the closest siblings to left and right.[6] Pos+sibl→Pos is a source function returning the word's own POS, the POS of a word's parent, and the POS tags of the parent's siblings (left- and right-adjacent).

Figure 4 represents the sentence from Figure 2 during decoding in a system with an integrated Pos→Pos→Pos+sibl•Lex feature. It shows the sequence of produced bilingual tokens and corresponding labels in the introduced notation.

3.3 Training

Training of dependency-based BiLMs consists of a sequence of extraction steps: After having produced word-alignments for a bitext (Section 4),

[6] In case there is no sibling on one of the sides, ϵ (empty word) is returned.

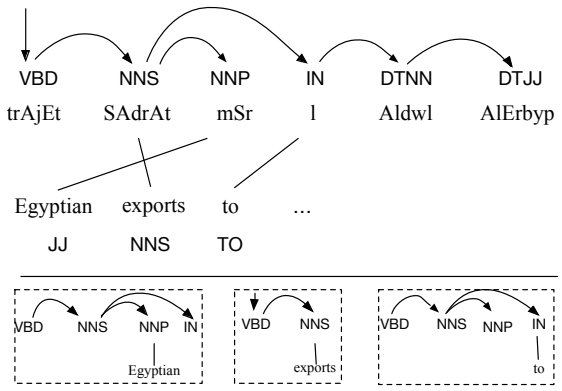


Figure 4: Sequence of bilingual tokens produced by a Pos→Pos→Pos+sibl•Lex after translating three words of the source sentence: VBD→NNS→ϵ+NNS+IN•Egyptian, ROOT→VBD→ϵ+NNS+ϵ•exports, VBD→NNS→NNP+IN+ϵ•to (if there is no sibling on either of the sides, ϵ is returned).

sentences are segmented according to Equation 3. We produce a dependency parse of a source sentence and a POS-tag labeling of a target sentence. For Chinese, we use the Stanford dependency parser (Chang et al., 2009). For Arabic a dependency parser is not available for public use, so we produce a constituency parse with the Stanford parser (Green and Manning, 2010) and extract dependencies based on the rules in Collins (1999). For English POS-tagging, we use the Stanford POS-tagger (Toutanova et al., 2003). After having produced a labeled sequence of tokens, we learn a 5-gram model using SRILM (Stolcke et al., 2011). Kneyser-Ney smoothing is used for all model variations except for Pos•Pos where Witten-Bell smoothing is used due to zero count-of-counts.

3.4 Decoder integration

Dependency-based BiLMs are integrated into our phrase-based SMT decoder as follows: Before translating a sentence, we produce its dependency parse. Phrase-internal word-alignments, needed to segment the translation hypothesis into tokens, are stored in the phrase table, based on the most frequent internal alignment observed during training. Likewise, we store the most likely target-side POS-labeling for each phrase pair.

The decoding algorithm is augmented with one additional feature function and one additional, corresponding feature weight. At each step of the derivation, as a new phrase pair is added to the partial translation hypothesis, this function segments the new phrase into bilingual tokens (given the internal alignment information) and substitutes the words in the phrase pair with syntactic labels (given the source parse and the target POS labeling associated with the phrase). The new syntactified bilingual tokens are added to the stack of preceding $n-1$ tokens, and the feature function computes the weighted updated model probability. During decoding, the probabilities of the BiLMs are computed in a stream-based fashion, with bilingual tokens as string tokens, and not in a class-based fashion, with syntactic source-side representations emitting the corresponding target words (Bisazza and Monz, 2014).

Training set	N. of lines	N. of tokens
Source side of Ar-En set	4,376,320	148M
Target side of Ar-En set	4,376,320	146M
Source side of Ch-En set	2,104,652	20M
Target side of Ch-En set	2,104,652	28M

Table 1: Training data for Arabic-English and Chinese-English experiments.

4 Experiments

4.1 Setup

We conduct translation experiments with a baseline PBSMT system with additionally one of the dependency-based BiLM feature functions specified in Section 3. We compare the translation performance to a baseline PBSMT system and to a baseline augmented with the original BiLMs from (Niehues et al., 2011).

Word-alignment is produced with GIZA++ (Och and Ney, 2003). We use an in-house implementation of a PBSMT system similar to Moses (Koehn et al., 2007). Our baseline contains all standard PBSMT features including language model, lexical weighting, and lexicalized reordering. The distortion limit is set to 5. A 5-gram LM is trained on the English Gigaword corpus (1.6B tokens) using SRILM with modified Kneyser-Ney smoothing and interpolation. The BiLMs were trained as described in Section 3.3. Information about the parallel data used for training the Arabic-English[7] and Chinese-English systems[8] is

[7]The following Arabic-English parallel corpora were used: LDC2006E25, LDC2004T18, several gale corpora, LDC2004T17, LDC2005E46, LDC2007T08, LDC2004E13.

[8]The following Chinese-English parallel corpora were used: LDC2002E18, LDC2002L27, LDC2003E07, LDC2003E14, LDC2005T06, LDC2005T10, LDC2005T34,

	Configuration	MT08		MT09		MT08+MT09	
		BLEU	TER	BLEU	TER	BLEU	TER
a	PBSMT baseline	45.12	47.94	48.16	44.30	46.57	46.21
b	Lex•Lex	45.27	47.79	$48.85^{\blacktriangle}$	$43.96^{\triangle}$	$46.98^{\blacktriangle}$	$45.96^{\triangle}$
	Pos•Pos	44.80	47.84	48.22	$44.14^{\triangle,-}$	46.44	46.07
c	Pos→Pos•Pos	$45.66^{\blacktriangle,\triangle}$	$\mathbf{47.17}^{\blacktriangle,\blacktriangle}$	$49.00^{\blacktriangle,-}$	$\mathbf{43.45}^{\blacktriangle,\blacktriangle}$	$47.25^{\blacktriangle,\triangle}$	$\mathbf{45.40}^{\blacktriangle,\blacktriangle}$
d	Pos→Pos−sibl•Pos	$45.46^{\triangle,-}$	$47.45^{\blacktriangle,\triangle}$	$48.69^{\blacktriangle,-}$	$43.64^{\blacktriangle,\triangle}$	$47.00^{\blacktriangle,-}$	$45.64^{\blacktriangle,-}$
e	Pos→Pos→Pos•Pos	$\mathbf{45.68}^{\blacktriangle,\triangle}$	$47.42^{\blacktriangle,\triangle}$	$49.09^{\blacktriangle,-}$	$43.59^{\blacktriangle,\blacktriangle}$	$\mathbf{47.30}^{\blacktriangle,\triangle}$	$45.60^{\blacktriangle,\blacktriangle}$
f	Lex•Lex + Pos→Pos→Pos•Pos	$45.63^{\blacktriangle,\triangle}$	$47.48^{\blacktriangle,\triangle}$	$\mathbf{49.30}^{\blacktriangle,\blacktriangle}$	$43.60^{\blacktriangle,\triangle}$	$\mathbf{47.38}^{\blacktriangle,\blacktriangle}$	$45.63^{\blacktriangle,\blacktriangle}$

Table 2: BLEU and TER scores for Arabic-English experiments. Statistically significant improvements over the baseline (a) are marked $^{\blacktriangle}$ at the $p < .01$ level and $^{\triangle}$ at the $p < .05$ level. Additionally, $^{\cdot,\blacktriangle}$ and $^{\cdot,\triangle}$ indicate significant improvements with respect to BiLM Lex•Lex (b). Since TER is an error rate, lower scores are better.

Configuration	MT08		MT09		MT08+MT09	
	BLEU	TER	BLEU	TER	BLEU	TER
Pos→Pos• ϵ	$45.66^{\blacktriangle,\triangle}$	$47.44^{\blacktriangle,\triangle}$	$48.78^{\blacktriangle,-}$	$43.94^{\blacktriangle,-}$	$47.15^{\blacktriangle,-}$	$45.77^{\blacktriangle,\triangle}$
Pos→Pos•Pos	$45.66^{\blacktriangle,\triangle}$	$\mathbf{47.17}^{\blacktriangle,\blacktriangle}$	$49.00^{\blacktriangle,-}$	$\mathbf{43.45}^{\blacktriangle,\blacktriangle}$	$47.25^{\blacktriangle,\triangle}$	$\mathbf{45.40}^{\blacktriangle,\blacktriangle}$
Pos→Pos•Lex	$45.48^{\triangle,-}$	$47.34^{\blacktriangle,\blacktriangle}$	$48.90^{\blacktriangle,-}$	$43.87^{\blacktriangle,\triangle}$	$47.12^{\blacktriangle,-}$	$45.69^{\blacktriangle,\blacktriangle}$

Table 3: Different combinations of a target contextual function with the Pos→Pos source contextual function for Arabic-English. See Table 2 for the notation regarding statistical significance.

shown in Table 1.

The feature weights were tuned by using pairwise ranking optimization (Hopkins and May, 2011) on the MT04 benchmark (for both language pairs). During tuning, 14 PRO parameter estimation runs are performed in parallel on different samples of the n-best list after each decoder iteration. The weights of the individual PRO runs are then averaged and passed on to the next decoding iteration. Performing weight estimation independently for a number of samples corrects for some of the instability that can be caused by individual samples. For testing, we used MT08 and MT09 for Arabic, and MT06 and MT08 for Chinese. We use approximate randomization (Noreen, 1989; Riezler and Maxwell, 2005) to test for statistically significant differences.

In the next two subsections we discuss the general results for Arabic and Chinese, where we use case-insensitive BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) as evaluation metrics. This is followed by a preliminary analysis of observed reorderings where we compare 4-gram precision results and conduct experiments with an increased distortion limit.

4.2 Arabic-English translation experiments

We are interested in how a translation system with an integrated dependency-based BiLM feature performs as compared to the standard PBSMT baseline and, more importantly, to the original BiLM model. We consider two variants of BiLM discussed by Niehues et al. (2011): the standard one, Lex•Lex, and the simplest syntactic one, Pos•Pos. Results for the experiments can be found in Table 2. In the discussion below we mostly focus on the experimental results for the large, combined test set MT08+MT09.

Table 2.a–b compares the performance of the baseline and original BiLM systems. Lex•Lex yields strongly significant improvements over the baseline for BLEU and weakly significant improvements for TER. Therefore, for the rest of the experiments we are interested in obtaining further improvements over Lex•Lex.

Pos→Pos•Pos (Table 2.c) demonstrates the effect of adding minimal dependency information to a BiLM.[9] It results in strongly significant improvements over the baseline and weak improvements over Lex•Lex in terms of BLEU. We additionally ran experiments with the different target functions (Table 3). •Pos shows the highest results, and •ϵ the lowest ones: this implies that a rather expressive source syntactic representation alone still benefits from target-side syntactic information. Below, our dependency-based systems only use •Pos.

Next, we tested the effect of adding more source

and several gale corpora.

[9] Additional significance testing, which is not shown in Table 2, shows a strongly significant improvement over the original syntactic BiLM Pos•Pos.

	Configuration	MT06		MT08		MT06+MT08	
		BLEU	TER	BLEU	TER	BLEU	TER
a	PBSMT baseline	31.89	57.79	25.53	60.71	28.99	59.14
b	Lex•Lex	$32.84^{\blacktriangle}$	$57.40^{\blacktriangle}$	$25.91^{\triangle}$	$60.23^{\blacktriangle}$	$29.69^{\blacktriangle}$	$58.72^{\blacktriangle}$
	Pos•Pos	$32.31^{\blacktriangle}$	57.89	25.66	60.79	29.28	59.24
c	Pos→Pos•Pos	$32.86^{\blacktriangle,-}$	$57.05^{\blacktriangle,\triangle}$	$26.09^{\blacktriangle,-}$	$59.87^{\blacktriangle,\triangle}$	$29.78^{\blacktriangle,-}$	$58.36^{\blacktriangle,\blacktriangle}$
d	Pos→Pos−sibl•Pos	$32.27^{\triangle,-}$	$\mathbf{56.63}^{\blacktriangle,\triangle}$	25.75	$\mathbf{59.47}^{\blacktriangle,\blacktriangle}$	$29.30^{\triangle,-}$	$\mathbf{57.95}^{\blacktriangle,\blacktriangle}$
e	Pos→Pos→Pos•Pos	$33.09^{\blacktriangle,-}$	57.54	$26.35^{\blacktriangle,\triangle}$	$59.70^{\blacktriangle,\blacktriangle}$	$\mathbf{30.05}^{\blacktriangle,\blacktriangle}$	$58.54^{\blacktriangle,-}$
f	Lex•Lex + Pos→Pos→Pos•Pos	$\mathbf{33.43}^{\blacktriangle,\blacktriangle}$	$57.00^{\blacktriangle,\blacktriangle}$	$\mathbf{26.50}^{\blacktriangle,\blacktriangle}$	$\mathbf{59.79}^{\blacktriangle,\blacktriangle}$	$\mathbf{30.28}^{\blacktriangle,\blacktriangle}$	$58.30^{\blacktriangle,\blacktriangle}$

Table 4: BLEU and TER scores for Chinese-English PBSMT baseline and BiLM pipelines. See Table 2 for the notation regarding statistical significance.

Configuration	MT06		MT08		MT06+MT08	
	BLEU	TER	BLEU	TER	BLEU	TER
Pos→Pos• ϵ	$32.43^{\blacktriangle,-}$	$57.42^{\blacktriangle,-}$	25.84	60.51	$29.43^{\blacktriangle,-}$	$58.86^{\blacktriangle,-}$
Pos→Pos•Pos	$32.86^{\blacktriangle,-}$	$57.05^{\blacktriangle,\triangle}$	$26.09^{\blacktriangle,-}$	$59.87^{\blacktriangle,\triangle}$	$29.78^{\blacktriangle,-}$	$58.36^{\blacktriangle,\blacktriangle}$
Pos→Pos•Lex	$32.69^{\blacktriangle,-}$	$57.03^{\blacktriangle,\triangle}$	25.72	$60.17^{\blacktriangle,-}$	$29.52^{\blacktriangle,-}$	$58.49^{\blacktriangle,\triangle}$

Table 5: Different combinations of a target contextual function with the Pos→Pos source contextual function for Chinese-English. See Table 2 for the notation regarding statistical significance.

dependency information. Pos→Pos+sibl•Pos (Table 2.d) only improves over the PBSMT baseline (but also shows weak improvements over Lex•Lex for TER). It significantly degrades the performance with respect to the Pos→Pos•Pos system (Table 2.c). Pos→Pos→Pos•Pos (Table 2.e) shows the best results overall for BLEU, although it must be pointed out that the difference with Pos→Pos•Pos is very small. With respect to TER, Pos→Pos•Pos outperforms the grandparent variant.

So far, we can conclude that source parent information helps improve translation performance. Increased specificity of a parent (parent specified by a grandparent) tends to further improve performance. Up to now, we have only used syntactic information and obtained considerable improvements over Pos•Pos, surpassing the improvement provided by Lex•Lex. Can we gain further improvements by also adding lexical information? To this end, we conduct experiments combining the best performing dependency-based BiLM (Pos→Pos→Pos•Pos) and the lexicalized BiLM (Lex•Lex). We hypothesize that the two models improve different aspects of translation: Lex•Lex is biased towards improving lexical choice and Pos→Pos→Pos•Pos towards improving reordering. Combining these two models, we may improve both aspects. The metric results for the combined set indeed support this hypothesis (Table 2.f).

4.3 Chinese-English translation experiments

The results of the Chinese-English experiments are shown in Table 4. In the discussion below we mostly focus on the experimental results for the large, combined test set MT06+MT08. We observe the same general pattern for the Pos→Pos source function (Table 4.c) as for Arabic-English: the system with the •Pos target function has the highest scores (Table 5). All of the Pos→Pos• configurations show statistically significant improvements over the PBSMT baseline. For TER, two of the three Pos→Pos• variants significantly outperform Lex•Lex. The system with sibling information (Table 4.d) obtains quite low BLEU results, just as in the Arabic experiments. On the other hand, its TER results are the highest overall. The system with the Pos→Pos→Pos•Pos function (Table 4.e) achieves the best results among dependency-based BiLMs for BLEU. Finally, combining Pos→Pos→Pos•Pos and Lex•Lex results in the largest and significant improvements over all competing systems for BLEU.

4.4 Preliminary analysis of reordering in translation experiments

In general, the experimental results show that using source dependency information yields consistent improvements for translating from Arabic and Chinese into English. On the other hand, we have pointed out some discrepancies between the two metrics employed, suggesting that different system configurations may improve different aspects

	Configuration	Ar-En			Ch-En		
		MT08	MT09	MT08+MT09	MT06	MT08	MT06+MT08
a	PBSMT baseline	26.14	29.81	27.88	14.48	10.96	12.89
b	Lex•Lex	26.33	30.55	28.32	15.43	11.45	13.65
	Pos•Pos	25.95	30.06	27.89	14.76	11.01	13.07
c	Pos→Pos•Pos	**26.91**	31.08	28.87	15.29	11.52	13.60
e	Pos→Pos−sibl•Pos	26.71	30.73	28.60	15.27	11.67	13.66
d	Pos→Pos→Pos•Pos	26.78	31.09	28.80	**15.42**	**11.70**	**13.77**
f	Lex•Lex + Pos→Pos→Pos•Pos	26.80	**31.27**	**28.90**	**15.87**	**11.85**	**14.07**

Table 6: 4-gram precision scores for Arabic-English and Chinese-English baseline and BiLM systems.

Configuration	MT08			MT09			MT08+MT09		
	BLEU	TER	4gram	BLEU	TER	4gram	BLEU	TER	4gram
Lex•Lex	45.19	47.06	26.41	48.39	44.11	30.23	46.72	45.97	28.21
Pos→Pos→Pos•Pos	45.49	47.31$^{\triangle}$	26.66	48.90$^{\blacktriangle}$	43.57$^{\blacktriangle}$	30.92	47.12$^{\blacktriangle}$	45.52$^{\blacktriangle}$	28.66

Table 7: BLEU, TER and 4-gram precision scores for Arabic-English Lex•Lex and Pos→Pos→Pos•Pos with a distortion limit of 10.

Configuration	MT06			MT08			MT06+MT08		
	BLEU	TER	4gram	BLEU	TER	4gram	BLEU	TER	4gram
Lex•Lex	33.26	56.81	16.06	25.67	60.19	11.42	29.79	58.38	13.96
Pos→Pos→Pos•Pos	33.92$^{\blacktriangle}$	56.29$^{\blacktriangle}$	16.26	27.00$^{\blacktriangle}$	59.58$^{\blacktriangle}$	12.26	30.77$^{\blacktriangle}$	57.82$^{\blacktriangle}$	14.46

Table 8: BLEU, TER and 4-gram precision scores for Chinese-English Lex•Lex and Pos→Pos→Pos•Pos with a distortion limit of 10.

of translation. To this end, we conducted some additional evaluations to understand how reordering is affected by the proposed features.

We use 4-gram precision as a metric of how much of the reference set word order is preserved. Table 6 shows the corresponding results for both languages. Just as in the previous two sections, configurations with parental information produce the best results. For Arabic, all of the dependency configurations outperform Lex•Lex. But the system with two feature functions, one of which is Lex•Lex, still obtains the best results, which may suggest that the lexicalized BiLM also helps to differentiate between word orders. For Chinese, Pos→Pos→Pos•Pos and the system combining the latter and Lex•Lex also obtain the best results. However, other dependency-based configurations do not outperform Lex•Lex.

All the experiments so far were run with a distortion limit of 5. But both of the languages, especially Chinese, often require reorderings over a longer distance. We performed additional experiments with a distortion limit of 10 for the Lex•Lex and Pos→Pos→Pos•Pos systems (Tables 7 and 8). It is more difficult to translate with a higher distortion limit (Green et al., 2010) as the set of permutations grows larger thereby making it more difficult to differentiate between correct and incorrect continuations of the current hypothesis. It has also been noted that higher distortion limits are more likely to result in improvements for Chinese rather than Arabic to English translation (Chiang, 2007; Green et al., 2010).

We compared performance of fixed BiLM models at distortion lengths of 5 and 10. Arabic-English results did not reveal statistically significant differences between the two distortion limits for Pos→Pos→Pos•Pos. On the other hand, for Lex•Lex BLEU decreases when using a distortion limit of 10 compared to a limit of 5. This implies that the dependency BiLM is more robust in the more challenging reordering setting than the lexicalized BiLM. Chinese-English results for Pos→Pos→Pos•Pos do show significant improvements over the distortion limit of 5 (up to 0.49 BLEU higher than the best result in Table 4). This indicates that the dependency-based BiLM is better capable to take advantage of the increased distortion limit and discriminate between correct and incorrect reordering choices.

Comparing the results for Pos→Pos→Pos•Pos and Lex•Lex at a distortion limit of 10, we obtain strongly significant improvements for all metrics. For Chinese, a larger distortion limit helps for both configurations, but more so for our dependency BiLM, yielding an improvement of 0.98 BLEU

over the original, lexicalized BiLM (Table 8).

5 Conclusions

In this paper, we have introduced a simple, yet effective way to include syntactic information into phrase-based SMT. Our method consists of enriching the representation of units of a bilingual language model (BiLM). We argued that the very limited contextual information used in the original bilingual models (Niehues et al., 2011) can capture reorderings only to a limited degree and proposed a method to incorporate information from a source dependency tree in bilingual units. In a series of translation experiments we performed a thorough comparison between various syntactically-enriched BiLMs and competing models. The results demonstrated that adding syntactic information from a source dependency tree to the representations of bilingual tokens in an n-gram model can yield statistically significant improvements over the competing systems.

A number of additional evaluations provided an indication for better modeling of reordering phenomena. The proposed dependency-based BiLMs resulted in an increase in 4-gram precision and provided further significant improvements over all considered metrics in experiments with an increased distortion limit.

In this paper, we have focused on rather elementary dependency relations, which we are planning to expand on in future work. Our current approach is still strictly tied to the number of target tokens. In particular, we are interested in exploring ways to better capture the notion of syntactic cohesion in translation (Fox, 2002; Cherry, 2008) within our framework.

Acknowledgments

We thank Arianna Bisazza and the reviewers for their useful comments. This research was funded in part by the Netherlands Organization for Scientific Research (NWO) under project numbers 639.022.213 and 612.001.218.

References

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 529–536, Sydney, Australia, July. Association for Computational Linguistics.

Alexandra Birch. 2011. *Reordering Metrics for Statistical Machine Translation*. Ph.D. thesis, University of Edinburgh.

Arianna Bisazza and Christof Monz. 2014. Class-based language modeling for translating into morphologically rich languages. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 1918–1927, Dublin, Ireland, August.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D. Manning. 2009. Discriminative reordering with chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*, pages 51–59. Association for Computational Linguistics.

Colin Cherry. 2008. Cohesive phrase-based decoding for statistical machine translation. In *Proceedings of Association for Computational Linguistics*, pages 72–80.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452. Association for Computational Linguistics.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Josep M. Crego and François Yvon. 2010a. Factored bilingual n-gram language models for statistical machine translation. *Machine Translation*, 24(2):159–175.

Josep M. Crego and François Yvon. 2010b. Improving reordering with linguistically informed bilingual n-grams. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 197–205. Association for Computational Linguistics.

Nadir Durrani, Helmut Schmid, and Alexander Fraser. 2011. A joint sequence translation model with integrated reordering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 1045–1054. Association for Computational Linguistics.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 304–3111. Association for Computational Linguistics.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 848–856. Association for Computational Linguistics.

Spence Green and Christopher D. Manning. 2010. Better arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 394–402. Association for Computational Linguistics.

Spence Green, Michel Galley, and Christopher D. Manning. 2010. Improved models of distortion cost for statistical machine translation. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 867–875. Association for Computational Linguistics.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362. Association for Computational Linguistics.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 273–283. Association for Computational Linguistics.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*, pages 223–226.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

Uri Lerner and Slav Petrov. 2013. Source-side classifier preordering for machine translation. In *Proceedings of the Empirical Methods in Natural Language Processing*.

Junhui Li, Zhaopeng Tu, Guodong Zhou, and Josef van Genabith. 2012. Head-driven hierarchical phrase-based translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 33–37. Association for Computational Linguistics.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 609–616. Association for Computational Linguistics.

José B Marino, Rafael E Banchs, Josep M. Crego, Adria de Gispert, Patrik Lambert, José A.R. Fonollosa, and Marta R. Costa-Jussà. 2006. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of the Association for Computational Linguistics*, pages 1003–1011.

Jan Niehues, Teresa Herrmann, Stephan Vogel, and Alex Waibel. 2011. Wider context by using bilingual language models in machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 198–206. Association for Computational Linguistics.

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

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

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

Libin Shen, Jinxi Xu, and Ralph M. Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of the Association for Computational Linguistics*, pages 577–585.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*, pages 223–231.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. Srilm at sixteen: Update and outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, page 5.

Christoph Tillmann. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of of the North American Chapter of the Association for Computational Linguistics*, pages 101–104. Association for Computational Linguistics.

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

Richard Zens and Hermann Ney. 2003. A comparative study on reordering constraints in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 144–151. Association for Computational Linguistics.

Combining String and Context Similarity for Bilingual Term Alignment from Comparable Corpora

Georgios Kontonatsios[1,2] **Ioannis Korkontzelos**[1,2] **Jun'ichi Tsujii**[3] **Sophia Ananiadou**[1,2]
National Centre for Text Mining, University of Manchester, Manchester, UK[1]
School of Computer Science, University of Manchester, Manchester, UK[2]
Microsoft Research Asia, Beijing, China[3]
{gkontonatsios,ikorkontzelos,sananiadou}@cs.man.ac.uk
jtsujii@microsoft.com

Abstract

Automatically compiling bilingual dictionaries of technical terms from comparable corpora is a challenging problem, yet with many potential applications. In this paper, we exploit two independent observations about term translations: (a) terms are often formed by corresponding sub-lexical units across languages and (b) a term and its translation tend to appear in similar lexical context. Based on the first observation, we develop a new character n-gram compositional method, a logistic regression classifier, for learning a string similarity measure of term translations. According to the second observation, we use an existing context-based approach. For evaluation, we investigate the performance of compositional and context-based methods on: (a) similar and unrelated languages, (b) corpora of different degree of comparability and (c) the translation of frequent and rare terms. Finally, we combine the two translation clues, namely string and contextual similarity, in a linear model and we show substantial improvements over the two translation signals.

1 Introduction

Bilingual dictionaries of technical terms are resources useful for various tasks, such as computer-aided human translation (Dagan and Church, 1994; Fung and McKeown, 1997), *Statistical Machine Translation* (Och and Ney, 2003) and *Cross-Language Information Retrieval* (Ballesteros and Croft, 1997). In the last two decades, researchers have focused on automatically compiling bilingual term dictionaries either from *parallel* (Smadja et al., 1996; Van der Eijk, 1993) or *comparable* corpora (Rapp, 1999; Fung and Yee, 1998). While parallel corpora contain the same sentences in two languages, comparable corpora consist of bilingual pieces of text that share some features, only, such as topic, domain, or time period. Comparable corpora can be constructed more easily than parallel corpora. Freely available, up-to-date, on-line resources (e.g., Wikipedia) can be employed.

In this paper, we exploit two different sources of information to extract bilingual terminology from comparable corpora: the *compositional* and the *contextual clue*. The *compositional clue* is the hypothesis that the representations of a term in any pair of languages tend to consist of corresponding lexical or sub-lexical units, e.g., prefixes, suffices and morphemes. In order to capture associations of textual units across languages, we investigate three different character n-gram approaches, namely a *Random Forest* (RF) classifier (Kontonatsios et al., 2014), *Support Vector Machines* with an RBF kernel (SVM-RBF) and a *Logistic Regression* (LogReg) classifier. Whilst the previous approaches take as an input monolingual features and then try to find cross-lingual mappings, our proposed method (LogReg classifier) considers multilingual features, i.e., tuples of co-occurring n-grams.

The *contextual* clue is the hypothesis that mutual translations of a term tend to occur in similar lexical context. Context-based approaches are unsupervised methods that compare the context distributions of a source and a target term. A bilingual seed dictionary is used to map context vector dimensions of two languages. Li and Gaussier (2010) suggested that the seed dictionary can be used to estimate the degree of comparability of a bilingual corpus. Given a seed dictionary, the *corpus comparability* is the expectation of finding for each word of the source corpus, its translation in the target part of the corpus. The performance of context-based methods has been shown to depend on the frequency of terms to be translated and the

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1701–1712,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

corpus comparability. In this work, we use an existing distributional semantics approach to locate term translations.

Furthermore, we hypothesise that the compositional and contextual clue are orthogonal, since the former considers the internal structure of terms while the latter exploits the surrounding lexical context. Based on the above hypothesis, we combine the two translation clues in a linear model.

For experimentation, we construct comparable corpora for four language pairs (English-Spanish, English-French, English-Greek and English-Japanese) of the biomedical domain.

We choose this domain because a large proportion of the medical terms tends to compositionally translate across languages (Lovis et al., 1997; Namer and Baud, 2007). Additionally, given the vast amount of newly introduced terms (neologisms) in the medical domain (Pustejovsky et al., 2001), term alignment methods are needed in order to automatically update existing resources.

We investigate the following aspects of term alignment: (a) the performance of compositional methods on closely related and on distant languages, (b) the performance of context vectors and compositional methods when translating frequent or rare terms, (c) the degree to which the corpus comparability affects the performance of context-based and compositional methods (d) the improvements that we can achieve when we combine the compositional and context clue.

Our experiments show that the performance of compositional methods largely depends on the distance between the two languages. The performance of the context-based approach is greatly affected by corpus-specific parameters (the frequency of occurrence of the terms to be translated and the degree of corpora comparability). It is also shown that the combination of compositional and contextual methods performs better than each of the clues, separately. Combined systems can be deployed in application environments with different language pairs, comparable corpora and seeds dictionaries.

The LogReg, dictionary extraction method described in this paper is freely available [1].

[1] http://personalpages.manchester.ac.uk/postgrad/georgios.kontonatsios/Software/LogReg-TermAlign.tar.gz

2 Related Work

Context-based methods (Fung and Yee, 1998; Rapp, 1999) adapt the *Distributional Hypothesis* (Harris, 1954), i.e., words that occur in similar lexical context tend to have the same meaning, in a multilingual environment. They represent the context of each term t as a context vector, usually following the *bag-of-words* model. Each dimension of the vector corresponds to a context word occurring within a predefined window, while the corresponding value is computed by a correlation metric, e.g., Log-Likelihood Ratio (Morin et al., 2007; Chiao and Zweigenbaum, 2002) or Point-wise Mutual Information (Andrade et al., 2010). A general bilingual dictionary is then used to translate/project the target context vectors into the source language. As a result, the source and target context vectors become directly comparable. In a final step, candidate translations are being ranked according to a distance metric, e.g., cosine similarity (Tamura et al., 2012) or Jaccard index (Zanzotto et al., 2010; Apidianaki et al., 2012).

Whilst context-based methods have become a common practise for bilingual dictionary extraction from comparable corpora, nonetheless, their performance is subject to various factors, one of which is the quality of the comparable corpus. Li and Gaussier (2010) introduced the corpus comparability metric and showed that it is related to the performance of context vectors. The higher the corpus comparability is, the higher the performance of context vectors is. Furthermore, context vector approaches are sensitive to the frequency of terms. For frequent terms, distributional semantics methods exhibit robust performance since the corresponding context is more informative. Chiao and Zweigenbaum (2002) reported an accuracy of 91% for the top 20 candidates when translating terms that occur 100 times or more. However, the performance of context vectors drastically decreases for lower frequency terms (Kontonatsios et al., 2014; Morin and Daille, 2010).

Our work is more closely related to a second class of term alignment methods that exploits the internal structure of terms between a source and a target language. Compositional translation algorithms are based on the *principal of compositionality* (Keenan and Faltz, 1985), which claims that the translation of the whole is a function of the translation of its parts. Lexical (Morin and Daille, 2010; Daille, 2012; Robitaille et al., 2006;

Tanaka, 2002) and sub-lexical (Delpech et al., 2012) compositional algorithms are knowledge-rich approaches that proceed in two steps, namely generation and selection. In the generation step, an input source term is segmented into basic translation units: words (lexical compositional methods) or morphemes (sub-lexical methods). Then a pre-compiled, seed dictionary of words or morphemes is used to translate the components of the source term. Finally, a permutation function generates candidate translations using the list of the translated segments. In the selection step, candidate translations are ranked according to their frequency (Morin and Daille, 2010; Robitaille et al., 2006) or their context similarity with the source term (Tanaka, 2002). The performance of the compositional translation algorithms is bound to the coverage of the seed dictionary (Daille, 2012). Delpech et al. (2012) noted that 30% of untranslated terms were due to the low coverage of the seed dictionary.

Kontonatsios et al. (2014) introduced a Random Forest (RF) classifier that learns correspondences of character n-grams between a source and target language. Unlike lexical and sub-lexical compositional methods, a RF classifier does not require a bilingual dictionary of translation units. The model is able to automatically build correlation paths between source and target sub-lexical segments that best discriminate translation from non-translation pairs. However, being a supervised method, it still requires a seed bilingual dictionary of technical terms for training. The RF classifier was previously applied on an English-Spanish comparable corpus and it was shown to significantly outperform context-based approaches.

3 Methods

In this section we describe the character n-gram models, the context vector method and the hybrid system. The lexicon induction task is formalised as a two-class classification problem. Given a pair of terms in a source and a target language, the output is a prediction of whether the terms are mutual translations are not. Furthermore, each term alignment method implements a ranking function that calculates a similarity score between a source and a target term. The methods rank target terms according to the similarity score and select the top N ranked terms as candidate translations. The ranking functions will be discussed in the following subsections.

3.1 Character n-gram models

Let s be a source term containing p character n-grams (s=$\{s_1, s_2, ..., s_p\}$ $s_i \in S$, $\forall i \in [1, p]$) and t a target term of q n-grams (t=$\{t_1, t_2, ..., t_q\}$ $t_i \in T$, $\forall i \in [1, q]$). We extract character n-grams by considering any contiguous, non-linguistically motivated sequence of characters that occurs within a window size of $[2-5]$ [2]) for English, French and Greek. For Japanese, unigrams are included (window size of $[1-5]$ because Japanese terms often contain Kanji (Chinese) characters.

Given the two lists of source and target n-grams, our objective is to find an underlying relationship between S and T that best discriminates translation from non-translation pairs. The RF classifier was previously shown to exhibit such behaviour (Kontonatsios et al., 2014). An RF classifier (Breiman, 2001) is a collection of decision trees voting for the most popular class. For a pair of source and target terms $\langle s, t \rangle$, the RF method creates feature vectors of a fixed size $2r$, i.e., *first order* feature space. The first r features are extracted from the source term, while the last r features from the target term. Each feature has a boolean value (0 or 1) that designates the presence/absence of the corresponding n-gram in the input instance.

The ability of the RF to detect latent associations between S and T relies on the decision trees. The internal nodes of a decision tree represent the n-gram features that are linked together in the tree-hierarchy. Each leaf node of the trees is labelled as *translation* or *non-translation* indicating whether the parent path of n-gram features is positively or negatively associated. The classification margin that we use to rank the candidate translations is given by a margin function (Breiman, 2001):

$$mg(X, Y) = av(I(x) = 1) - av(I(x)) = 0) \quad (1)$$

where x is an instance $\langle s, t \rangle$, $y \in Y = \{0, 1\}$ the class label, $I(\cdot) : (s, t) \longrightarrow \{0, 1\}$ is the indicator function of a decision tree and $av(I(\cdot))$ the average number of trees voting for the same class label. In our experiments, we used the same settings as the ones reported in Kontonatsios et al. (2014).

[2] we have experiments with larger and narrower window sizes but this setting resulted in better translation accuracy

We used 140 decision trees and $\log_2 |2q| + 1$ random features. For training an RF model, we used the WEKA platform (Hall et al., 2009).

The second class of machine learning algorithms that we investigate is Support Vector Machines (SVMs). The simplest version of SVMs is a linear classifier (linear-SVM) that tries to place a hyperplane, a decision boundary, that separates translation from non-translation instances. A linear-SVM is a feature agnostic method since the model only exploits the position of the vectors in the hyperspace to achieve class separation (Hastie et al., 2009).

The first order feature representation used with the RF classifier does not model associations between S and T. Hence, intuitively, a first order feature space is not linearly separable, i.e., there exists no decision boundary that divides the data points into translations and non-translations. [3]. To solve non-linear classification problems, SVMs employ non-linear kernels. A kernel function projects input instances into a higher dimensional space to discover non-linear associations between the initial features. In this new, projected feature space, the SVM attempts to define a separating plane. For training an non-linear SVM on the first order feature space, we used the LIBSVM package (Chang and Lin, 2011) with a *radial basis function* (RBF) kernel. For ranking candidate translations, we used the decision value given by LIBSVM which represents the distance between an instance and the hyperplane. To translate a source term, the method ranks candidate translations by decision value and suggests as best translation the candidate with the maximum distance (*maximum margin*).

While the first order models try to find cross-lingual mappings between monolingual features, our proposed method follows a different approach. It models cross-lingual links between the source and target character n-grams and uses them as *second order* features to train a linear classifier. A second order feature is a tuple of n-grams in S and T, respectively, that co-occur in a training, translation instance. Second order feature values are boolean. Given a translation instance $\langle s, t \rangle$ of p source and q target n-grams, there are $p \times q$ second order features. For dimensionality reduction, we consider as second order features the most frequent out of all possible first order feature combinations, only. Experiments indicate that a large number of features needs to be considered to achieve robust performance. To cope with the high dimensional second order space, we use LIBLINEAR (Fan et al., 2008), which is designed to solve large-scale, linear classifications problems. LIBLINEAR implements two linear classification algorithms: LogReg and linear-SVM. Both models solve the same optimisation problem, i.e., determine the optimal separating plane, but they adopt different loss functions. Since LIBLINEAR does not support decision value estimations for the linear-SVM, we only experimented with LogReg. Similarly to SVM-RBF, LogReg ranks candidate translations by classification margin.

3.2 Context vectors

We follow a standard approach to calculate context similarity of source and target terms (Rapp, 1999; Morin and Daille, 2010; Morin and Prochasson, 2011a; Delpech et al., 2012). Context vectors of candidate terms in the source and target language are populated after normalising each bilingual corpus, separately. Normalisation consists of stop-word filtering, tokenisation, lemmatisation and Part-of-Speech (PoS) tagging. For English, Spanish and French we used the TreeTagger (Schmid, 1994) while for Greek we used the ILSP toolkit (Papageorgiou et al., 2000). The Japanese corpus was segmented and PoS-tagged using Juman (Kurohashi and Kawahara, 2005).

In succession, monolingual context vectors are compiled by considering all lexical units that occur within a window of 3 words before or after a term (a seven-word window). Only lexical units (seeds) that occur in a bilingual dictionary are retained The values in context vectors are Log-Likelihood Ratio associations (Dunning, 1993) of the term and a seed lexical unit occurring in it. In a second step, we use the translations in the seed dictionary to map target context vectors into the source vector space. If there are several translations for a term, they are all considered with equal weights. Finally, candidate translations are ranked in descending order of the cosine of the angle between the mapped target vectors and the source

[3] We applied a linear-SVM with the first order feature representation on the four comparable corpora for English-French, English-Spanish, English-Greek and English-Japanese. In all cases, the best accuracies achieved were close to zero. Additionally, the ranked list of candidate translations was the same for all source terms. Hence, we can empirically suggest that the linear-SVM cannot exploit a first order feature space.

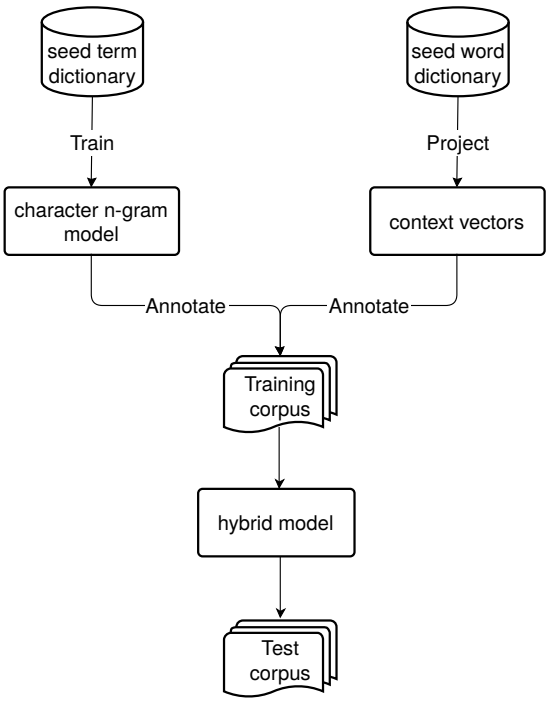


Figure 1: Architecture of the hybrid term alignment system.

vector.

3.3 Hybrid term alignment system

Figure 1 illustrates a block diagram of our term alignment system. We use two bilingual seed dictionaries: (a) a dictionary of term translation pairs to train the n-gram models and (b) a dictionary of word-to-word correspondences to translate target context vectors. The n-gram and context vector methods are used separately to score term pairs. The n-gram model computes the value of the *compositional clue* while the context vector estimates the score of the *contextual clue*. The hybrid model combines both methods by using the corresponding scores as features to train a linear classifier. For this, we used a linear-SVM of the LIBSVM package with default values for all parameters.

4 Data

Following previous research (Prochasson and Fung, 2011; Irvine and Callison-Burch, 2013; Klementiev et al., 2012), we construct comparable biomedical corpora using Wikipedia as a freely available resource.

Starting with a list of 4K biomedical English terms (query-terms), we collected 4K English Wikipedia articles, by matching query-terms to the topic signatures of articles. Then, we followed the Wikipedia interlingual links to retrieve thematically related articles in each target language. Since not all English articles contain links for all four target languages (Spanish, French, Greek and Japanese), we used a different list of query-terms for each language pair. Corpora were randomly divided into training and testing parts. For training we used 3K documents and for testing the remaining 1K. Table 1 shows the size of corpora in terms of numbers of source (SW) and target words (TW).

4.1 Seed dictionaries

As shown in Figure 1, the term alignment methods require two seed bilingual dictionaries: a term and a word dictionary. The character n-gram models rely on a bilingual term dictionary to learn associations of n-grams that appear often in technical terms. The dictionary may contain both single-word and multi-word terms. For English-Spanish and English-French we used UMLS (Bodenreider, 2004) while for English-Japanese we used an electronic dictionary of medical terms (Denshika and Kenkyukai, 1991).

An English-Greek biomedical dictionary was not available at the time of conducting these experiments, thus we automatically compiled a dictionary from a parallel corpus. For this, we trained a standard Statistical Machine Translation system (Koehn et al., 2007) on *EMEA* (Tiedemann, 2009), a biomedical parallel corpus containing sentence-aligned documents from the European Medicines Agency. Then, we extracted all English-Greek pairs for which: (a) the English sequence was listed in UMLS and (b) the translation probability was equal or higher to 0.7.

The sizes of the seed term dictionaries vary significantly, e.g., 500K entries for English-French but only 20K entries for English-Greek. However, the character n-gram models require a relatively small portion of the corresponding dictionary to converge. In the reported experiments, we used 10K translation pairs as positive, training instances. In addition, we generated an equal number of *pseudo-negative* instances by randomly matching non-translation terms.

Morin and Prochasson (2011b) showed that the translation accuracy of context vectors is higher when using bilingual dictionaries that contain both general language entries and technical terms rather than general or domain-specific dictionaries, sep-

	Training corpus		Test Corpus	
	# SW	# TW	# SW	# TW
en-fr	4.8M	2.2M	1.9M	1.1M
en-es	4.9M	2.5M	1.8M	0.9M
en-el	10.2M	2.4M	3.3M	1.3M
en-jpn	5.3M	2.4M	2.3M	1.2M

Table 1: Statistics of the English-French (en-fr), Engish-Spanish (en-es), English-Greek (en-el) and English-Japanese (en-jpn) Wikipedia comparable corpora. SW: source words, TW: target words

	Corpus Comparability	Seed words in dictionary
en-fr	0.71	66K
en-es	0.75	40K
en-el	0.68	22K
en-jpn	0.49	57K

Table 2: Corpus comparability and number of features of the seed word dictionaries

arately. In a mixed dictionary, lexical units are either single-word technical terms, such as "disease" and "patient", or general language words, such as "occur" and "high". Note that we have already compiled a seed term dictionary for each pair of languages. Following the suggestion of Morin and Prochasson (2011b), we attempt to enrich the seed term dictionaries with general language entries. For this, we extracted bilingual word dictionaries for English-Spanish, English-French and English-Greek by applying GIZA++ (Och and Ney, 2003) on the EMEA corpus. We then concatenated the word with the term dictionaries to obtain enhanced seeds for the three language pairs. For English-Japanese, we only used the term dictionary to translate the target context vectors.

Once the word dictionaries have been compiled, we compute the *corpus comparability* measure. Li and Gaussier (2010) define corpus comparability as the percentage of words that can be translated bi-directionally, given a seed dictionary.

Table 2 shows corpus comparability scores of the four corpora accompanied with the number of English, single words in the seed dictionaries. It can be observed that seed dictionary sizes are not necessarily proportional to the corresponding corpus comparability scores. As expected, for English-Japanese, corpus comparability is low because the dictionary contains single-word terms, only. The English-Spanish dictionary is smaller than the English-French but achieved higher corpus comparability, i.e., a higher percentage of words can be bi-directionally translated using the corresponding seed dictionary. A possible explanation is that the comparable corpora were constructed using different lists of query-terms. Hence, the query-terms used for English-Spanish retrieved a more coherent corpus. The resulting values of corpus comparability indicate that the context vectors will perform the best for English-Spanish while for English-Japanese the performance is expected to be substantially lower.

4.2 Training and evaluation datasets

For evaluation, we construct a test dataset of single-word terms, in particular nouns or adjectives. The dataset contains $1K$ terms that occur more frequently than 20 but not more than 200 times and are listed in the English part of the UMLS. In order to extract candidate translations, we considered all nouns or adjectives that occur at least 5 times in the target part of the corpus. Furthermore, we do not constraint the evaluation datasets only to those terms whose corresponding translation occurs in the corpus.

The hybrid model that combines the compositional and context clue, is based on a two-feature model. Therefore, the model converges using only a few hundred instances. For training a hybrid model, we used $1K$ translation instances that occurred in the training comparable corpora. Similarly, to the character n-gram models, *pseudo-negative* instances were generated by randomly coupling non-translation terms. The ratio of positive to negative instances is $1:1$.

5 Experiments

In this section, we present three experiments conducted to evaluate the *character n-gram*, *context vector* and *hybrid* methods. Firstly, we examine the performance of the n-gram models on closely related language pairs (English-French, English-Spanish), on a distant language pair (English-Greek) and on an unrelated language pair (English-Japanese). English and Greek are not unrelated because they are members of the same language family, but also not closely related because they use different scripts. Secondly,

we compare the character n-gram methods against context vectors when translating frequent or rare terms and on comparable corpora of similar language pairs (English-French, English-Spanish) but of different corpus comparability scores. Thirdly, we evaluate the hybrid method on all four comparable corpora and investigate the improvement margin of combining the *contextual* with the *compositional* clue.

As evaluation metrics, we adopt the top-N translation accuracy, following most previous approaches (Rapp, 1999; Chiao and Zweigenbaum, 2002; Morin et al., 2007; Tamura et al., 2012). The top-N translation accuracy is defined as the percentage of source terms for which a given method has output the correct translation among the top N candidate translations.

5.1 Character n-gram models

In the first experiment, we investigate the performance of the character n-gram models considering an increasing number of features. The features were sorted in order of decreasing frequency of occurrence. Starting from the top of the list, more features were incrementally added and translation accuracy was recorded.

Figure 2 shows the top-20 translation accuracy of single-word terms on an increasing number of first and second order features. With regards to the first order models (Subfigure 2a), the Random Forest (RF) classifier outperforms our baseline method (SVM-RBF) for all four language pairs. The largest margin between RF and SVM-RBF can be observed for the English-Greek dataset while for closely related language pairs, i.e., English-French and English-Spanish, the margin is smaller. Furthermore, it can be noted that using only a small number of first order features, $1K$ features (500 for the source and 500 for the target language, both n-gram models reach a stable performance.

In contrast to the first order models, the LogReg classifier requires a large number of second order features to achieve a robust performance (Subfigure 2b). Starting from $100K$ features, the translation accuracy continuously increases. The best performance is observed for a total number of $4M$ second order features when considering the English-French, English-Spanish and English-Greek datasets. For English-Japanese, the best performance is achieved for $2M$ features. Beyond this point, translation accuracy decreases slightly.

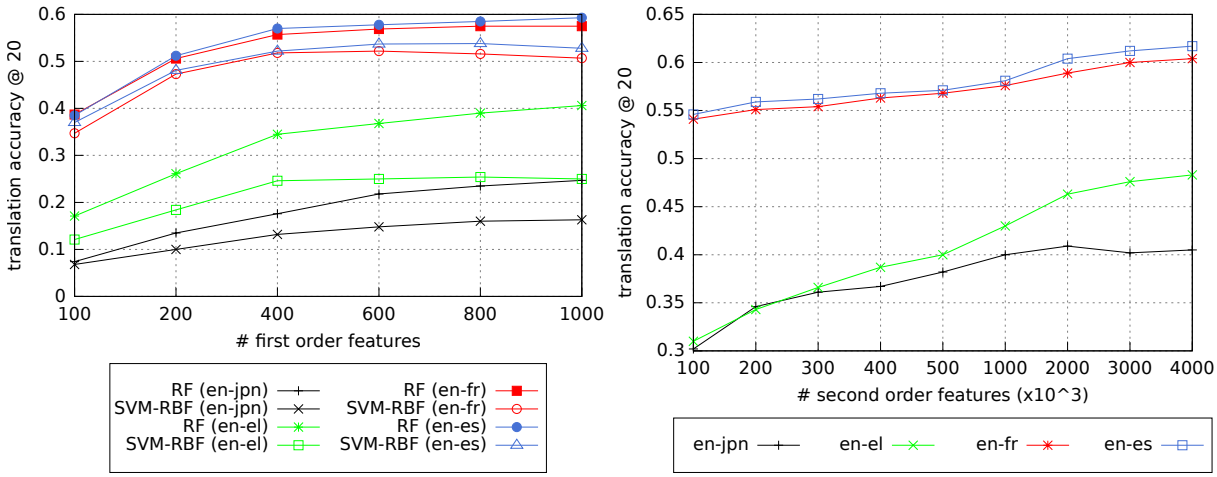
After feature selection is performed, we directly compare all the character n-gram models. Table 3 summarises performance achieved by the LogReg, RF and SVM-RBF models. It can be noted that LogReg and RF performed similarly for closely related languages (no statistically significant differences were observed) while both methods outperformed the SVM-RBF. However, for English-Greek and English-Japanese, LogReg achieved a statistically significant improvement over the translation accuracy of RF and SVM-RBF. LogReg outperformed RF by 7% for English-Greek, while for English-Japanese the improvement was 10% and 17% percent for top-1 and top-20 accuracy, respectively. Finally, it can be observed that the more distant the language pair is, the lower the performance.

5.2 N-gram methods and context vectors

In this experiment, we compare the n-gram methods against context vectors with regards to two parameters: (a) the frequency of source terms to be translated and (b) corpus comparability. English-French and English-Spanish are similar language pairs but the corresponding corpora are of different corpus comparability scores. To investigate how performance is affected by term occurrence frequency, we compiled an additional test dataset of $1K$ rare English terms in the frequency range $[10, 20]$. Our intuition is, that character n-gram methods will perform similarly for all settings since character n-grams are corpus independent features.

We compare (a) the character n-gram models (LogReg, RF and SVM-RBF) with (b) the context vector method (context) and (c) an upper bound. The latter represents the percentage of source terms for which a reference translation actually occurs in the target corpus. Hence, the upper bound is the maximum performance achievable according to the reference evaluation.

Figure 3a shows the top-20 translation accuracy for high and medium frequency terms, within the frequency range $[20, 200]$. Context vectors achieved a robust performance of 52% and 45% for English-Spanish and English-French, respectively. The difference in corpus comparability can explain this 7% margin between these performances. As shown in Table 2, the corpus comparability scores for English-Spanish and English-



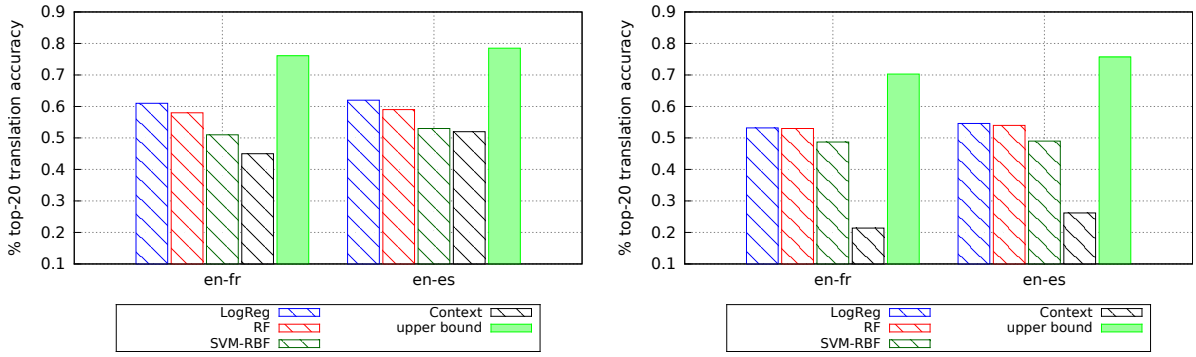
(a) First order n-gram models

(b) Second order n-gram model

Figure 2: Top-20 translation accuracy of models trained on (a) first and (b) second order features

	English-French		English-Spanish		English-Greek		English-Japanese	
	acc@1	acc@20	acc@1	acc@20	acc@1	acc@20	acc@1	acc@20
LogReg	0.45	**0.61**	0.42	**0.62**	**0.3**	**0.48**	**0.25**	**0.41**
RF	**0.47**	0.58	**0.43**	0.59	0.23	0.41	0.15	0.24
SVM-RBF	0.38	0.51	0.33	0.53	0.1	0.25	0.06	0.16

Table 3: Top-1 (acc@1) and top-20 (acc@20) translation accuracy of LogReg, RF and SVM-RBF



(a) Test terms with frequency $[20, 200]$

(b) Test terms with frequency $[10, 20]$

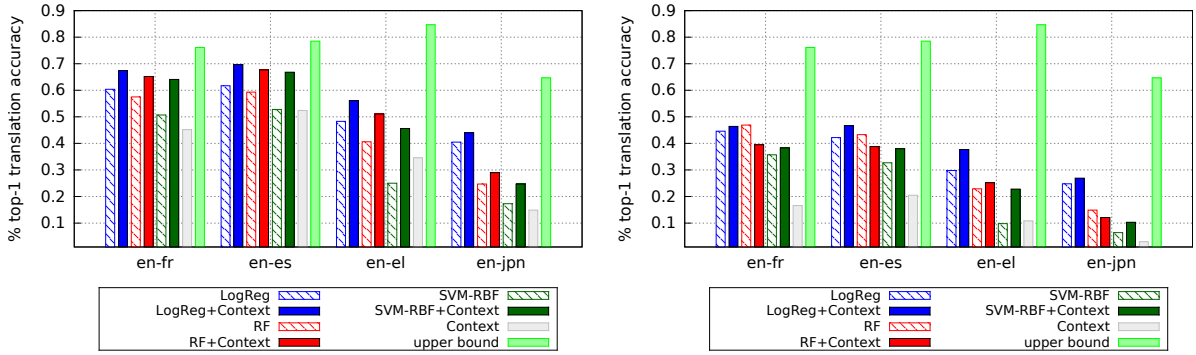
Figure 3: Top-20 translation accuracy of terms in the frequency range of $[10, 200]$ and $[10, 20]$

French are 0.75 and 0.71, respectively. In contrast to context vectors, the character n-gram methods performed comparably.

A second factor that affects the performance of context vectors, is the frequency of the terms to be translated. The translation of rare terms has been shown to be a challenging case for context vectors. For example, Morin and Daille (2010) reported low accuracy (21% for the top-20 candidates) of context vectors for terms occurring 20 times or less. In our experiments, Figure 3b illustrates accuracies achieved for less frequent terms ($[10, 20]$). The performance of context vectors is significantly lower, 26% for English-Spanish and 21% for English-French. Furthermore, the translation accuracy of the n-gram methods decreases slightly ($\sim$ 5% to 8%). This can be explained by the decrease of the upper bound for lower frequency terms ($\sim$ 3% to 6%).

5.3 Combining internal and contextual similarity

We have hypothesised that the *compositional* and *contextual* clue are orthogonal, i.e., they convey



(a) Top-20 accuracy (acc@20)

(b) Top-1 accuracy (acc@1)

Figure 4: Overall performance. Top-20 and top-1 translation accuracy

different and possibly complimentary information. To investigate this intuition, we evaluate the hybrid model on all four comparable corpora, for term occurrence frequencies in $[20, 200]$.

Figure 4a illustrates top-20 translation accuracy scores for (a) the character n-gram models, (b) the context vector method and (c) the hybrid models, i.e., LogReg+Context, RF+Context, SVM-RBF+Context. We observe that the combination of the *compositional* and *contextual* clue improved the performance of all methods. The hybrid model largely improved the performance of the SVM-RBF ($\sim$ 14% to 20%). With regards to the combined signals the translation accuracy of LogReg and RF increased by $\sim$ 4% for the English-Japanese corpus and $\sim$ 8% for all other corpora.

For the top 1 candidate translation, we observe in Figure 4 smaller improvements achieved by the hybrid model in comparison to the top-20 accuracy. Interestingly, the RF classifier performed slightly better on its own for English-French, English-Spanish and English-Japanese. This indicates that the hybrid method ranks more correct translations in the top 20 candidates but it does not always assign the best score to the correct answer.

6 Discusion and Future work

In this paper, we investigated a compositional and a context-based approach useful for compiling bilingual dictionaries of terms automatically from comparable corpora. Compositional translation methods exploit the internal structure of terms across languages while context-based approaches investigate the surrounding lexical context.

We proposed a character n-gram compositional method, i.e., a *Logistic Regression* classifier, which uses a multilingual representation, i.e., source and target terms. Experimental evidence showed that the LogReg classifier significantly outperformed the baseline methods on distant languages. For closely related languages, LogReg performed comparably to an existing n-gram method based on a *Random Forest* classifier.

Furthermore, we compared the n-gram models against a context-based approach under different corpus-specific parameters: (a) corpus comparability, which is relevant to the seed dictionary, and (b) the occurrence frequency of the terms to be translated. It was shown that the performance of n-gram methods was not affected by different parameter settings. Only small fluctuations were observed, since the n-gram methods are based on corpus-independent features, only. In contrast, the context-based method was affected by corpus comparability scores. The corresponding translation accuracy declined significantly for rare terms.

Finally, we hypothesised that the n-gram and context-based methods provide complimentary information. To test this hypothesis, we developed a hybrid method that combines compositional and contextual similarity scores as features in a linear classifier. The hybrid model achieved significantly better top-20 translation accuracy than the two methods separately but minor improvements were observed in terms of top-1 accuracy.

As future work, we plan to improve the quality of the extracted dictionary further by exploiting additional translation signals. For example, previous works (Schafer and Yarowsky, 2002; Klementiev et al., 2012) have reported that the *temporal* and *topic* similarity are clues that indicate translation equivalence. It would be interesting to investigate the contribution of different clues for various

experimental parameters, e.g., domain, distance of languages, types of comparable corpora.

Acknowledgements

The authors would like to thank Dr. Danushka Bollegala for providing feedback on this paper and the three anonymous reviewers for their useful comments and suggestions. This work was funded by the European Community's Seventh Framework Program (FP7/2007-2013) [grant number 318736 (OSSMETER)].

References

Daniel Andrade, Tetsuya Nasukawa, and Jun'ichi Tsujii. 2010. Robust measurement and comparison of context similarity for finding translation pairs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Marianna Apidianaki, Nikola Ljubešic, and Darja Fišer. 2012. Disambiguating vectors for bilingual lexicon extraction from comparable corpora. In *Eighth Language Technologies Conference*, pages 10–15.

Lisa Ballesteros and W.Bruce Croft. 1997. Phrasal translation and query expansion techniques for cross-language information retrieval. In *ACM SIGIR Forum*, volume 31, pages 84–91. ACM.

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl 1):D267–D270.

Leo Breiman. 2001. Random Forests. *Machine Learning*, 45:5–32.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Yun-Chuang Chiao and Pierre Zweigenbaum. 2002. Looking for Candidate Translational Equivalents in Specialized, Comparable Corpora. In *International Conference on Computational Linguistics*.

Ido Dagan and Ken Church. 1994. Termight: Identifying and translating technical terminology. In *Proceedings of the fourth conference on Applied natural language processing*, pages 34–40. Association for Computational Linguistics.

Emmanuel Morin Bīeatrice Daille. 2012. Revising the compositional method for terminology acquisition from comparable corpora. *COLING 2012*, 1810.

Estelle Delpech, Béatrice Daille, Emmanuel Morin, and Claire Lemaire. 2012. Extraction of domain-specific bilingual lexicon from comparable corpora: Compositional translation and ranking. In *COLING*, pages 745–762.

Igakuyo Denshika and Jisho Kenkyukai. 1991. 250,000 medical term dictionary (in japanese). Nichigai Associates, Inc.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19(1):61–74.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Pascale Fung and Kathleen McKeown. 1997. A technical word-and term-translation aid using noisy parallel corpora across language groups. *Machine Translation*, 12(1-2):53–87.

Pascale Fung and Lo Yuen Yee. 1998. An ir approach for translating new words from nonparallel, comparable texts. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 414–420. Association for Computational Linguistics.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. 2009. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

Z.S. Harris. 1954. Distributional structure. *Word*.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. 2009. *The elements of statistical learning*, volume 2. Springer.

Ann Irvine and Chris Callison-Burch. 2013. Supervised bilingual lexicon induction with multiple monolingual signals. In *Proceedings of NAACL-HLT*, pages 518–523.

Edward L Keenan and Leonard M Faltz. 1985. *Boolean semantics for natural language*, volume 23. Springer.

Alexandre Klementiev, Ann Irvine, Chris Callison-Burch, and David Yarowsky. 2012. Toward statistical machine translation without parallel corpora. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 130–140. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*,

pages 177–180. Association for Computational Linguistics.

G. Kontonatsios, I. Korkontzelos, J. Tsujii, and S. Ananiadou. 2014. Using a random forest classifier to compile bilingual dictionaries of technical terms from comparable corpora. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 111–116. Association for Computational Linguistics.

Sadao Kurohashi and Daisuke Kawahara. 2005. Japanese morphological analysis system juman version 5.1 manual.

Bo Li and Eric Gaussier. 2010. Improving corpus comparability for bilingual lexicon extraction from comparable corpora. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 644–652. Association for Computational Linguistics.

Christian Lovis, R Baud, PA Michel, JR Scherrer, and AM Rassinoux. 1997. Building medical dictionaries for patient encoding systems: A methodology. In *Artificial Intelligence in Medicine*, pages 373–380. Springer.

Emmanuel Morin and Béatrice Daille. 2010. Compositionality and lexical alignment of multi-word terms. *Language Resources and Evaluation*, 44(1-2):79–95.

Emmanuel Morin and Emmanuel Prochasson. 2011a. Bilingual lexicon extraction from comparable corpora enhanced with parallel corpora. In *Proceedings of the 4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web*, pages 27–34. Association for Computational Linguistics.

Emmanuel Morin and Emmanuel Prochasson. 2011b. Bilingual lexicon extraction from comparable corpora enhanced with parallel corpora. In *Proceedings of the 4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web*, pages 27–34, Portland, Oregon, June. Association for Computational Linguistics.

Emmanuel Morin, Béatrice Daille, Koichi Takeuchi, and Kyo Kageura. 2007. Bilingual terminology mining - using brain, not brawn comparable corpora. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 664–671, Prague, Czech Republic, June. Association for Computational Linguistics.

Fiammetta Namer and Robert Baud. 2007. Defining and relating biomedical terms: towards a cross-language morphosemantics-based system. *International Journal of Medical Informatics*, 76(2):226–233.

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

Harris Papageorgiou, Prokopis Prokopidis, Voula Giouli, and Stelios Piperidis. 2000. A unified pos tagging architecture and its application to greek. In *Proceedings of the 2nd Language Resources and Evaluation Conference*, pages 1455–1462, Athens, June. European Language Resources Association.

Emmanuel Prochasson and Pascale Fung. 2011. Rare word translation extraction from aligned comparable documents. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1327–1335. Association for Computational Linguistics.

James Pustejovsky, Jose Castano, Brent Cochran, Maciej Kotecki, and Michael Morrell. 2001. Automatic extraction of acronym-meaning pairs from medline databases. *Studies in health technology and informatics*, (1):371–375.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 519–526. Association for Computational Linguistics.

Xavier Robitaille, Yasuhiro Sasaki, Masatsugu Tonoike, Satoshi Sato, and Takehito Utsuro. 2006. Compiling french-japanese terminologies from the web. In *EACL*.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–7. Association for Computational Linguistics.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, volume 12, pages 44–49. Manchester, UK.

Frank Smadja, Kathleen R McKeown, and Vasileios Hatzivassiloglou. 1996. Translating collocations for bilingual lexicons: A statistical approach. *Computational linguistics*, 22(1):1–38.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2012. Bilingual lexicon extraction from comparable corpora using label propagation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 24–36. Association for Computational Linguistics.

Takaaki Tanaka. 2002. Measuring the similarity between compound nouns in different languages using non-parallel corpora. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Jörg Tiedemann. 2009. News from opus-a collection of multilingual parallel corpora with tools and interfaces. In *Recent Advances in Natural Language Processing*, volume 5, pages 237–248.

Pim Van der Eijk. 1993. Automating the acquisition of bilingual terminology. In *Proceedings of the sixth conference on European chapter of the Association for Computational Linguistics*, pages 113–119. Association for Computational Linguistics.

Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1263–1271, Stroudsburg, PA, USA. Association for Computational Linguistics.

Random Manhattan Integer Indexing: Incremental L_1 Normed Vector Space Construction

Behrang Q. Zadeh[†]
[†] Insight Centre
National University of Ireland, Galway
Galway, Ireland
behrang.qasemizadeh@insight-centre.org

Siegfried Handschuh[†‡]
[‡] Dept. of Computer Science and Mathematics
University of Passau
Bavaria, Germany
siegfried.handschuh@uni-passau.de

Abstract

Vector space models (VSMs) are mathematically well-defined frameworks that have been widely used in the distributional approaches to semantics. In VSMs, high-dimensional vectors represent linguistic entities. In an application, the similarity of vectors—and thus the entities that they represent—is computed by a distance formula. The high dimensionality of vectors, however, is a barrier to the performance of methods that employ VSMs. Consequently, a dimensionality reduction technique is employed to alleviate this problem. This paper introduces a novel technique called Random Manhattan Indexing (RMI) for the construction of ℓ_1 normed VSMs at reduced dimensionality. RMI combines the construction of a VSM and dimension reduction into an incremental and thus scalable two-step procedure. In order to attain its goal, RMI employs the sparse Cauchy random projections. We further introduce Random Manhattan Integer Indexing (RMII): a computationally enhanced version of RMI. As shown in the reported experiments, RMI and RMII can be used reliably to estimate the ℓ_1 distances between vectors in a vector space of low dimensionality.

1 Introduction

Distributional semantics embraces a set of methods that decipher the meaning of linguistic entities using their usages in large corpora (Lenci, 2008). In these methods, the distributional properties of linguistic entities in various contexts, which are collected from their observations in corpora, are compared to quantify their meaning. Vector spaces are intuitive, mathematically well-defined frameworks to represent and process such information.[1] In a vector space model (VSM), linguistic entities are represented by vectors and a distance formula is employed to measure their distributional similarities (Turney and Pantel, 2010).

In a VSM, each element $\vec{s}_i$ of the standard basis of the vector space (informally, each dimension of the VSM) represents a context element. Given n context elements, an entity whose meaning is being analyzed is expressed by a vector $\vec{v}$ as a linear combination of $\vec{s}_i$ and scalars $\alpha_i \in \mathbb{R}$ such that $\vec{v} = \alpha_1\vec{s}_1 + \cdots + \alpha_n\vec{s}_n$. The value of α_i is derived from the frequency of the occurrences of the entity that $\vec{v}$ represents in/with the context element that $\vec{s}_i$ represents. As a result, the values assigned to the coordinates of a vector (i.e. α_i) exhibit the correlation of entities and context elements in an n-dimensional real vector space $\mathbb{R}^n$. Each vector can be written as a $1 \times n$ row matrix, e.g. $(\alpha_1, \cdots, \alpha_n)$. Therefore, a group of m vectors in a vector space is often represented by a matrix $\mathbf{M}_{m\times n}$.

Latent semantic analysis (LSA) is a familiar technique that employs a *word-by-document* VSM (Deerwester et al., 1990).[2] In this word-by-document model, the meaning of words (i.e. the linguistic entities) is described by their occurrences in documents (i.e. the context elements). Given m words and n distinct documents, each word is represented by an n-dimensional vector $\vec{v_i} = (\alpha_{i1}, \cdots, \alpha_{in})$, where α_{ij} is a numeric value that associates the word $\vec{v_i}$ represents to the document d_j, for $1 < j < n$. For instance, the value of α_{ij} may correspond to the frequency of the word in the document. It is hypothesized that the relevance of words can be assessed by counting the documents in which they co-occur. Therefore, words with similar vectors are assumed to have the same meaning (Figure 1).

[1] Amongst other representation frameworks.

[2] See Martin and Berry (2007) for an overview of the mathematical foundation of LSA.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1713–1723,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

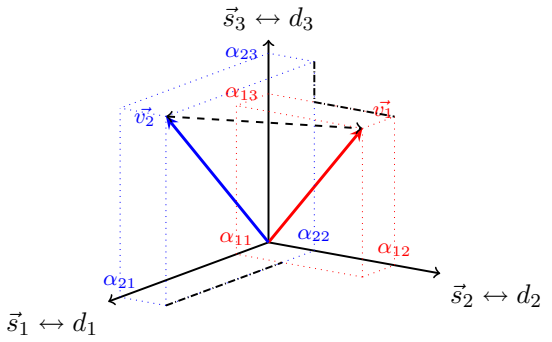


Figure 1: Illustration of a *word-by-document* model consisting of 2 words and 3 documents. The words are represented in a 3-dimensional vector space, in which each $\vec{s_i}$ (each dimension) represents each of the 3 documents in the model. $\vec{v_1} = (\alpha_{11}, \alpha_{12}, \alpha_{13})$ and $\vec{v_2} = (\alpha_{21}, \alpha_{22}, \alpha_{23})$ represent the two words in the model. The dashed line shows the Euclidean distance between the two vectors that represent words, while the sum of dash-dotted lines is the Manhattan distance between them.

In order to assess the similarity between vectors, a vector space V is endowed with a *norm* structure. A norm $\|.\|$ is a function that maps vectors from V to the set of non-negative real numbers, i.e. $V \mapsto [0, \infty)$. The pair of $(V, \|.\|)$ is then called a *normed* space. In a normed space, the similarity between vectors is assessed by their distances. The distance between vectors is defined by a function that satisfies certain axioms and assigns a real value to each pair of vectors, i.e.

$$dist : V \times V \mapsto \mathbb{R}, \quad d(\vec{v}, \vec{t}) = \|\vec{v} - \vec{u}\|. \quad (1)$$

The smaller the distance between two vectors, the more similar they are.

Euclidean space is the most familiar example of a normed space. It is a vector space that is endowed by the ℓ_2 norm. In Euclidean space, the ℓ_2 norm—which is also called the Euclidean norm—of a vector $\vec{v} = (v_1, \cdots, v_n)$ is defined as

$$\|\vec{v}\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}. \quad (2)$$

Using the definition of distance given in Equation 1 and the ℓ_2 norm, the Euclidean distance is measured as

$$dist_2(\vec{v}, \vec{u}) = \|\vec{v} - \vec{u}\|_2 = \sqrt{\sum_{i=1}^{n} (v_i - u_i)^2}. \quad (3)$$

In Figure 1, the dashed line shows the Euclidean distance between the two vectors. In ℓ_2 normed vector spaces, various similarity metrics are defined using different normalization of the Euclidean distance between vectors, e.g. the *cosine similarity*.

The similarity between vectors, however, can also be computed in ℓ_1 normed spaces.[3] The ℓ_1 norm for $\vec{v}$ is given by

$$\|\vec{v}\|_1 = \sum_{i=1}^{n} |v_i|, \quad (4)$$

where $|.|$ signifies the modulus. The distance in an ℓ_1 normed vector space is often called the *Manhattan* or the *city block* distance. According to the definition given in Equation 1, the Manhattan distance between two vectors $\vec{v}$ and $\vec{u}$ is given by

$$dist_1(\vec{v}, \vec{u}) = \|\vec{v} - \vec{u}\|_1 = \sum_{k=1}^{n} |v_i - u_j|. \quad (5)$$

In Figure 1, the collection of the dash-dotted lines is the ℓ_1 distance between the two vectors. Similar to the ℓ_2 spaces, various normalizations of the ℓ_1 distance[4] define a family of ℓ_1 normed similarity metrics.

As the number of text units that are being modelled in a VSM increases, the number of context elements that are required to be utilized to capture their meaning escalates. This phenomenon is explained using power-law distributions of text units in context elements (e.g. the familiar Zipfian distribution of words). As a result, extremely high-dimensional vectors, which are also sparse—i.e. most of the elements of the vectors are zero—represent text units. The high dimensionality of the vectors results in setbacks, which are colloquially known as *the curse of dimensionality*. For instance, in a word-by-document model that consists of a large number of documents, a word appears only in a few documents, and the rest of the documents are irrelevant to the meaning of the word. Few common documents between words results in sparsity of the vectors; and the presence of irrelevant documents introduces noise.

Dimension reduction, which usually follows the construction of a VSM, alleviates the problems

[3] The definition of the norm is generalized to ℓ_p spaces with $\|\vec{v}\|_p = \left(\sum_i |v_i|^p\right)^{1/p}$, which is beyond the scope of this paper.

[4] As long as the axioms in the distance definition hold.

listed above by reducing the number of context elements that are employed for the construction of the VSM. In its simple form, dimensionality reduction can be performed using a *selection process*: choose a subset of contexts and eliminate the rest using a heuristic. Alternatively, *transformation* methods can be employed. A transformation method maps a vector space V_n onto a V_m of lowered dimension, i.e. $\tau : V_n \mapsto V_m, m \ll n$. The vector space at reduced dimension, i.e. V_m, is often the best approximation of the original V_n in a *sense*. LSA employs a dimension reduction technique called truncated singular value decomposition (SVD). In a standard truncated SVD, the transformation guarantees the least distortion in the ℓ_2 distances.[5]

Besides the problem of high computational complexity of SVD computation,[6] which can be addressed by incremental techniques (see e.g. Brand (2006)), matrix factorization methods such as truncated SVD are *data-sensitive*: if the structure of the data being analyzed changes, i.e. when either the linguistic entities or context elements are updated, e.g. some are removed or new ones are added, the transformation should be recomputed and reapplied to the whole VSM to reflect the updates. In addition, a VSM at the original high dimension must be first constructed. Following the construction of the VSM, the dimension of the VSM is reduced in an independent process. Therefore, the VSM at reduced dimension is available for processing only after the whole sequence of these processes. Construction of the VSM at its original dimension is computationally expensive and a delay in access to the VSM at reduced dimension is not desirable. Hence, the application of truncated SVD is not suitable in several applications, particularly when dealing with frequently updated big text–data such as applications in the web context.

Random indexing (RI) is an alternative method that solves the problems stated above by combining the construction of a vector space and the dimensionality reduction process. RI, which is introduced in Kanerva et al. (2000), constructs a VSM directly at reduced dimension. Unlike methods that first construct a VSM at its original high dimension and conduct a dimensionality reduction afterwards, the RI method avoids the construction of the original high-dimensional VSM. Instead, it merges the vector space construction and the dimensionality reduction process. RI, thus, significantly enhances the computational complexity of deriving a VSM from text. However, the application of the RI technique (likewise the standard truncated SVD in LSA) is limited to ℓ_2 normed spaces, i.e. when similarities are assessed using a measure based on the ℓ_2 distance. It can be verified that using RI causes large distortions in the ℓ_1 distances between vectors (Brinkman and Charikar, 2005). Hence, if the similarities are computed using the ℓ_1 distance, then the RI technique is not suitable for the VSM construction.

Depending on the distribution of vectors in a VSM, the performance of similarity measures based on the ℓ_1 and the ℓ_2 norms varies from one task to another. For instance, it is known that the ℓ_1 distance is more robust to the presence of outliers and non-Gaussian noise than the ℓ_2 distance (e.g. see the problem description in Ke and Kanade (2003)). Hence, the ℓ_1 distance can be more reliable than the ℓ_2 distance in certain applications. For instance, Weeds et al. (2005) suggest that the ℓ_1 distance outperforms other similarity metrics in a term classification task. In another experiment, Lee (1999) observed that the ℓ_1 distance gives more desirable results than the Cosine and the ℓ_2 measures.

In this paper, we introduce a novel method called *Random Manhattan Indexing* (RMI). RMI constructs a vector space model directly at reduced dimension while it preserves the pairwise ℓ_1 distances between vectors in the original high-dimensional VSM. We then introduced a computationally enhanced version of RMI called *Random Manhattan Integer Indexing* (RMII). RMI and RMII, similar to RI, merge the construction of a VSM and dimension reduction into an incremental and thus efficient and scalable process.

In Section 2, we explain and evaluate the RMI method. In Section 3, the RMII method is explained. We compare the proposed method with RI in Section 4. We conclude in Section 5.

2 Random Manhattan Indexing

We propose the RMI method: a novel technique that adapts an incremental procedure for the construction of ℓ_1 normed vector spaces at a reduced dimension. The RMI method employs a two-step

[5] Please note that there are matrix factorization techniques that guarantee the least distortion in the ℓ_1 distances, see e.g. Kwak (2008).

[6] Matrix factorization techniques, in general.

procedure: (a) the creation of *index vectors* and (b) the construction of *context vectors* .

In the first step, each context element is assigned exactly to one *index vector* $\vec{r_i}$. Index vectors are high-dimensional and generated randomly such that entries r_j of index vectors have the following distribution:

$$r_i = \begin{cases} \frac{-1}{U_1} & \text{with probability } \frac{s}{2} \\ 0 & \text{with probability } 1-s \\ \frac{1}{U_2} & \text{with probability } \frac{s}{2} \end{cases}, \quad (6)$$

where U_1 and U_2 are independent uniform random variables in $(0, 1)$. In the second step, each target linguistic entity that is being analyzed in the model is assigned to a context vector $\vec{v_c}$ in which all the elements are initially set to 0. For each encountered occurrence of a linguistic entity and a context element—e.g. through a sequential scan of an input text collection—$\vec{v_c}$ that represents the linguistic entity is accumulated by the index vector $\vec{r_i}$ that represents the context element, i.e. $\vec{v_c} = \vec{v_c} + \vec{r_i}$. This process results in a VSM of a reduced dimensionality that can be used to estimate the ℓ_1 distances between linguistic entities. In the constructed VSM by RMI, the ℓ_1 distance between vectors is given by the *sample median* (Indyk, 2000). For given vectors $\vec{v}$ and $\vec{u}$, the approximate ℓ_1 distance between vectors is estimated by

$$\hat{L}_1(\vec{u}, \vec{v}) = \text{median}\{|v_i - u_i|, i = 1, \cdots, m\}, \quad (7)$$

where m is the dimension of the VSM constructed by RMI, and $|.|$ denotes the modulus.

RMI is based on the random projection (RP) technique for dimensionality reduction. In RP, a high-dimensional vector space is mapped onto a random subspace of lowered dimension expecting that—with a high probability—relative distances between vectors are approximately preserved. Using the matrix notation, this projection is given by

$$\mathbf{M}'_{p\times m} = \mathbf{M}_{p\times n}\mathbf{R}_{n\times m}, \quad m \ll p, n, \quad (8)$$

where $\mathbf{R}$ is often called the *random projection matrix*, and $\mathbf{M}$ and $\mathbf{M}'$ denote p vectors in the original n-dimensional and reduced m-dimensional vector spaces, respectively.

In RMI, the stated mapping in Equation 8 is given by *Cauchy random projections*. Indyk (2000) suggests that vectors in a high-dimensional space $\mathbb{R}^n$ can be mapped onto a vector space of lowered dimension $\mathbb{R}^m$ while the relative pairwise ℓ_1 distances between vectors are preserved with a high probability. In Indyk (2000, Theorem 3) and Indyk (2006, Theorem 5), it is shown that for an $m \geq m_0 = \log(1/\delta)^{O(1/\epsilon)}$, where $\delta > 0$ and $\epsilon \leq 1/2$, there exists a mapping from $\mathbb{R}^n$ onto $\mathbb{R}^m$ that guarantees the ℓ_1 distances between any pair of vectors $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$ after the mapping does not increase by a factor more than $1+\epsilon$ with constant probability δ, and it does not decrease by more than $1-\epsilon$ with probability $1-\delta$.

In Indyk (2000), this projection is proved to be obtained using a random projection matrix $\mathbf{R}$ that has *Cauchy distribution*—i.e. for r_{ij} in $\mathbf{R}$, $r_{ij} \sim C(0, 1)$. Since $\mathbf{R}$ has a Cauchy distribution, for every two vectors $\vec{u}$ and $\vec{v}$ in the high-dimensional space $\mathbb{R}^n$, the projected differences $x = \hat{\vec{u}} - \hat{\vec{v}}$ also have Cauchy distribution, with the scale parameter being the ℓ_1 distances, i.e. $x \sim C(0, \sum_{i=1}^{n} |u_i - v_i|)$. As a result, in Cauchy random projections, estimating the ℓ_1 distances boils down to the estimation of the Cauchy scale parameter from independent and identically distributed (i.i.d.) samples x. Because the expectation value of x is infinite,[7] the sample mean cannot be employed to estimate the Cauchy scale parameter. Instead, using the 1-stability of Cauchy distribution, Indyk (2000) proves that the median can be employed to estimate the Cauchy scale parameter, and thus the ℓ_1 distances at the projected space $\mathbb{R}^m$.

Subsequent studies simplified the method proposed by Indyk (2000). Li (2007) shows that $\mathbf{R}$ with Cauchy distribution can be substituted by a *sparse* $\mathbf{R}$ that has a mixture of symmetric 1-Pareto distribution. A 1-Pareto distribution can be sampled by $1/U$, where U is an independent uniform random variable in $(0, 1)$. This results in a random matrix $\mathbf{R}$ that has the same distribution as described by Equation 6.

The RMI's two-step procedure is explained using the basic properties of matrix arithmetic and the descriptions given above. Given the projection in Equation 8, the first step of RMI refers to the construction of $\mathbf{R}$: index vectors are the row vectors of $\mathbf{R}$. The second step of the process refers to the construction of $\mathbf{M}'$: context vectors are the row vectors of $\mathbf{M}'$. Using the distributive property of multiplication over addition in matrices,[8]

[7] That is $E(x) = \infty$, since x has a Cauchy distribution.

[8] That is, $(\mathbf{A} + \mathbf{B})\mathbf{C} = \mathbf{AC} + \mathbf{BC}$.

it can be verified that the explicit construction of $\mathbf{M}$ and its multiplication to $\mathbf{R}$ can be substituted by a number of summation operations. $\mathbf{M}$ can be represented by the sum of unit vectors in which a unit vector corresponds to the co-occurrence of a linguistic entity and a context element. The result of the multiplication of each unit vector and $\mathbf{R}$ is the row vector that represents the context element in $\mathbf{R}$—i.e. the index vector. Therefore, $\mathbf{M}'$ can be computed by the accumulation of the row vectors of $\mathbf{R}$ that represent encountered context elements, as stated in the second step of the RMI procedure.

2.1 Alternative Distance Estimators

As stated above, Indyk (2000) suggests using the sample median for the estimation of the ℓ_1 distances. However, Li (2008) argues that sample median estimator can be biased and inaccurate, specifically if m—i.e. the targeted (reduced) dimensionality—is small. Hence, Li (2008) suggests using the geometric mean estimator instead of the median sample:[9]

$$\hat{L}_1(\vec{u},\vec{v}) = \Big(\prod_{i=1}^{m} |u_i - v_i|\Big)^{\frac{1}{m}}. \qquad (9)$$

We suggest computing the $\hat{L}_1(\vec{u},\vec{v})$ in Equation 9 using arithmetic mean of logarithm-transformed values of $|u_i - v_i|$. Therefore, using the logarithmic identities, the multiplications and the power in Equation 9 are, respectively, transformed to a sum and a multiplication:

$$\hat{L}_1(\vec{u},\vec{v}) = \exp\Big(\frac{1}{m}\sum_{i=1}^{m} \ln(|u_i - v_i|)\Big). \qquad (10)$$

Equation 10 for computing $\hat{L}_1$ is more plausible for computational implementation than Equation 9 (e.g. the overflow is less likely to happen during the process). Moreover, calculating the median involves sorting an array of real numbers. Thus, computation of the geometric mean in logarithmic scales can be faster than computation of the median sample, especially when the value of m is large.

2.2 RMI's Parameters

In order to employ the RMI method for the construction of a VSM at reduced dimension and the estimation of the ℓ_1 distance between vectors, two model parameters should be decided: (a) the targeted (reduced) dimensionality of the VSM, which is indicated by m in Equation 8 and (b) the number of non-zero elements in index vectors, which is determined by s in Equation 6. In contrast to the classic *one-dimension-per-context-element* methods of VSM construction,[10] the value of m in RPs and thus in RMI is chosen independently of the number of context elements in the model (n in Equation 8).

In RMI, m determines the probability and the maximum expected amount of distortions ϵ in the pairwise distance between vectors. Based on the proposed refinements of Indyk (2000, Theorem 3) by Li et al. (2007), it is verified that the pairwise ℓ_1 distance between any p vectors is approximated within a factor $1 \pm \epsilon$, if $m = \mathrm{O}(\log p/\epsilon^2)$, with a constant probability. Therefore, the value of ϵ in RMI is subject to the number of vectors p in the model. For a fixed p, a larger m yields to lower bounds on the distortion with a higher probability. Because a small m is desirable from the computational complexity outlook, the choice of m is often a trade-off between accuracy and efficiency. According to our experiment, $m > 400$ is suitable for most applications.

The number of non-zero elements in index vectors, however, is decided by the number of context elements n and the sparseness of the VSM β at its original dimension. Li (2007) suggests $\frac{1}{\mathrm{O}(\sqrt{\beta n})}$ as the value of s in Equation 6. VSMs employed in distributional semantics are highly sparse. The sparsity of a VSM in its original dimension β is often considered to be around 0.0001–0.01. As the original dimension of VSM n is very large—otherwise there would be no need for dimensionality reduction—the index vectors are often very sparse. Similar to m, larger s produces smaller errors; however, it imposes more processes during the construction of a VSM.

2.3 Experimental Evaluation of RMI

We report the performance of the RMI method with respect to its ability to preserve the relative ℓ_1 distance between linguistic entities in a VSM. Therefore, instead of a task-specific evaluation, we show that the relative ℓ_1 distance between a set of words in a high-dimensional *word-by-document* model remains intact when the model

[9] See also Li et al. (2007, Lemma 5–9).

[10] That is, n context elements are modelled in an n-dimensional VSM.

is constructed at reduced dimensionality using the RMI technique. We further explore the effect of the RMI's parameter setting in the observed results.

Depending on the structure of the data that is being analyzed and the objective of the task in hand, the performance of the ℓ_1 distance for similarity measurement varies from one application to another.[11] The purpose of our reported evaluation, thus, is not to show the superiority of the ℓ_1 distance (thus RMI) to another similarity measure (e.g. the ℓ_2 distance or the cosine similarity) and employed techniques for dimensionality reduction (e.g. RI or truncated SVD) in a specific task. If, in a task, the ℓ_1 distance shows higher performance than the ℓ_2 distance, then the RMI technique is preferable to the RI technique or truncated SVD. Contrariwise, if the ℓ_2 norm shows higher performance than the ℓ_1, then RI or truncated SVD are more desirable than the RMI method.

In our experiment, a word-by-document model is first constructed from the UKWaC corpus at its original high dimension. UKWaC is a freely available corpus of 2,692,692 web documents, nearly 2 billion tokens and 4 million types (Baroni et al., 2009).[12] Therefore, a word-by-document model constructed from this corpus using the classic one-dimension-per-context-element method has a dimension of 2.69 million. In order to keep the experiments computationally tractable, the reported results are limited to 31 words from this model, which are listed in Table 1.

In the designed experiment, a word from the list is taken as the reference and its ℓ_1 distance to the remaining 30 words is calculated using the vector representations in the high-dimensional VSM. These 30 words are then sorted in ascending order by the calculated ℓ_1 distance. The procedure is repeated for all the 31 words in the list, one by one. Therefore, the procedure results in 31 sorted lists, each containing 30 words. Figure 2 shows an example of the obtained sorted list, in which the reference is the word 'research'.[13]

The procedure described above is replicated to obtain the lists of sorted words from VSMs that are constructed by the RMI method at reduced

PoS	Words			
Noun	website	email	support	software
	students	skills	project	research
	nhs	link	services	organisations
Adj	online	digital	mobile	sustainable
	global	unique	excellent	disabled
	new	current	fantastic	innovative
Verb	use	visit	improve	provided
	help	ensure	develop	

Table 1: Words employed in the experiments.

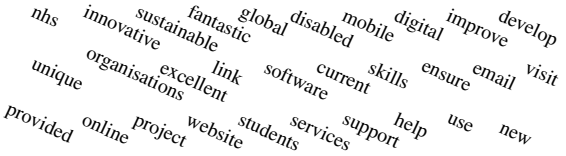


Figure 2: List of words sorted by their ℓ_1 distance to the word 'research'. The distance increases from left to right and top to bottom.

dimensionality, when the method's parameters—i.e. the dimensionality of VSM and the number of non-zero elements in index vectors—are set differently. We expect the obtained relative ℓ_1 distances between each reference word and the 30 other words in an RMI-constructed VSM to be the same as the obtained relative distances in the original high-dimensional VSM. Therefore, for each VSM that is constructed by the RMI technique, the resulting sorted lists of words are compared by the sorted lists that are obtained from the original high-dimensional VSM.

We employ the *Spearman's rank correlation coefficient* (ρ) to compare the sorted lists of words and thus the degree of distance preservation in the RMI-constructed VSMs at reduced dimensionality. The Spearman's rank correlation measures the strength of association between two ranked variables, i.e. two lists of sorted words in our experiments. Given a list of sorted words obtained from the original high-dimensional VSM (list_o) and its corresponding list obtained from a VSM of reduced dimensionality (list_{RMI}), the Spearman's rank correlation for the two lists is calculated by

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2-1)}, \qquad (11)$$

where d_i is the difference in paired ranks of words in list_o and list_{RMI}, and $n = 30$ is the number of words in each list. We report the average of ρ over the 31 lists of sorted words, denoted by $\bar{\rho}$, to

[11] E.g. see the experiments in Bullinaria and Levy (2007).

[12] UkWaC can be obtained from http://goo.gl/3isfIE.

[13] Please note that the number of possible arrangements of 30 words without repetition in a list in which the order is important (i.e. all permutations of 30 words) is 30!.

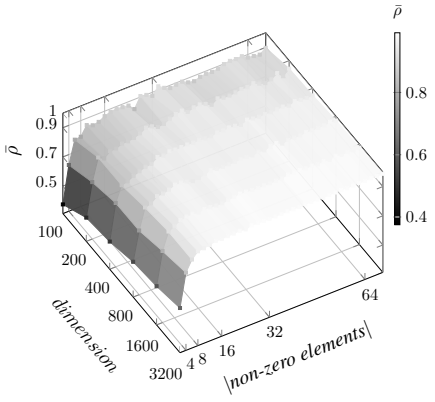


Figure 3: The $\bar{\rho}$ axis shows the observed average Spearman' rank correlation between the order of the words in the lists that are sorted by the ℓ_1 distance obtained from the original high-dimensional VSM and the VSMs that are constructed by RMI at reduced dimensionality using index vectors of various numbers of non-zero elements.

indicate the performance of RMI with respect to its ability for distance preservation. The closer $\bar{\rho}$ is to 1, the better the performance of RMI.

Figure 3 shows the observed results at a glance when the distances are estimated using the median (Equation 7). As shown in the figure, when the dimension of the VSM is above 400 and the number of non-zero elements is more than 12, the obtained relative distances from the VSMs constructed by the RMI technique start to be analogous to the relative distances that are obtained from the original high-dimensional VSM, i.e. a high correlation ($\bar{\rho} > 0.90$). For the baseline, we report the average correlation of $\bar{\rho}_{random} = -0.004$ between the sorted lists of words obtained from the high-dimensional VSM and 31×1000 lists of sorted words that are obtained by randomly assigned distances.

Figure 4 shows the same results as Figure 3, however, in minute detail and only for VSMs of dimension $m \in \{100, 400, 800, 3200\}$. In these plots, squares (■) indicate the $\bar{\rho}$ while the error bars show the best and the worst observed ρ amongst all the sorted lists of words. The minimum value of ρ-axis is set to 0.611, which is the worst observed correlation in the baseline (i.e. randomly generated distances). The dotted line ($\rho = .591$) shows the best observed correlation in the baseline and the dashed-dotted line shows the average correlation in the baseline ($\rho = -0.004$). As suggested in Section 2.2, it can be verified that an increase in the dimension of VSMs (i.e. m) increases the stability of the obtained results (i.e. the probability of preserving distances increases). Therefore, for large values of m (i.e. $m > 400$), the difference between the best and the worst observed ρ decreases; average correlation $\bar{\rho} \to 1$ and the observed relative distances in RMI-constructed VSMs tend to be identical to those in the original high-dimensional VSM.

Figure 5 represents the obtained results in the same setting as above, however, when the distances are approximated using the geometric mean (Equation 10). The obtained average correlations $\bar{\rho}$ from the geometric mean estimations are almost identical to the median estimations. However, as expected, the geometric mean estimations are more reliable for small values of m; particularly, the worst observed correlations when using the geometric mean are higher than those observed when using the median estimator.

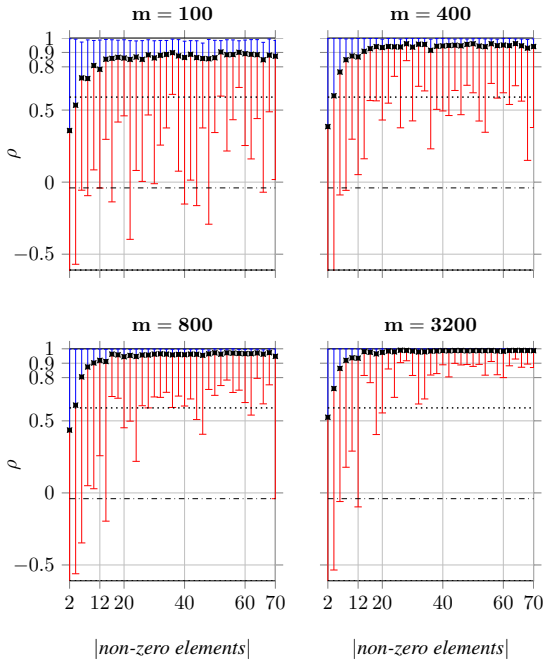


Figure 4: Detailed observation of the obtained correlation between relative distances in RMI-constructed VSMs and the original high-dimensional VSM. The ℓ_1 distance is estimated using the median. The squares denote $\bar{\rho}$ and the error bars show the best and the worst observed correlations. The dashed-dotted line shows the random baseline.

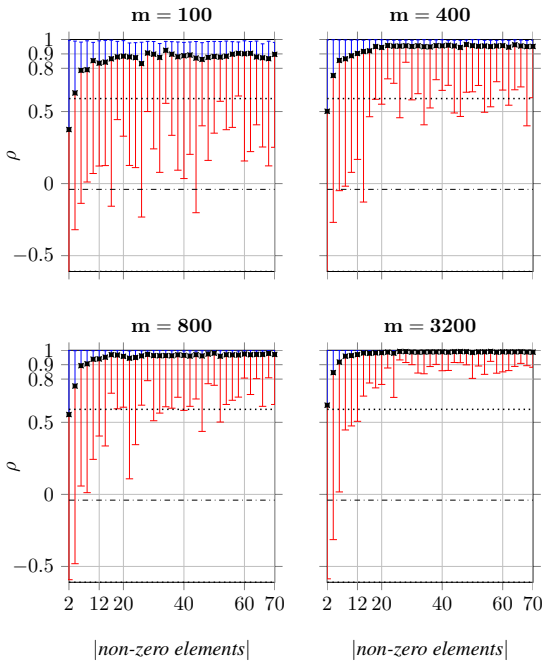


Figure 5: The observed results when the ℓ_1 distance in RMI-constructed VSMs is estimated using the geometric mean.

3 Random Manhattan Integer Indexing

The application of the RMI method is hindered by two obstacles: float arithmetic operations required for the construction and processing of the RMI-constructed VSMs and the calculation of the product of large numbers when ℓ_1 distances are estimated using the geometric mean.

The proposed method for the generation of index vectors in RMI results in index vectors of non-zero elements that are real numbers. Consequently, index vectors and thus context vectors are arrays of floating point numbers. These vectors must be stored and accessed efficiently when using the RMI technique. However, resources that are required for the storage and processing of floating numbers is high. Even if the requirement for the storage of index vectors is alleviated, e.g., using a derandomization technique for their generation, context vectors that are derived from these index vectors are still arrays of float numbers. To tackle this problem, we suggest substituting the value of non-zero elements of RMI's index vectors (given in Equation 6) from $\frac{1}{U}$ to integer values of $\lfloor \frac{1}{U} \rfloor$, where $\lfloor \frac{1}{U} \rfloor \neq 0$. We argue that the resulting random projection matrix still has a Cauchy distribution. Therefore, the proposed methodology to estimate the ℓ_1 distance between vectors is also valid.

The ℓ_1 distance between context vectors must be estimated using either the median or the geometric mean. The use of the median estimator—for the reasons stated in Section 2.1—is not plausible. On the other hand, the computation of the geometric mean can be laborious as the overflow is highly likely to happen during its computation. Using the value of $\lfloor \frac{1}{U} \rfloor$ for non-zero elements of index vectors, we know that for any pair of context vectors $\vec{u} = (u_1, \cdots, u_m)$ and $\vec{v} = (v_1, \cdots, v_m)$, if $u_i \neq v_i$ then $|u_i - v_i| \geq 1$. Therefore, for $u_i \neq v_i$, $\ln |u_i - v_i| \geq 0$ and thus $\sum_{i=1}^{m} \ln(|u_i - v_i|) \geq 0$. In this case, the exponent in Equation 10 is a scale factor that can be discarded without a change in the relative distances between vectors.[14] Based on the intuition that the distance between a vector and itself is zero and the explanation given above, inspired by smoothing techniques and without being able to provide mathematical proofs, we suggest estimating the relative distances between vectors using

$$\hat{L}_1(\vec{u}, \vec{v}) = \sum_{\substack{i=1 \\ u_i \neq v_i}}^{m} \ln(|u_i - v_i|). \qquad (12)$$

In order to distinguish the above changes in RMI, we name the resulting technique random Manhattan integer indexing (RMII). The experiment described in Section 2.2 is repeated using the RMII method. As shown in Figure 6, the obtained results are almost identical to the observed results when using the RMI technique. While RMI performs slightly better than RMII in lower dimensions, e.g. $m = 400$, RMII shows more stable behaviour than RMI at higher dimensions, e.g. $m = 800$.

4 Comparison of RMI and RI

RMI and RI utilize a similar two-step procedure consisting of the creation of index vectors and the construction of context vectors. Both methods are incremental techniques that construct a VSM at reduced dimensionality directly, without requiring the VSM to be constructed at its original high dimension. Despite these similarities, RMI and RI are motivated by different applications and math-

[14] Please note that according to the axioms in the distance definition, the distance between two numbers is always a non-negative value. When index vectors consist of non-zero elements of real numbers, the value of $|u_i - v_i|$ can be between 0 and 1, i.e. $0 < |u_i - v_i| < 1$. Therefore, $\ln(|u_i - v_i|)$ can be a negative number and thus the exponent scale is required to make sure that the result is a non-negative number.

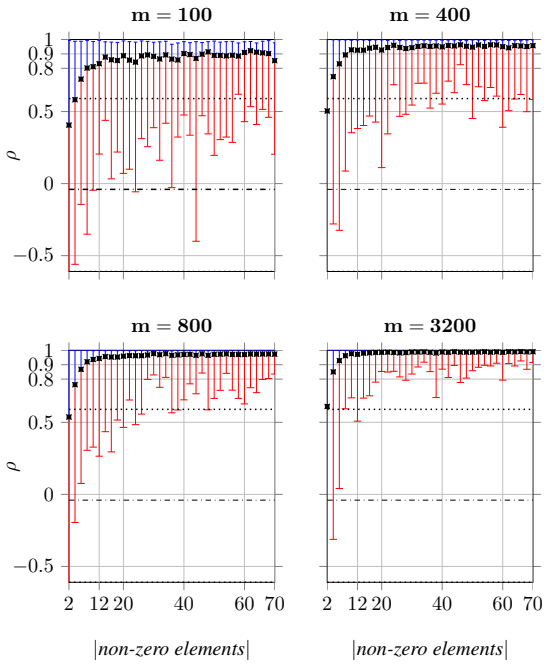


Figure 6: The observed results when using the RMII method for the construction and estimation of the ℓ_1 distances between vectors. The method is evaluated in the same setup as the RMI technique.

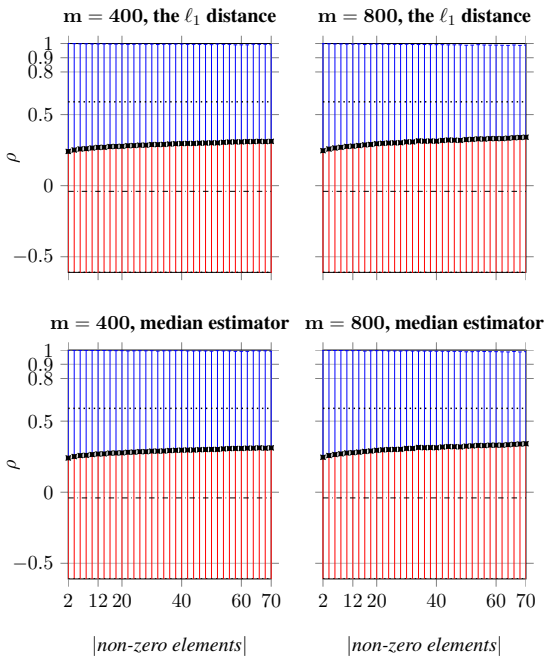


Figure 7: Evaluation of RI for the ℓ_1 distance estimation for $m = 400$ and $m = 800$ when the distances are calculated using the standard definition of distance in ℓ_1 normed spaces and the median estimator. The obtained results using RI do not show correlation to the ℓ_1 distances in the original high-dimensional VSM.

ematical theorems. As described above, RMI approximates the ℓ_1 distance using a *non-linear estimator*, which has not yet been employed for the construction of VSMs and the calculation of ℓ_1 distances in distributional approaches to semantics. Moreover, RMI is justified using Cauchy random projections.

In contrast, RI approximates the ℓ_2 distance using a linear estimator. RI has initially been justified using the mathematical model of the sparse distributed memory (SDM)[15]. Later, Sahlgren (2005) delineates the RI method using the lemma proposed by Johnson and Lindenstrauss (1984)—which elucidates random projections in Euclidean spaces—and the reported refinement in Achlioptas (2001) for the projections employed in the lemma. Although both the RMI and RI methods can be established as α-stable random projections—respectively for $\alpha = 1$ and $\alpha = 2$—the methods cannot be compared as they address different goals. If, for a given task, the ℓ_1 norm outperforms the ℓ_2 norm, then RMI is preferable to RI. Contrariwise, if the ℓ_2 norm outperforms the ℓ_1 norm, then RI is preferable to RMI.

To support the earlier claim that RI-constructed VSMs cannot be used for the ℓ_1 distance estimation, we evaluate the RI method in the experimental setup that has been used for the evaluation of RMI and RMII. In these experiments, however, we use RI to construct vector spaces at reduced dimensionality and estimate the ℓ_1 distance using Equation 5 (the standard ℓ_1 distance definition) and Equation 7 (the median estimator) for $m \in 400, 800$. As shown in Figure 7, the experiments support the theoretical claims.

5 Conclusion

In this paper, we introduce a novel technique, named Random Manhattan Indexing (RMI), for the construction of ℓ_1 normed VSMs directly at reduced dimensionality. We further suggest the Random Manhattan Integer Indexing (RMII) technique, a computationally enhanced version of the RMI technique. We demonstrated the ℓ_1 distance preservation ability of the proposed technique in an experimental setup using a word-by-document model. In these experiments, we showed how the variable parameters of the methods, i.e. the number of non-zero elements in index vectors and the

[15] See Kanerva (1993) for an overview of the SDM model.

dimensionality of the VSM, influence the obtained results. The proposed incremental (and thus efficient and scalable) methods significantly enhance the computation of the ℓ_1 distances in VSMs.

Acknowledgements

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant Number SFI/12/RC/2289.

References

[Achlioptas2001] Dimitris Achlioptas. 2001. Database-friendly random projections. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '01, pages 274–281, New York, NY, USA. ACM.

[Baroni et al.2009] Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

[Brand2006] Matthew Brand. 2006. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20–30. Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems.

[Brinkman and Charikar2005] Bo Brinkman and Moses Charikar. 2005. On the impossibility of dimension reduction in l1. *J. ACM*, 52(5):766–788.

[Bullinaria and Levy2007] John A. Bullinaria and Joseph P. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526.

[Deerwester et al.1990] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

[Indyk2000] Piotr Indyk. 2000. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 189–197.

[Indyk2006] Piotr Indyk. 2006. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, May.

[Johnson and Lindenstrauss1984] William Johnson and Joram Lindenstrauss. 1984. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, volume 26 of *Contemporary Mathematics*, pages 189–206. American Mathematical Society.

[Kanerva et al.2000] Pentti Kanerva, Jan Kristoferson, and Anders Holst. 2000. Random indexing of text samples for latent semantic analysis. In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, pages 103–6. Erlbaum.

[Kanerva1993] Pentti Kanerva. 1993. Sparse distributed memory and related models. In Mohamad H. Hassoun, editor, *Associative neural memories: theory and implementation*, chapter 3, pages 50–76. Oxford University Press, Inc., New York, NY, USA.

[Ke and Kanade2003] Qifa Ke and Takeo Kanade. 2003. Robust subspace computation using ℓ_1 norm. Technical Report CMU-CS-03-172, School of Computer Science, Carnegie Mellon University.

[Kwak2008] Nojun Kwak. 2008. Principal component analysis based on l1-norm maximization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(9):1672–1680, Sept.

[Lee1999] Lillian Lee. 1999. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 25–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Lenci2008] Alessandro Lenci. 2008. Distributional semantics in linguistic and cognitive research. *From context to meaning: Distributional models of the lexicon in linguistics and cognitive science, special issue of the Italian Journal of Linguistics*, 20/1:1–31.

[Li et al.2007] Ping Li, Trevor J. Hastie, and Kenneth W. Church. 2007. Nonlinear estimators and tail bounds for dimension reduction in L_1 using cauchy random projections. *J. Mach. Learn. Res.*, 8:2497–2532.

[Li2007] Ping Li. 2007. Very sparse stable random projections for dimension reduction in l_α $(0 < \alpha < 2)$ norm. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 440–449, New York, NY, USA. ACM.

[Li2008] Ping Li. 2008. Estimators and tail bounds for dimension reduction in ℓ_α $(0 < \alpha \leq 2)$ using stable random projections. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 10–19, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

[Martin and Berry2007] Dian I. Martin and Michael W. Berry, 2007. *Handbook of latent semantic analysis*, chapter Mathematical foundations behind latent semantic analysis, pages 35–55. Ro.

[Sahlgren2005] Magnus Sahlgren. 2005. An introduction to random indexing. In *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005*.

[Turney and Pantel2010] Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188, January.

[Weeds et al.2005] Julie Weeds, James Dowdall, Gerold Schneider, Bill Keller, and David Weir. 2005. Using distributional similarity to organise biomedical terminology. *Terminology*, 11(1):3–4.

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

Kyunghyun Cho
Bart van Merriënboer **Caglar Gulcehre**
Université de Montréal
firstname.lastname@umontreal.ca

Dzmitry Bahdanau
Jacobs University, Germany
d.bahdanau@jacobs-university.de

Fethi Bougares **Holger Schwenk**
Université du Maine, France
firstname.lastname@lium.univ-lemans.fr

Yoshua Bengio
Université de Montréal, CIFAR Senior Fellow
find.me@on.the.web

Abstract

In this paper, we propose a novel neural network model called RNN Encoder–Decoder that consists of two recurrent neural networks (RNN). One RNN encodes a sequence of symbols into a fixed-length vector representation, and the other decodes the representation into another sequence of symbols. The encoder and decoder of the proposed model are jointly trained to maximize the conditional probability of a target sequence given a source sequence. The performance of a statistical machine translation system is empirically found to improve by using the conditional probabilities of phrase pairs computed by the RNN Encoder–Decoder as an additional feature in the existing log-linear model. Qualitatively, we show that the proposed model learns a semantically and syntactically meaningful representation of linguistic phrases.

1 Introduction

Deep neural networks have shown great success in various applications such as objection recognition (see, e.g., (Krizhevsky et al., 2012)) and speech recognition (see, e.g., (Dahl et al., 2012)). Furthermore, many recent works showed that neural networks can be successfully used in a number of tasks in natural language processing (NLP). These include, but are not limited to, language modeling (Bengio et al., 2003), paraphrase detection (Socher et al., 2011) and word embedding extraction (Mikolov et al., 2013). In the field of statistical machine translation (SMT), deep neural networks have begun to show promising results. (Schwenk, 2012) summarizes a successful usage of feedforward neural networks in the framework of phrase-based SMT system.

Along this line of research on using neural networks for SMT, this paper focuses on a novel neural network architecture that can be used as a part of the conventional phrase-based SMT system. The proposed neural network architecture, which we will refer to as an *RNN Encoder–Decoder*, consists of two recurrent neural networks (RNN) that act as an encoder and a decoder pair. The encoder maps a variable-length source sequence to a fixed-length vector, and the decoder maps the vector representation back to a variable-length target sequence. The two networks are trained jointly to maximize the conditional probability of the target sequence given a source sequence. Additionally, we propose to use a rather sophisticated hidden unit in order to improve both the memory capacity and the ease of training.

The proposed RNN Encoder–Decoder with a novel hidden unit is empirically evaluated on the task of translating from English to French. We train the model to learn the translation probability of an English phrase to a corresponding French phrase. The model is then used as a part of a standard phrase-based SMT system by scoring each phrase pair in the phrase table. The empirical evaluation reveals that this approach of scoring phrase pairs with an RNN Encoder–Decoder improves the translation performance.

We qualitatively analyze the trained RNN Encoder–Decoder by comparing its phrase scores with those given by the existing translation model. The qualitative analysis shows that the RNN Encoder–Decoder is better at capturing the linguistic regularities in the phrase table, indirectly explaining the quantitative improvements in the overall translation performance. The further analysis of the model reveals that the RNN Encoder–Decoder learns a continuous space representation of a phrase that preserves both the semantic and syntactic structure of the phrase.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

2 RNN Encoder–Decoder

2.1 Preliminary: Recurrent Neural Networks

A recurrent neural network (RNN) is a neural network that consists of a hidden state $\mathbf{h}$ and an optional output $\mathbf{y}$ which operates on a variable-length sequence $\mathbf{x} = (x_1, \ldots, x_T)$. At each time step t, the hidden state $\mathbf{h}_{\langle t \rangle}$ of the RNN is updated by

$$\mathbf{h}_{\langle t \rangle} = f\left(\mathbf{h}_{\langle t-1 \rangle}, x_t\right), \tag{1}$$

where f is a non-linear activation function. f may be as simple as an element-wise logistic sigmoid function and as complex as a long short-term memory (LSTM) unit (Hochreiter and Schmidhuber, 1997).

An RNN can learn a probability distribution over a sequence by being trained to predict the next symbol in a sequence. In that case, the output at each timestep t is the conditional distribution $p(x_t \mid x_{t-1}, \ldots, x_1)$. For example, a multinomial distribution (1-of-K coding) can be output using a softmax activation function

$$p(x_{t,j} = 1 \mid x_{t-1}, \ldots, x_1) = \frac{\exp\left(\mathbf{w}_j \mathbf{h}_{\langle t \rangle}\right)}{\sum_{j'=1}^{K} \exp\left(\mathbf{w}_{j'} \mathbf{h}_{\langle t \rangle}\right)}, \tag{2}$$

for all possible symbols $j = 1, \ldots, K$, where $\mathbf{w}_j$ are the rows of a weight matrix $\mathbf{W}$. By combining these probabilities, we can compute the probability of the sequence $\mathbf{x}$ using

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t \mid x_{t-1}, \ldots, x_1). \tag{3}$$

From this learned distribution, it is straightforward to sample a new sequence by iteratively sampling a symbol at each time step.

2.2 RNN Encoder–Decoder

In this paper, we propose a novel neural network architecture that learns to *encode* a variable-length sequence into a fixed-length vector representation and to *decode* a given fixed-length vector representation back into a variable-length sequence. From a probabilistic perspective, this new model is a general method to learn the conditional distribution over a variable-length sequence conditioned on yet another variable-length sequence, e.g. $p(y_1, \ldots, y_{T'} \mid x_1, \ldots, x_T)$, where one should note that the input and output sequence lengths T and T' may differ.

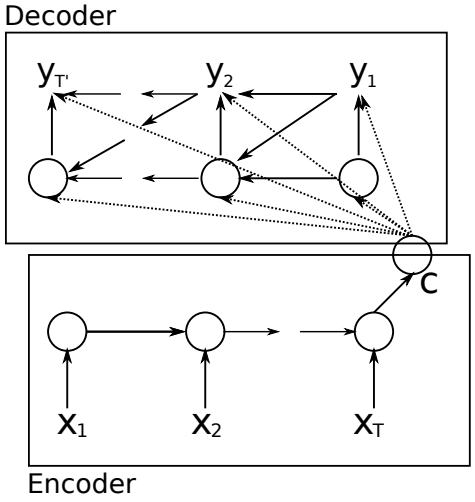


Figure 1: An illustration of the proposed RNN Encoder–Decoder.

The encoder is an RNN that reads each symbol of an input sequence $\mathbf{x}$ sequentially. As it reads each symbol, the hidden state of the RNN changes according to Eq. (1). After reading the end of the sequence (marked by an end-of-sequence symbol), the hidden state of the RNN is a summary $\mathbf{c}$ of the whole input sequence.

The decoder of the proposed model is another RNN which is trained to *generate* the output sequence by predicting the next symbol y_t given the hidden state $\mathbf{h}_{\langle t \rangle}$. However, unlike the RNN described in Sec. 2.1, both y_t and $\mathbf{h}_{\langle t \rangle}$ are also conditioned on y_{t-1} and on the summary $\mathbf{c}$ of the input sequence. Hence, the hidden state of the decoder at time t is computed by,

$$\mathbf{h}_{\langle t \rangle} = f\left(\mathbf{h}_{\langle t-1 \rangle}, y_{t-1}, \mathbf{c}\right),$$

and similarly, the conditional distribution of the next symbol is

$$P(y_t | y_{t-1}, y_{t-2}, \ldots, y_1, \mathbf{c}) = g\left(\mathbf{h}_{\langle t \rangle}, y_{t-1}, \mathbf{c}\right).$$

for given activation functions f and g (the latter must produce valid probabilities, e.g. with a softmax).

See Fig. 1 for a graphical depiction of the proposed model architecture.

The two components of the proposed *RNN Encoder–Decoder* are jointly trained to maximize the conditional log-likelihood

$$\max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{y}_n \mid \mathbf{x}_n), \tag{4}$$

where $\boldsymbol{\theta}$ is the set of the model parameters and each $(\mathbf{x}_n, \mathbf{y}_n)$ is an (input sequence, output sequence) pair from the training set. In our case, as the output of the decoder, starting from the input, is differentiable, we can use a gradient-based algorithm to estimate the model parameters.

Once the RNN Encoder–Decoder is trained, the model can be used in two ways. One way is to use the model to generate a target sequence given an input sequence. On the other hand, the model can be used to *score* a given pair of input and output sequences, where the score is simply a probability $p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$ from Eqs. (3) and (4).

2.3 Hidden Unit that Adaptively Remembers and Forgets

In addition to a novel model architecture, we also propose a new type of hidden unit (f in Eq. (1)) that has been motivated by the LSTM unit but is much simpler to compute and implement.[1] Fig. 2 shows the graphical depiction of the proposed hidden unit.

Let us describe how the activation of the j-th hidden unit is computed. First, the *reset* gate r_j is computed by

$$r_j = \sigma\left(\left[\mathbf{W}_r\mathbf{x}\right]_j + \left[\mathbf{U}_r\mathbf{h}_{\langle t-1\rangle}\right]_j\right), \quad (5)$$

where σ is the logistic sigmoid function, and $[.]_j$ denotes the j-th element of a vector. $\mathbf{x}$ and $\mathbf{h}_{t-1}$ are the input and the previous hidden state, respectively. $\mathbf{W}_r$ and $\mathbf{U}_r$ are weight matrices which are learned.

Similarly, the *update* gate z_j is computed by

$$z_j = \sigma\left(\left[\mathbf{W}_z\mathbf{x}\right]_j + \left[\mathbf{U}_z\mathbf{h}_{\langle t-1\rangle}\right]_j\right). \quad (6)$$

The actual activation of the proposed unit h_j is then computed by

$$h_j^{\langle t\rangle} = z_j h_j^{\langle t-1\rangle} + (1 - z_j)\tilde{h}_j^{\langle t\rangle}, \quad (7)$$

where

$$\tilde{h}_j^{\langle t\rangle} = \phi\left(\left[\mathbf{W}\mathbf{x}\right]_j + \left[\mathbf{U}\left(\mathbf{r} \odot \mathbf{h}_{\langle t-1\rangle}\right)\right]_j\right). \quad (8)$$

In this formulation, when the reset gate is close to 0, the hidden state is forced to ignore the previous hidden state and reset with the current input only. This effectively allows the hidden state to *drop* any information that is found to be irrelevant later in the future, thus, allowing a more compact representation.

On the other hand, the update gate controls how much information from the previous hidden state will carry over to the current hidden state. This acts similarly to the memory cell in the LSTM network and helps the RNN to remember long-term information. Furthermore, this may be considered an adaptive variant of a leaky-integration unit (Bengio et al., 2013).

As each hidden unit has separate reset and update gates, each hidden unit will learn to capture dependencies over different time scales. Those units that learn to capture short-term dependencies will tend to have reset gates that are frequently active, but those that capture longer-term dependencies will have update gates that are mostly active.

In our preliminary experiments, we found that it is crucial to use this new unit with gating units. We were not able to get meaningful result with an oft-used $\tanh$ unit without any gating.

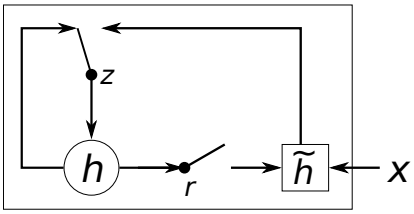


Figure 2: An illustration of the proposed hidden activation function. The update gate z selects whether the hidden state is to be updated with a new hidden state $\tilde{h}$. The reset gate r decides whether the previous hidden state is ignored. See Eqs. (5)–(8) for the detailed equations of r, z, h and $\tilde{h}$.

3 Statistical Machine Translation

In a commonly used statistical machine translation system (SMT), the goal of the system (decoder, specifically) is to find a translation $\mathbf{f}$ given a source sentence $\mathbf{e}$, which maximizes

$$p(\mathbf{f} \mid \mathbf{e}) \propto p(\mathbf{e} \mid \mathbf{f})p(\mathbf{f}),$$

where the first term at the right hand side is called *translation model* and the latter *language model* (see, e.g., (Koehn, 2005)). In practice, however, most SMT systems model $\log p(\mathbf{f} \mid \mathbf{e})$ as a log-linear model with additional features and corre-

[1] The LSTM unit, which has shown impressive results in several applications such as speech recognition, has a memory cell and four gating units that adaptively control the information flow inside the unit, compared to only two gating units in the proposed hidden unit. For details on LSTM networks, see, e.g., (Graves, 2012).

sponding weights:

$$\log p(\mathbf{f} \mid \mathbf{e}) = \sum_{n=1}^{N} w_n f_n(\mathbf{f}, \mathbf{e}) + \log Z(\mathbf{e}), \quad (9)$$

where f_n and w_n are the n-th feature and weight, respectively. $Z(\mathbf{e})$ is a normalization constant that does not depend on the weights. The weights are often optimized to maximize the BLEU score on a development set.

In the phrase-based SMT framework introduced in (Koehn et al., 2003) and (Marcu and Wong, 2002), the translation model $\log p(\mathbf{e} \mid \mathbf{f})$ is factorized into the translation probabilities of matching phrases in the source and target sentences.[2] These probabilities are once again considered additional features in the log-linear model (see Eq. (9)) and are weighted accordingly to maximize the BLEU score.

Since the neural net language model was proposed in (Bengio et al., 2003), neural networks have been used widely in SMT systems. In many cases, neural networks have been used to *rescore* translation hypotheses (n-best lists) (see, e.g., (Schwenk et al., 2006)). Recently, however, there has been interest in training neural networks to score the translated sentence (or phrase pairs) using a representation of the source sentence as an additional input. See, e.g., (Schwenk, 2012), (Son et al., 2012) and (Zou et al., 2013).

3.1 Scoring Phrase Pairs with RNN Encoder–Decoder

Here we propose to train the RNN Encoder–Decoder (see Sec. 2.2) on a table of phrase pairs and use its scores as additional features in the log-linear model in Eq. (9) when tuning the SMT decoder.

When we train the RNN Encoder–Decoder, we ignore the (normalized) frequencies of each phrase pair in the original corpora. This measure was taken in order (1) to reduce the computational expense of randomly selecting phrase pairs from a large phrase table according to the normalized frequencies and (2) to ensure that the RNN Encoder–Decoder does not simply learn to rank the phrase pairs according to their numbers of occurrences. One underlying reason for this choice was that the existing translation probability in the phrase table already reflects the frequencies of the phrase pairs in the original corpus. With a fixed capacity of the RNN Encoder–Decoder, we try to ensure that most of the capacity of the model is focused toward learning linguistic regularities, i.e., distinguishing between plausible and implausible translations, or learning the "manifold" (region of probability concentration) of plausible translations.

Once the RNN Encoder–Decoder is trained, we add a new score for each phrase pair to the existing phrase table. This allows the new scores to enter into the existing tuning algorithm with minimal additional overhead in computation.

As Schwenk pointed out in (Schwenk, 2012), it is possible to completely replace the existing phrase table with the proposed RNN Encoder–Decoder. In that case, for a given source phrase, the RNN Encoder–Decoder will need to generate a list of (good) target phrases. This requires, however, an expensive sampling procedure to be performed repeatedly. In this paper, thus, we only consider rescoring the phrase pairs in the phrase table.

3.2 Related Approaches: Neural Networks in Machine Translation

Before presenting the empirical results, we discuss a number of recent works that have proposed to use neural networks in the context of SMT.

Schwenk in (Schwenk, 2012) proposed a similar approach of scoring phrase pairs. Instead of the RNN-based neural network, he used a feedforward neural network that has fixed-size inputs (7 words in his case, with zero-padding for shorter phrases) and fixed-size outputs (7 words in the target language). When it is used specifically for scoring phrases for the SMT system, the maximum phrase length is often chosen to be small. However, as the length of phrases increases or as we apply neural networks to other variable-length sequence data, it is important that the neural network can handle variable-length input and output. The proposed RNN Encoder–Decoder is well-suited for these applications.

Similar to (Schwenk, 2012), Devlin et al. (Devlin et al., 2014) proposed to use a feedforward neural network to model a translation model, however, by predicting one word in a target phrase at a time. They reported an impressive improvement, but their approach still requires the maximum length of the input phrase (or context words) to be fixed a priori.

[2] Without loss of generality, from here on, we refer to $p(\mathbf{e} \mid \mathbf{f})$ for each phrase pair as a translation model as well

Although it is not exactly a neural network they train, the authors of (Zou et al., 2013) proposed to learn a bilingual embedding of words/phrases. They use the learned embedding to compute the distance between a pair of phrases which is used as an additional score of the phrase pair in an SMT system.

In (Chandar et al., 2014), a feedforward neural network was trained to learn a mapping from a bag-of-words representation of an input phrase to an output phrase. This is closely related to both the proposed RNN Encoder–Decoder and the model proposed in (Schwenk, 2012), except that their input representation of a phrase is a bag-of-words. A similar approach of using bag-of-words representations was proposed in (Gao et al., 2013) as well. Earlier, a similar encoder–decoder model using two recursive neural networks was proposed in (Socher et al., 2011), but their model was restricted to a monolingual setting, i.e. the model reconstructs an input sentence. More recently, another encoder–decoder model using an RNN was proposed in (Auli et al., 2013), where the decoder is conditioned on a representation of either a source sentence or a source context.

One important difference between the proposed RNN Encoder–Decoder and the approaches in (Zou et al., 2013) and (Chandar et al., 2014) is that the order of the words in source and target phrases is taken into account. The RNN Encoder–Decoder naturally distinguishes between sequences that have the same words but in a different order, whereas the aforementioned approaches effectively ignore order information.

The closest approach related to the proposed RNN Encoder–Decoder is the Recurrent Continuous Translation Model (Model 2) proposed in (Kalchbrenner and Blunsom, 2013). In their paper, they proposed a similar model that consists of an encoder and decoder. The difference with our model is that they used a convolutional n-gram model (CGM) for the encoder and the hybrid of an inverse CGM and a recurrent neural network for the decoder. They, however, evaluated their model on rescoring the n-best list proposed by the conventional SMT system and computing the perplexity of the gold standard translations.

4 Experiments

We evaluate our approach on the English/French translation task of the WMT'14 workshop.

4.1 Data and Baseline System

Large amounts of resources are available to build an English/French SMT system in the framework of the WMT'14 translation task. The bilingual corpora include Europarl (61M words), news commentary (5.5M), UN (421M), and two crawled corpora of 90M and 780M words respectively. The last two corpora are quite noisy. To train the French language model, about 712M words of crawled newspaper material is available in addition to the target side of the bitexts. All the word counts refer to French words after tokenization.

It is commonly acknowledged that training statistical models on the concatenation of all this data does not necessarily lead to optimal performance, and results in extremely large models which are difficult to handle. Instead, one should focus on the most relevant subset of the data for a given task. We have done so by applying the data selection method proposed in (Moore and Lewis, 2010), and its extension to bitexts (Axelrod et al., 2011). By these means we selected a subset of 418M words out of more than 2G words for language modeling and a subset of 348M out of 850M words for training the RNN Encoder–Decoder. We used the test set `newstest2012 and 2013` for data selection and weight tuning with MERT, and `newstest2014` as our test set. Each set has more than 70 thousand words and a single reference translation.

For training the neural networks, including the proposed RNN Encoder–Decoder, we limited the source and target vocabulary to the most frequent 15,000 words for both English and French. This covers approximately 93% of the dataset. All the out-of-vocabulary words were mapped to a special token ([UNK]).

The baseline phrase-based SMT system was built using Moses with default settings. This system achieves a BLEU score of 30.64 and 33.3 on the development and test sets, respectively (see Table 1).

4.1.1 RNN Encoder–Decoder

The RNN Encoder–Decoder used in the experiment had 1000 hidden units with the proposed gates at the encoder and at the decoder. The input matrix between each input symbol $x_{\langle t \rangle}$ and the hidden unit is approximated with two lower-rank matrices, and the output matrix is approximated

Models	BLEU	
	dev	test
Baseline	30.64	33.30
RNN	31.20	33.87
CSLM + RNN	31.48	34.64
CSLM + RNN + WP	31.50	34.54

Table 1: BLEU scores computed on the development and test sets using different combinations of approaches. WP denotes a *word penalty*, where we penalizes the number of unknown words to neural networks.

similarly. We used rank-100 matrices, equivalent to learning an embedding of dimension 100 for each word. The activation function used for $\tilde{h}$ in Eq. (8) is a hyperbolic tangent function. The computation from the hidden state in the decoder to the output is implemented as a deep neural network (Pascanu et al., 2014) with a single intermediate layer having 500 maxout units each pooling 2 inputs (Goodfellow et al., 2013).

All the weight parameters in the RNN Encoder–Decoder were initialized by sampling from an isotropic zero-mean (white) Gaussian distribution with its standard deviation fixed to 0.01, except for the recurrent weight parameters. For the recurrent weight matrices, we first sampled from a white Gaussian distribution and used its left singular vectors matrix, following (Saxe et al., 2014).

We used Adadelta and stochastic gradient descent to train the RNN Encoder–Decoder with hyperparameters $\epsilon = 10^{-6}$ and $\rho = 0.95$ (Zeiler, 2012). At each update, we used 64 randomly selected phrase pairs from a phrase table (which was created from 348M words). The model was trained for approximately three days.

Details of the architecture used in the experiments are explained in more depth in the supplementary material.

4.1.2 Neural Language Model

In order to assess the effectiveness of scoring phrase pairs with the proposed RNN Encoder–Decoder, we also tried a more traditional approach of using a neural network for learning a target language model (CSLM) (Schwenk, 2007). Especially, the comparison between the SMT system using CSLM and that using the proposed approach of phrase scoring by RNN Encoder–Decoder will clarify whether the contributions from multiple neural networks in different parts of the SMT system add up or are redundant.

We trained the CSLM model on 7-grams from the target corpus. Each input word was projected into the embedding space $\mathbb{R}^{512}$, and they were concatenated to form a 3072-dimensional vector. The concatenated vector was fed through two rectified layers (of size 1536 and 1024) (Glorot et al., 2011). The output layer was a simple softmax layer (see Eq. (2)). All the weight parameters were initialized uniformly between -0.01 and 0.01, and the model was trained until the validation perplexity did not improve for 10 epochs. After training, the language model achieved a perplexity of 45.80. The validation set was a random selection of 0.1% of the corpus. The model was used to score partial translations during the decoding process, which generally leads to higher gains in BLEU score than n-best list rescoring (Vaswani et al., 2013).

To address the computational complexity of using a CSLM in the decoder a buffer was used to aggregate n-grams during the stack-search performed by the decoder. Only when the buffer is full, or a stack is about to be pruned, the n-grams are scored by the CSLM. This allows us to perform fast matrix-matrix multiplication on GPU using Theano (Bergstra et al., 2010; Bastien et al., 2012).

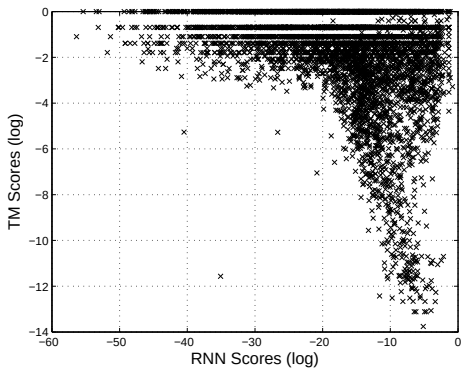


Figure 3: The visualization of phrase pairs according to their scores (log-probabilities) by the RNN Encoder–Decoder and the translation model.

4.2 Quantitative Analysis

We tried the following combinations:

1. Baseline configuration
2. Baseline + RNN
3. Baseline + CSLM + RNN
4. Baseline + CSLM + RNN + Word penalty

Source	Translation Model	RNN Encoder–Decoder
at the end of the	[a la fin de la] [f la fin des années] [être supprimés à la fin de la]	[à la fin du] [à la fin des] [à la fin de la]
for the first time	[**r** © pour la premi**r**ère fois] [été donnés pour la première fois] [été commémorée pour la première fois]	[pour la première fois] [pour la première fois ,] [pour la première fois que]
in the United States and	[**?** aux **?**tats-Unis et] [été ouvertes aux États-Unis et] [été constatées aux États-Unis et]	[aux Etats-Unis et] [des Etats-Unis et] [des États-Unis et]
, as well as	[**?**s , qu'] [**?**s , ainsi que] [**?**re aussi bien que]	[, ainsi qu'] [, ainsi que] [, ainsi que les]
one of the most	[**?**t **?**l' un des plus] [**?**l' un des plus] [être retenue comme un de ses plus]	[l' un des] [le] [un des]

(a) Long, frequent source phrases

Source	Translation Model	RNN Encoder–Decoder
, Minister of Communications and Transport	[Secrétaire aux communications et aux transports :] [Secrétaire aux communications et aux transports]	[Secrétaire aux communications et aux transports] [Secrétaire aux communications et aux transports :]
did not comply with the	[vestimentaire , ne correspondaient pas à des] [susmentionnée n' était pas conforme aux] [présentées n' étaient pas conformes à la]	[n' ont pas respecté les] [n' était pas conforme aux] [n' ont pas respecté la]
parts of the world .	[© gions du monde .] [régions du monde considérées .] [région du monde considérée .]	[parties du monde .] [les parties du monde .] [des parties du monde .]
the past few days .	[le petit texte .] [cours des tout derniers jours .] [les tout derniers jours .]	[ces derniers jours .] [les derniers jours .] [cours des derniers jours .]
on Friday and Saturday	[vendredi et samedi à la] [vendredi et samedi à] [se déroulera vendredi et samedi ,]	[le vendredi et le samedi] [le vendredi et samedi] [vendredi et samedi]

(b) Long, rare source phrases

Table 2: The top scoring target phrases for a small set of source phrases according to the translation model (direct translation probability) and by the RNN Encoder–Decoder. Source phrases were randomly selected from phrases with 4 or more words. **?** denotes an incomplete (partial) character. **r** is a Cyrillic letter *ghe*.

The results are presented in Table 1. As expected, adding features computed by neural networks consistently improves the performance over the baseline performance.

The best performance was achieved when we used both CSLM and the phrase scores from the RNN Encoder–Decoder. This suggests that the contributions of the CSLM and the RNN Encoder–Decoder are not too correlated and that one can expect better results by improving each method independently. Furthermore, we tried penalizing the number of words that are unknown to the neural networks (i.e. words which are not in the shortlist). We do so by simply adding the number of unknown words as an additional feature the log-linear model in Eq. (9).[3] However, in this case we were not able to achieve better performance on the test set, but only on the development set.

4.3 Qualitative Analysis

In order to understand where the performance improvement comes from, we analyze the phrase pair scores computed by the RNN Encoder–Decoder against the corresponding $p(\mathbf{f} \mid \mathbf{e})$ from the translation model. Since the existing translation model relies solely on the statistics of the phrase pairs in the corpus, we expect its scores to be better estimated for the frequent phrases but badly estimated for rare phrases. Also, as we mentioned earlier in Sec. 3.1, we further expect the RNN Encoder–Decoder which was trained without any frequency information to score the phrase pairs based rather on the linguistic regularities than on the statistics of their occurrences in the corpus.

We focus on those pairs whose source phrase is *long* (more than 3 words per source phrase) and

[3] To understand the effect of the penalty, consider the set of all words in the 15,000 large shortlist, SL. All words $x^i \notin$ SL are replaced by a special token [UNK] before being scored by the neural networks. Hence, the conditional probability of any $x_t^i \notin$ SL is actually given by the model as

$$p(x_t = [\text{UNK}] \mid x_{<t}) = p(x_t \notin \text{SL} \mid x_{<t}) = \sum_{x_t^j \notin SL} p\left(x_t^j \mid x_{<t}\right) \geq p\left(x_t^i \mid x_{<t}\right),$$

where $x_{<t}$ is a shorthand notation for $x_{t-1}, \ldots, x_1$.

As a result, the probability of words not in the shortlist is always overestimated. It is possible to address this issue by backing off to an existing model that contain non-shortlisted words (see (Schwenk, 2007)) In this paper, however, we opt for introducing a word penalty instead, which counteracts the word probability overestimation.

Source	Samples from RNN Encoder–Decoder
at the end of the	[à la fin de la] (×11)
for the first time	[pour la première fois] (×24) [pour la première fois que] (×2)
in the United States and	[aux États-Unis et] (×6) [dans les États-Unis et] (×4)
, as well as	[, ainsi que] [,] [ainsi que] [, ainsi qu'] [et UNK]
one of the most	[l' un des plus] (×9) [l' un des] (×5) [l' une des plus] (×2)

(a) Long, frequent source phrases

Source	Samples from RNN Encoder–Decoder
, Minister of Communications and Transport	[, ministre des communications et le transport] (×13)
did not comply with the	[n' tait pas conforme aux] [n' a pas respect l'] (×2) [n' a pas respect la] (×3)
parts of the world .	[arts du monde .] (×11) [des arts du monde .] (×7)
the past few days .	[quelques jours .] (×5) [les derniers jours .] (×5) [ces derniers jours .] (×2)
on Friday and Saturday	[vendredi et samedi] (×5) [le vendredi et samedi] (×7) [le vendredi et le samedi] (×4)

(b) Long, rare source phrases

Table 3: Samples generated from the RNN Encoder–Decoder for each source phrase used in Table 2. We show the top-5 target phrases out of 50 samples. They are sorted by the RNN Encoder–Decoder scores.

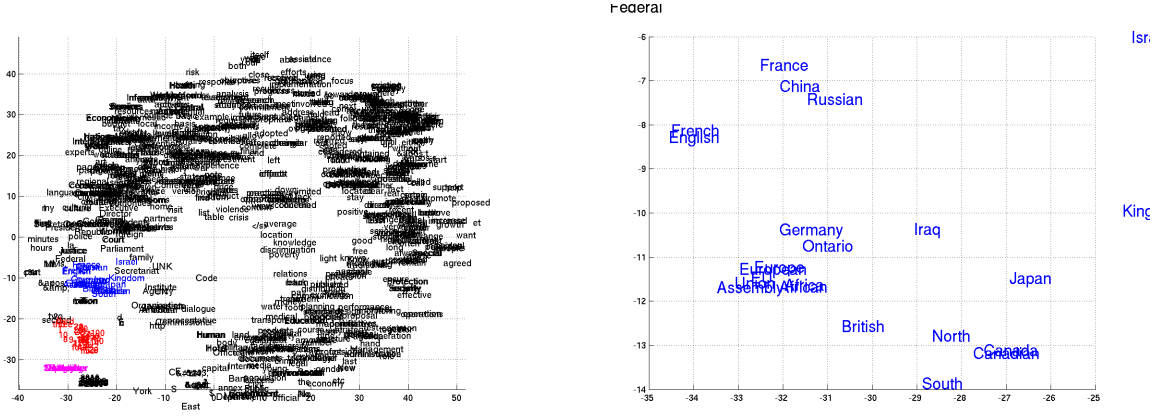


Figure 4: 2–D embedding of the learned word representation. The left one shows the full embedding space, while the right one shows a zoomed-in view of one region (color–coded). For more plots, see the supplementary material.

frequent. For each such source phrase, we look at the target phrases that have been scored high either by the translation probability $p(\mathbf{f} \mid \mathbf{e})$ or by the RNN Encoder–Decoder. Similarly, we perform the same procedure with those pairs whose source phrase is *long* but *rare* in the corpus.

Table 2 lists the top-3 target phrases per source phrase favored either by the translation model or by the RNN Encoder–Decoder. The source phrases were randomly chosen among long ones having more than 4 or 5 words.

In most cases, the choices of the target phrases by the RNN Encoder–Decoder are closer to actual or literal translations. We can observe that the RNN Encoder–Decoder prefers shorter phrases in general.

Interestingly, many phrase pairs were scored similarly by both the translation model and the RNN Encoder–Decoder, but there were as many other phrase pairs that were scored radically different (see Fig. 3). This could arise from the proposed approach of training the RNN Encoder–Decoder on a set of unique phrase pairs, discouraging the RNN Encoder–Decoder from learning simply the frequencies of the phrase pairs from the corpus, as explained earlier.

Furthermore, in Table 3, we show for each of the source phrases in Table 2, the generated samples from the RNN Encoder–Decoder. For each source phrase, we generated 50 samples and show the top-five phrases accordingly to their scores. We can see that the RNN Encoder–Decoder is able to propose well-formed target phrases without looking at the actual phrase table. Importantly, the generated phrases do not overlap completely with the target phrases from the phrase table. This encourages us to further investigate the possibility of replacing the whole or a part of the phrase table

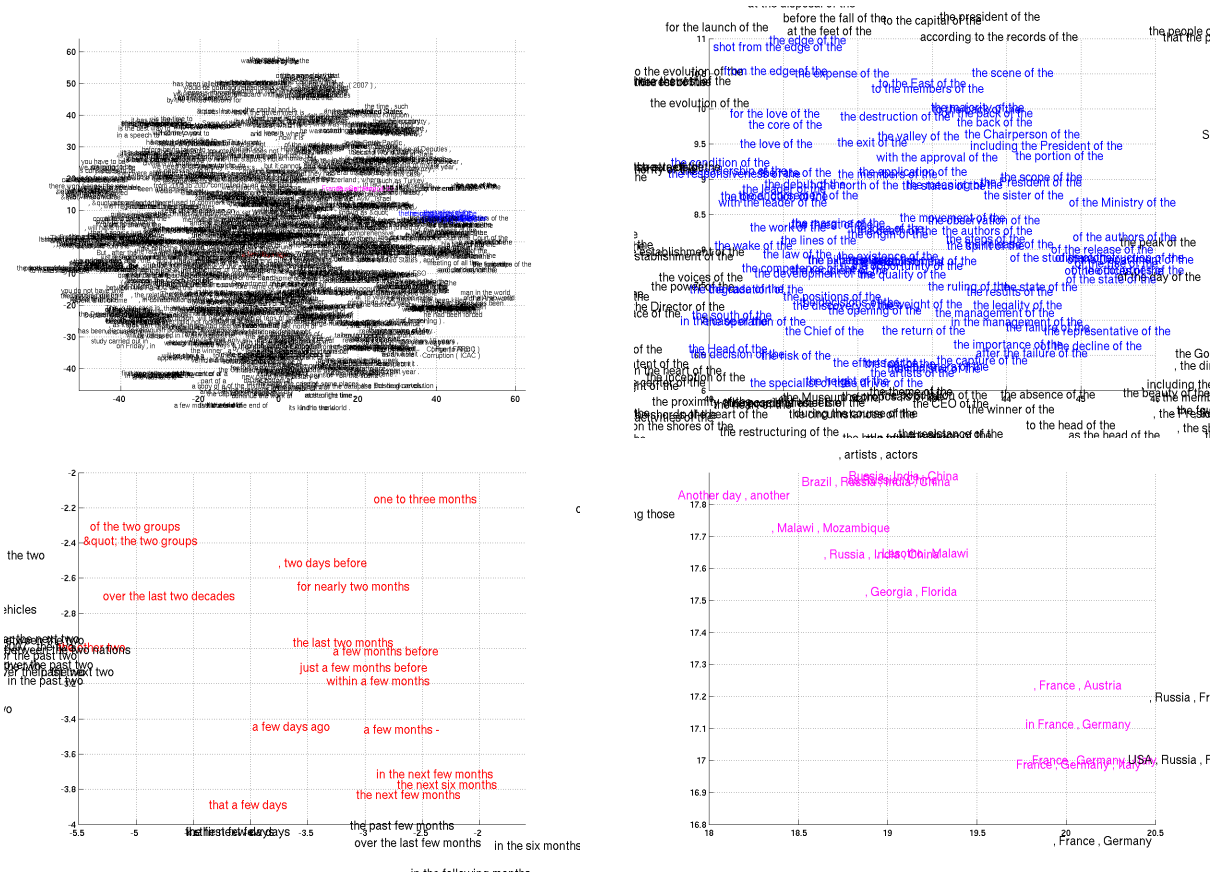


Figure 5: 2–D embedding of the learned phrase representation. The top left one shows the full representation space (5000 randomly selected points), while the other three figures show the zoomed-in view of specific regions (color–coded).

with the proposed RNN Encoder–Decoder in the future.

4.4 Word and Phrase Representations

Since the proposed RNN Encoder–Decoder is not specifically designed only for the task of machine translation, here we briefly look at the properties of the trained model.

It has been known for some time that continuous space language models using neural networks are able to learn semantically meaningful embeddings (See, e.g., (Bengio et al., 2003; Mikolov et al., 2013)). Since the proposed RNN Encoder–Decoder also projects to and maps back from a sequence of words into a continuous space vector, we expect to see a similar property with the proposed model as well.

The left plot in Fig. 4 shows the 2–D embedding of the words using the word embedding matrix learned by the RNN Encoder–Decoder. The projection was done by the recently proposed Barnes-Hut-SNE (van der Maaten, 2013). We can clearly see that semantically similar words are clustered with each other (see the zoomed-in plots in Fig. 4).

The proposed RNN Encoder–Decoder naturally generates a continuous-space representation of a phrase. The representation ($\mathbf{c}$ in Fig. 1) in this case is a 1000-dimensional vector. Similarly to the word representations, we visualize the representations of the phrases that consists of four or more words using the Barnes-Hut-SNE in Fig. 5.

From the visualization, it is clear that the RNN Encoder–Decoder captures *both semantic and syntactic* structures of the phrases. For instance, in the bottom-left plot, most of the phrases are about the duration of time, while those phrases that are syntactically similar are clustered together. The bottom-right plot shows the cluster of phrases that are semantically similar (countries or regions). On the other hand, the top-right plot shows the phrases that are syntactically similar.

5 Conclusion

In this paper, we proposed a new neural network architecture, called an *RNN Encoder–Decoder* that is able to learn the mapping from a sequence

of an arbitrary length to another sequence, possibly from a different set, of an arbitrary length. The proposed RNN Encoder–Decoder is able to either score a pair of sequences (in terms of a conditional probability) or generate a target sequence given a source sequence. Along with the new architecture, we proposed a novel hidden unit that includes a reset gate and an update gate that adaptively control how much each hidden unit remembers or forgets while reading/generating a sequence.

We evaluated the proposed model with the task of statistical machine translation, where we used the RNN Encoder–Decoder to score each phrase pair in the phrase table. Qualitatively, we were able to show that the new model is able to capture linguistic regularities in the phrase pairs well and also that the RNN Encoder–Decoder is able to propose well-formed target phrases.

The scores by the RNN Encoder–Decoder were found to improve the overall translation performance in terms of BLEU scores. Also, we found that the contribution by the RNN Encoder–Decoder is rather orthogonal to the existing approach of using neural networks in the SMT system, so that we can improve further the performance by using, for instance, the RNN Encoder–Decoder and the neural net language model together.

Our qualitative analysis of the trained model shows that it indeed captures the linguistic regularities in multiple levels i.e. at the word level as well as phrase level. This suggests that there may be more natural language related applications that may benefit from the proposed RNN Encoder–Decoder.

The proposed architecture has large potential for further improvement and analysis. One approach that was not investigated here is to replace the whole, or a part of the phrase table by letting the RNN Encoder–Decoder propose target phrases. Also, noting that the proposed model is not limited to being used with written language, it will be an important future research to apply the proposed architecture to other applications such as speech transcription.

Acknowledgments

KC, BM, CG, DB and YB would like to thank NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs and CIFAR. FB and HS were partially funded by the European Commission under the project MateCat, and by DARPA under the BOLT project.

References

[Auli et al.2013] Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1044–1054.

[Axelrod et al.2011] Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 355–362.

[Bastien et al.2012] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

[Bengio et al.2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

[Bengio et al.2013] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. 2013. Advances in optimizing recurrent networks. In *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013)*, May.

[Bergstra et al.2010] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June. Oral Presentation.

[Chandar et al.2014] Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. *arXiv:*`1402.1454 [cs.CL]`, February.

[Dahl et al.2012] George E. Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):33–42.

[Devlin et al.2014] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, , and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the ACL 2014 Conference*, ACL '14, pages 1370–1380.

[Gao et al.2013] Jianfeng Gao, Xiaodong He, Wen tau Yih, and Li Deng. 2013. Learning semantic representations for the phrase translation model. Technical report, Microsoft Research.

[Glorot et al.2011] X. Glorot, A. Bordes, and Y. Bengio. 2011. Deep sparse rectifier neural networks. In *AISTATS'2011*.

[Goodfellow et al.2013] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *ICML'2013*.

[Graves2012] Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in Computational Intelligence. Springer.

[Hochreiter and Schmidhuber1997] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[Kalchbrenner and Blunsom2013] Nal Kalchbrenner and Phil Blunsom. 2013. Two recurrent continuous translation models. In *Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1700–1709.

[Koehn et al.2003] Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54.

[Koehn2005] P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Machine Translation Summit X*, pages 79–86, Phuket, Thailand.

[Krizhevsky et al.2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS'2012)*.

[Marcu and Wong2002] Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 133–139.

[Mikolov et al.2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

[Moore and Lewis2010] Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 220–224, Stroudsburg, PA, USA.

[Pascanu et al.2014] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. 2014. How to construct deep recurrent neural networks. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, April.

[Saxe et al.2014] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. 2014. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, April.

[Schwenk et al.2006] Holger Schwenk, Marta R. Costa-Jussà, and José A. R. Fonollosa. 2006. Continuous space language models for the iwslt 2006 task. In *IWSLT*, pages 166–173.

[Schwenk2007] Holger Schwenk. 2007. Continuous space language models. *Comput. Speech Lang.*, 21(3):492–518, July.

[Schwenk2012] Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In Martin Kay and Christian Boitet, editors, *Proceedings of the 24th International Conference on Computational Linguistics (COLIN)*, pages 1071–1080.

[Socher et al.2011] Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems 24*.

[Son et al.2012] Le Hai Son, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 39–48, Stroudsburg, PA, USA.

[van der Maaten2013] Laurens van der Maaten. 2013. Barnes-hut-sne. In *Proceedings of the First International Conference on Learning Representations (ICLR 2013)*, May.

[Vaswani et al.2013] Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392.

[Zeiler2012] Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. Technical report, arXiv 1212.5701.

[Zou et al.2013] Will Y. Zou, Richard Socher, Daniel M. Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1393–1398.

Type-based MCMC for Sampling Tree Fragments from Forests

Xiaochang Peng and **Daniel Gildea**
Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

This paper applies type-based Markov Chain Monte Carlo (MCMC) algorithms to the problem of learning Synchronous Context-Free Grammar (SCFG) rules from a forest that represents all possible rules consistent with a fixed word alignment. While type-based MCMC has been shown to be effective in a number of NLP applications, our setting, where the tree structure of the sentence is itself a hidden variable, presents a number of challenges to type-based inference. We describe methods for defining variable types and efficiently indexing variables in order to overcome these challenges. These methods lead to improvements in both log likelihood and BLEU score in our experiments.

1 Introduction

In previous work, sampling methods have been used to learn Tree Substitution Grammar (TSG) rules from derivation trees (Post and Gildea, 2009; Cohn et al., 2009) for TSG learning. Here, at each node in the derivation tree, there is a binary variable indicating whether the node is internal to a TSG rule or is a split point, which we refer to as a **cut**, between two rules. The problem of extracting machine translation rules from word-aligned bitext is a similar problem in that we wish to automatically learn the best granularity for the rules with which to analyze each sentence. The problem of rule extraction is more complex, however, because the tree structure of the sentence is also unknown.

In machine translation applications, most previous work on joint alignment and rule extraction models uses heuristic methods to extract rules from learned word alignment or bracketing structures (Zhang et al., 2008; Blunsom et al., 2009; DeNero et al., 2008; Levenberg et al., 2012). Chung et al. (2014) present a MCMC algorithm schedule to learn Hiero-style SCFG rules (Chiang, 2007) by sampling tree fragments from phrase decomposition forests, which represent all possible rules that are consistent with a set of fixed word alignments. Assuming fixed word alignments reduces the complexity of the sampling problem, and has generally been effective in most state-of-the-art machine translation systems. The algorithm for sampling rules from a forest is as follows: from the root of the phrase decomposition forest, one samples a cut variable, denoting whether the current node is a cut, and an edge variable, denoting which incoming hyperedge is chosen, at each node of the current tree in a top-down manner. This sampling schedule is efficient in that it only samples the current tree and will not waste time on updating variables that are unlikely to be used in any tree.

As with many other token-based Gibbs Sampling applications, sampling one node at a time can result in slow mixing due to the strong coupling between variables. One general remedy is to sample blocks of coupled variables. Cohn and Blunsom (2010) and Yamangil and Shieber (2013) used blocked sampling algorithms that sample the whole tree structure associated with one sentence at a time for TSG and TAG learning. However, this kind of blocking does not deal with the coupling of variables correlated with the same type of structure across sentences. Liang et al. (2010) introduced a type-based sampling schedule which updates a block of variables of the same type jointly. The **type** of a variable is defined as the combination of new structural choices added when assigning different values to the variable. Type-based MCMC tackles the coupling issue by assigning the same type to variables that are strongly coupled.

In this paper, we follow the phrase decomposition forest construction procedures of Chung et

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1735–1745,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

al. (2014) and present a type-based MCMC algorithm for sampling tree fragments from phrase decomposition forests which samples the variables of the same type jointly. We define the type of the cut variable for each node in our sampling schedule. While type-based MCMC has been proven to be effective in a number of NLP applications, our sample-edge, sample-cut setting is more complicated as our tree structure is unknown. We need additional steps to maintain the cut type information when the tree structure is changed as we sample the edge variable. Like other type-based MCMC applications, we need bookkeeping of node sites to be sampled in order to loop through sites of the same type efficiently. As noted by Liang et al. (2010), indexing by the complete type information is too expensive in some applications like TSG learning. Our setting is different from TSG learning in that the internal structure of each SCFG rule is abstracted away when deriving the rule type from the tree fragment sampled.

We make the following contributions:

1. We apply type-based MCMC to the setting of SCFG learning and have achieved better log likelihood and BLEU score result.

2. We present an innovative way of storing the type information by indexing on partial type information and then filtering the retrieved nodes according to the full type information, which enables efficient updates to maintain the type information while the amount of bookkeeping is reduced significantly.

3. We replace the two-stage sampling schedule of Liang et al. (2010) with a simpler and faster one-stage method.

4. We use parallel programming to do inexact type-based MCMC, which leads to a speed up of four times in comparison with non-parallel type-based MCMC, while the likelihood result of the Markov Chain does not change. This strategy should also work with other type-based MCMC applications.

2 MCMC for Sampling Tree Fragments from Forests

2.1 Phrase Decomposition Forest

The phrase decomposition forest provides a compact representation of all machine translation rules

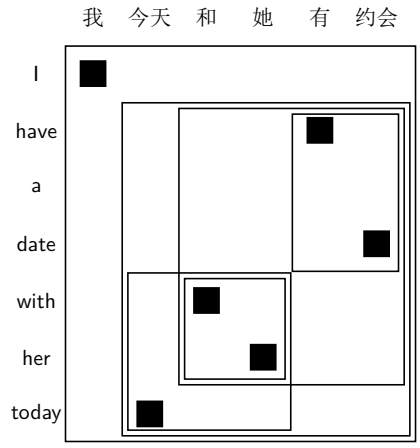


Figure 1: Example word alignment, with boxes showing valid phrase pairs. In this example, all individual alignment points are also valid phrase pairs.

that are consistent with our fixed input word alignment (Chung et al., 2014), and our sampling algorithm selects trees from this forest.

As in Hiero, our grammars will make use of a single nonterminal X, and will contain rules with a mixture of nonterminals and terminals on the righthand side (r.h.s.), with at most two nonterminal occurrences on the r.h.s. Under this restriction, the maximum number of rules that can be extracted from an input sentence pair is $O(n^{12})$ with respect to the length of the sentence pair, as the left and right boundaries of the lefthand side (l.h.s.) nonterminal and each of the two r.h.s. nonterminals can take $O(n)$ positions in each of the two languages. This complexity leads us to explore sampling algorithms instead of using dynamic programming.

A **span** $[i, j]$ is a set of contiguous word indices $\{i, i+1, \ldots, j-1\}$. Given an aligned Chinese-English sentence pair, a **phrase** n is a pair of spans $n = ([i_1, j_1], [i_2, j_2])$ such that Chinese words in positions $[i_1, j_1]$ are aligned only to English words in positions $[i_2, j_2]$, and vice versa. A **phrase forest** $H = \langle V, E \rangle$ is a hypergraph made of a set of hypernodes V and a set of hyperedges E. Each node $n = ([i_1, j_1], [i_2, j_2]) \in V$ is a **tight** phrase as defined by Koehn et al. (2003), i.e., a phrase containing no unaligned words at its boundaries. A phrase $n = ([i_1, j_1], [i_2, j_2])$ **covers** $n' = ([i'_1, j'_1], [i'_2, j'_2])$ if

$$i_1 \leq i'_1 \wedge j'_1 \leq j_1 \wedge i_2 \leq i'_2 \wedge j'_2 \leq j_2$$

Each edge in E, written as $T \rightarrow n$, is made of a

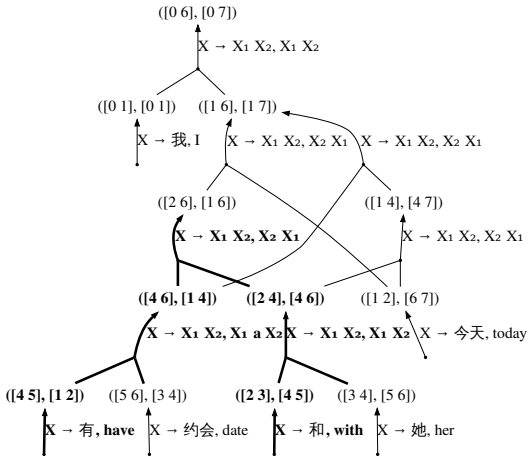


Figure 2: A phrase decomposition forest extracted from the sentence pair ⟨我 今天 和 她 有 约会, I have a date with her today⟩. Each edge is a minimal SCFG rule, and the rules at the bottom level are phrase pairs. Unaligned word "a" shows up in the rule $X \rightarrow X_1X_2, X_1aX_2$ after unaligned words are put back into the alignment matrix. The highlighted portion of the forest shows an SCFG rule built by composing minimal rules.

set of non-intersecting tail nodes $T \subset V$, and a single head node $n \in V$ that covers each tail node. We say an edge $T \rightarrow n$ is **minimal** if there does not exist another edge $T' \rightarrow n$ such that T' covers T. A **minimal edge** is an SCFG rule that cannot be decomposed by factoring out some part of its r.h.s. as a separate rule. We define a **phrase decomposition forest** to be made of all phrases from a sentence pair, connected by all minimal SCFG rules. A phrase decomposition forest compactly represents all possible SCFG rules that are consistent with word alignments. For the example word alignment shown in Figure 1, the phrase decomposition forest is shown in Figure 2. Each boxed phrase in Figure 1 corresponds to a node in the forest of Figure 2, while hyperedges in Figure 2 represent ways of building phrases out of shorter phrases.

A phrase decomposition forest has the important property that any SCFG rule consistent with the word alignment corresponds to a contiguous fragment of some complete tree found in the forest. For example, the highlighted tree fragment of the forest in Figure 2 corresponds to the SCFG rule:

$$X \rightarrow \text{和 } X_2 \text{ 有 } X_1, \text{ have a } X_1 \text{ with } X_2$$

Thus any valid SCFG rule can be formed by selecting a set of adjacent hyperedges from the forest and composing the minimal SCFG rules specified by each hyperedge. Therefore, the problem of SCFG rules extraction can be solved by sampling tree fragments from the phrase decomposition forest. We use a bottom-up algorithm to construct the phrase decomposition forest from the word alignments.

2.2 Sampling Tree Fragments From Forest

We formulate the rule sampling procedure into two phases: first we select a tree from a forest, then we select the cuts in the tree to denote the split points between fragments, with each fragment corresponding to a SCFG rule. A tree can be specified by attaching a variable e_n to each node n in the forest, indicating which hyperedge is turned on at the current node. Thus each assignment will specify a unique tree by tracing the edge variables from the root down to the leaves. We also attach a cut variable z_n to each node, indicating whether the node is a split point between two adjacent fragments.

Let all the edge variables form the random vector Y and all the cut variables form the random vector Z. Given an assignment y to the edge variables and assignment z to the cut variables, our desired distribution is proportional to the product of weights of the rules specified by the assignment:

$$P_t(Y = y, Z = z) \propto \prod_{r \in \tau(y,z)} w(r) \qquad (1)$$

where $\tau(y, z)$ is the set of rules identified by the assignment. We use a generative model based on a Dirichlet Process (DP) defined over composed rules. We draw a distribution G over rules from a DP, and then rules from G.

$$G \mid \alpha, P_0 \sim Dir(\alpha, P_0)$$
$$r \mid G \sim G$$

For the base distribution P_0, we use a uniform distribution where all rules of the same size have equal probability:

$$P_0(r) = V_f^{-|r_f|} V_e^{-|r_e|} \qquad (2)$$

where V_f and V_e are the vocabulary sizes of the source language and the target language, and $|r_f|$ and $|r_e|$ are the lengths of the source side and target side of rule r. By marginalizing out G we get a simple posterior distribution over rules which can be described using the Chinese Restaurant Process (CRP). For this analogy, we imagine a restaurant has infinite number of tables that represent rule types and customers that represent translation rule instances. Each customer enters the restaurant and chooses a table to sit at. Let z_i be the table chosen by the i-th customer, then the customer chooses a table k either having been seated or a new table with probability:

$$P(z_i = k|z_{-i}) = \begin{cases} \frac{n_k}{i-1+\alpha} & 1 \leq k \leq K \\ \frac{\alpha}{i-1+\alpha} & k = K+1 \end{cases} \quad (3)$$

where z_{-i} is the current seating arrangement, n_k is the number of customers at the table k, K is the total number of occupied tables. If the customer sits at a new table, the new table will be assigned a rule label r with probability $P_0(r)$. We can see from Equation 3 that the only history related to the current table assignment is the counts in z_{-i}. Therefore, we define a table of counts $N = \{N_C\}_{C \in I}$ which memorizes different categories of counts in z_{-i}. I is an index set for different categories of counts. Each N_C is a vector of counts for category C. We have $P(r_i = r|z_{-i}) = P(r_i = r|N)$. If we marginalize over tables labeled with the same rule, we get the following probability over rule r given the previous count table N:

$$P(r_i = r|N) = \frac{N_R(r) + \alpha P_0(r)}{n + \alpha} \quad (4)$$

here in the case of DP, $I = \{R\}$, where R is the index for the category of rule counts. $N_R(r)$ is the number of times that rule r has been observed in z_{-i}, $n = \sum_r N_R(r)$ is the total number of rules observed.

We also define a Pitman-Yor Process (PYP) (Pitman and Yor, 1997) over rules of each length l. We draw the rule distribution G from a PYP, and then rules of length l are drawn from G.

$$G|\alpha, d, P_0 \sim PY(\alpha, d, P_0)$$

$$r|G \sim G$$

The first two parameters, a concentration parameter α and a discount parameter d, control the shape of distribution G by controlling the size and the number of clusters. Integrating over G, we have the following PYP posterior probability:

$$P(r_i = r|N) = \frac{N_R(r) - T_r d + (T_l d + \alpha) P_0(r)}{N_L(l) + \alpha} \quad (5)$$

here for the case of PYP, $I = \{R, L\}$. We have an additional index L for the category of rule length counts, and $N_L(l)$ is the total number of rules of length l observed in z_{-i}. T_r is the number of tables labeled with r in z_{-i}. The length of the rule is drawn from a Poisson distribution, so a rule length probability $P(l; \lambda) = \frac{\lambda^l e^{-\lambda}}{l!}$ is multiplied by this probability to calculate the real posterior probability for each rule. In order to simplify the tedious book-keeping, we estimate the number of tables using the following equations (Huang and Renals, 2010):

$$T_r = N_R(r)^d \quad (6)$$

$$T_l = \sum_{r:|r|=l} N_R(r)^d \quad (7)$$

We use the *top-down* sampling algorithm of Chung et al. (2014) (see Algorithm 1). Starting from the root of the forest, we sample a value for the edge variable denoting which incoming hyperedge of the node is turned on in the current tree, and then we sample a cut value for the node denoting whether the node is a split point between two fragments in the tree. For each node n, we denote the composed rule type that we get when we set the cut of node n to 0 as r_1 and the two split rule types that we get when we set the cut to 1 as r_2, r_3. We sample the cut value z_i of the current node according to the posterior probability:

$$P(z_i = z|N) = \begin{cases} \frac{P(r_1|N)}{P(r_1|N)+P(r_2|N)P(r_3|N')} & \text{if } z = 0 \\ \frac{P(r_2|N)P(r_3|N')}{P(r_1|N)+P(r_2|N)P(r_3|N')} & \text{otherwise} \end{cases} \quad (8)$$

where the posterior probability $P(r_i|N)$ is according to either a DP or a PYP, and N, N' are tables of counts. In the case of DP, N, N' differ only in the rule counts of r_2, where $N'_R(r_2) = N_R(r_2) + 1$. In the case of PYP, there is an extra difference that

Algorithm 1 Top-down Sampling Algorithm

```
queue.push(root)
while queue is not empty do
    n = queue.pop()
    SAMPLEEDGE(n)
    SAMPLECUT(n)
    for each child c of node n do
        queue.push(c)
    end for
end while
```

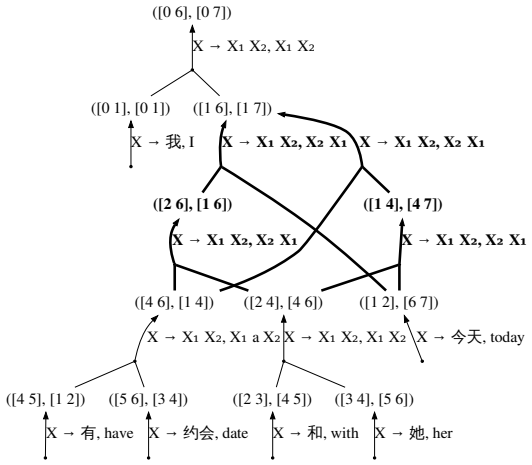


Figure 3: An example of cut type: Consider the two nodes marked in bold, $([2\,6],[1\,6])$, $([1\,4],[4\,7])$. These two non-split nodes are internal to the same composed rule: $X \rightarrow X_1X_2X_3, X_3X_2X_1$. We keep these two sites with the same index. However, when we set the cut value of these two nodes to 1, as the rules immediately above and immediately below are different for these two sites, they are not of the same type.

$N'_L(l) = N_L(l) + 1$, where l is the rule length of r_2.

As for edge variables e_i, we refer to the set of composed rules turned on below n including the composed rule fragments having n as an internal or root node as $\{r_1, \ldots, r_m\}$. We have the following posterior probability over the edge variable e_i:

$$P(e_i = e|N) \propto \prod_{i=1}^{m} P(r_i|N^{i-1}) \prod_{v \in \tau(e) \cap \text{in}(n)} \deg(v) \quad (9)$$

where $\deg(v)$ is the number of incoming edges for node v, $\text{in}(n)$ is the set of nodes in all subtrees under n, and $\tau(e)$ is the tree specified when we set $e_i = e$. N^0 to N^m are tables of counts where $N^0 = N$, $N^i_R(r_i) = N^{i-1}_R(r_i) + 1$ in the case of DP and additionally $N^i_L(l_i) = N^{i-1}_L(l_i) + 1$ in the case of PYP, where l_i is the rule length of r_i.

3 Type-based MCMC Sampling

Our goal in this paper is to organize blocks of variables that are strongly coupled into types and sample variables of each type jointly. One major property of type-based MCMC is that the joint probability of variables of the same type should be exchangeable so that the order of the variables does not matter. Also, the choices of the variables to be sampled jointly should not interfere with each other, which we define as a *conflict*. In this section, we define the type of cut variables in our sampling schedule and explain that with the two priors we introduced before, the joint probability of the variables will satisfy the exchangeability property. We will also discuss how to check conflict sites in our application.

In type-based MCMC, we need bookkeeping of sites as we need to loop through them to search for sites having the same type efficiently. In our two-stage sample-edge, sample-cut schedule, updating the edge variable would change the tree structure and trigger updates for the cut variable types in both the old and the new subtree. We come up with an efficient bookkeeping strategy to index on partial type information which significantly reduces the bookkeeping size, while updates are quite efficient when the tree structure is changed. The detail will become clear below.

3.1 Type-based MCMC

We refer to each node site to be sampled as a pair (t, n), indicating node n of forest t. For each site (t, n) and the corresponding composed rule types r_1 obtained when we set n's cut value to 0 and r_2, r_3 obtained when we set the cut value to 1, the cut variable type of site (t, n) is:

$$type(t, n) \stackrel{\text{def}}{=} (r_1, r_2, r_3)$$

We say that the cut variables of two sites are of the same type if the composed rule types r_1, r_2 and r_3 are exactly the same. For example, in Figure 3, assume that all the nodes in the hypergraph are currently set to be split points except for the two nodes marked in bold, $([2\,6],[1\,6]),([1\,4],[4\,7])$. Considering these two non-split nodes, the composed rule types they are internal to (r_1) are exactly the same. However, the situation changes if we set the cut variables of these two nodes to be 1, i.e., all of the nodes in the hypergraph are now split points. As the rule type immediately above and the rule type immediately below the two nodes (r_2 and r_3) are now different, they are not of the same type.

We sample the cut value z_i according to Equation 8. As each rule is sampled according to a DP or PYP posterior and the joint probabilities according to both posteriors are exchangeable, we can see from Equation 8 that the joint prob-

ability of a sequence of cut variables is also exchangeable. Consider a set of sites S containing n cut variables $z_S = (z_1, ..., z_n)$ of the same type. This exchangeability property leads to the fact that any sequence containing same number of cuts (cut value of 1) would have same probability. We have the following probability distribution:

$$P(z_S|N) \propto \prod_{i=1}^{n-m} P(r_1|N^{i-1}) \prod_{i=1}^{m} P(r_2|\bar{N}^{i-1})P(r_3|\hat{N}^{i-1}) \stackrel{\text{def}}{=} g(m) \quad (10)$$

where N is the count table for all the other variables except for S. $m = \sum_{i=1}^{n} z_i$ is the number of cut sites. The variables N, $\bar{N}$, and $\hat{N}$ keep track of the counts as the derivation proceeds step by step:

$$\begin{aligned} N^0 &= N \\ N_R^i(r_1) &= N_R^{i-1}(r_1) + 1 \\ \bar{N}^0 &= N^{n-m} \\ \hat{N}_R^{i-1}(r_2) &= \bar{N}_R^{i-1}(r_2) + 1 \\ \bar{N}_R^i(r_3) &= \hat{N}_R^{i-1}(r_3) + 1 \end{aligned}$$

For PYP, we add extra count indices for rule length counts similarly.

Given the exchangeability property of the cut variables, we can calculate the posterior probability of $m = \sum_{i=1}^{n} z_i$ by summing over all $\binom{n}{m}$ combinations of the cut sites:

$$p(m|N) \propto \sum_{z_S: m=\sum_i z_i} p(z_S|N) = \binom{n}{m} g(m) \quad (11)$$

3.2 Sampling Cut-types

Given Equation 11 and the exchangeability property, our sampling strategy falls out naturally: first we sample m according to Equation 11, then conditioned on m, we pick m sites of z_S as cut sites out of the $\binom{n}{m}$ combinations with uniform probability.

Now we proceed to define **conflict** sites. In addition to exchangeability, another important property of type-based MCMC is that the type of each site to be sampled should be independent of the assignment of the other sites sampled at the same time. That is, in our case, setting the cut value of each site should not change the (r_1, r_2, r_3) triple of another site. We can see that the cut value of the current site would have effect on and only on the nodes in the r_1 fragment. We denote $nodes(r)$ as the node set for all nodes within fragment r. Then for $\forall z, z' \in S$, z is not in conflict with z' if and only if $nodes(r_1) \cap nodes(r_1') = \emptyset$, where r_1 and r_1' are the corresponding composed rule types when we set z, z' to 0.

Another crucial issue in type-based sampling is the bookkeeping of sampling sites, as we need to loop through all sites having the same type with the current node. We only maintain the type information of nodes that are currently turned on in the chosen tree of the forest, as we only sample these nodes. It is common practice to directly use the type value of each variable as an index and maintain a set of sites for each type. However, maintaining a (r_1, r_2, r_3) triple for each node in the chosen tree is too memory heavy in our appli-

Algorithm 2 Type-based MCMC Algorithm for Sampling One Site

```
sample one type of sites, currently sample site (node, parent)
if parent is None or node is sampled then
    return
end if
old = node.cut
node.cut = 0
r_1 = composed_rule(parent)
node.cut = 1
r_2 = composed_rule(parent)
r_3 = composed_rule(node)
node.cut = old
sites =
for sites s ∈ index[r_1] do
    for sites s' in rule rooted at s do
        if s' of type (r_1, r_2, r_3) and no conflict then
            add s' to sites
        end if
    end for
end for
for sites s ∈ index[r_3] do
    if s of type (r_1, r_2, r_3) and no conflict then
        add s to sites
    end if
end for
sample m according to Equation 11
remove sites from index
uniformly choose m in sites to be cut sites.
add new cut sites to index
mark all nodes in sites as sampled
```

cation.

In our two-stage sample-edge, sample-cut schedule, there is an additional issue that we must deal with efficiently: when we have chosen a new incoming edge for the current node, we also have to update the bookkeeping index as the current tree structure is changed. Cut variable types in the old subtree will be turned off and a new subtree of variable types will be turned on. In the extreme case, when we have chosen a new incoming edge at the root node, we have chosen a new tree in the forest. So, we need to remove appearances of cut variable types in the old tree and add all cut variable types in the newly chosen tree.

Our strategy to deal with these two issues is to build a small, simple index, at the cost of some additional computation when retrieving nodes of a specified type. To be precise, we build an index from (single) rule types r to all occurrences of r in the data, where each occurrence is represented as a pointer to the root of r in the forest. Our strategy has two important differences from the standard strategy of building an index having the complete type (r_1, r_2, r_3) as the key and having every node as an entry. Specifically:

1. We index only the roots of the current rules, rather than every node, and

2. We key on a single rule type, rather than a triple of rule types.

Differences (1) and (2) both serve to keep the index small, and the dramatic savings in memory is essential to making our algorithm practical. Furthermore, difference (1) reduces the amount of work that needs to be done when an edge variable is resampled. While we must still re-index the entire subtree under the changed edge variable, we need only to re-index the roots of the current tree fragments, rather than all nodes in the subtree.

Given this indexing strategy, we now proceed to describe the process for retrieving nodes of a specified type (r_1, r_2, r_3). These nodes fall into one of two cases:

1. Internal nodes, i.e., nodes whose cut variable is currently set to 0. These nodes must be contained in a fragment of rule type r_1, and must furthermore have r_2 above them, and r_3 below them. We retrieve these nodes by looking up r_1 in the index, iterating over all nodes in each fragment retrieved, and retaining only those with r_2 above and r_3 below. (Lines 13–19 in Algorithm 2.)

2. Boundary nodes, i.e., nodes whose cut variable is currently set to 1. These nodes must form the root of a fragment r_3, and have a fragment r_2 above them. We retrieve these nodes by looking up r_3 in the index, and then checking each node retrieved to retain only those nodes with r_2 above them in the current tree. (Lines 20–24 in Algorithm 2.)

This process of winnowing down the nodes retrieved by the index adds some computational overhead to our algorithm, but we find that it is minimal in practice.

We still use the top-down sampling schedule of Algorithm 1, except that in the *sample-edge* step, when we choose a new incoming edge, we add additional steps to update the bookkeeping index. Furthermore, in the *sample-cut* step, we sample all non-conflict sites having the same type with n jointly. Our full algorithm for sampling one cut-type is shown in Algorithm 2. When sampling each site, we record a $parent$ node of the nearest cut ancestor of the current node so that we can build r_1 and r_2 more quickly, as they are both rooted at $parent$. We first identify the type of the current site. Then we search the bookkeeping index to find possible candidate sites of the same type, as described above. As for conflict checking, we keep a set of nodes that includes all nodes in the r_1 fragment of previous non-conflict sites. If the r_1 fragment of the current site has any node in common with this node set, we arrive at a conflict site.

4 Methods of Further Optimization

4.1 One-stage Sampling Schedule

Instead of calculating the posterior of each m according to Equation 11 and then sampling m, we can build our real m more greedily.

$$P(z_S|N) = \prod_{i=1}^{n} P(z_i|N^{i-1}) \qquad (12)$$

where $N, N^0, \ldots, N^n$ are count tables, and $N^0 = N$. N^i is the new count table after we update N^{i-1} according to the assignment of z_i. This equation gives us a hint to sample each z_i according to $P(z_i|N^{i-1})$ and then update the count table N^{i-1} according to the assignment of z_i. This greedy

sampling saves us the effort to calculate each m by multiplying over each posterior of cut variables but directly samples the real m. In our experiment, this one-stage sampling strategy gives us a 1.5 times overall speed up in comparison with the two-stage sampling schedule of Liang et al. (2010).

4.2 Parallel Implementation

As our type-based sampler involves tedious bookkeeping and frequent conflict checking and mismatch of cut types, one iteration of the type-based sampler is slower than an iteration of the token-based sampler when run on a single processor. In order to speed up our sampling procedure, we used a parallel sampling strategy similar to that of Blunsom et al. (2009) and Feng and Cohn (2013), who use multiple processors to perform inexact Gibbs Sampling, and find equivalent performance in comparison with an exact Gibbs Sampler with significant speed up. In our application, we split the data into several subsets and assign each subset to a processor. Each processor performs type-based sampling on its subset using local counts and local bookkeeping, and communicates the update of the local counts after each iteration. All the updates are then aggregated to generate global counts and then we refresh the local counts of each processor. We do not communicate the update on the bookkeeping of each processor. In this implementation, we have a slightly "out-of-date" counts at each processor and a smaller bookkeeping of sites of the same type, but we can perform type-based sampling independently on each processor. Our experiments show that, with proper division of the dataset, the final performance does not change, while the speed up is significant.

5 Experiments

We used the same LDC Chinese-English parallel corpus as Chung et al. (2014),[1] which is composed of newswire text. The corpus consists of 41K sentence pairs, which has 1M words on the English side. The corpus has a 392-sentence development set with four references for parameter tuning, and a 428-sentence test set with four references for testing.[2] The development set and the test set have sentences with less than 30 words. A trigram language model was used for all experiments. We plotted the log likelihood graph to compare the convergence property of each sampling schedule and calculated BLEU (Papineni et al., 2002) for evaluation.

5.1 Experiment Settings

We use the top-down token-based sampling algorithm of Chung et al. (2014) as our baseline. We use the same SCFG decoder for translation with both the baseline and the grammars sampled using our type-based MCMC sampler. The features included in our experiments are differently normalized rule counts and lexical weightings (Koehn et al., 2003) of each rule. Weights are tuned using Pairwise Ranking Optimization (Hopkins and May, 2011) using a grammar extracted by the standard heuristic method (Chiang, 2007) and the development set. The same weights are used throughout our experiments.

First we want to compare the DP likelihood of the baseline with our type-based MCMC sampler to see if type-based sampling would converge to a better sampling result. In order to verify if type-based MCMC really converges to a good optimum point, we use simulated annealing (Kirkpatrick et al., 1983) to search possible better optimum points. We sample from the real distribution modified by an annealing parameter β:

$$z \sim P(z)^{\beta}$$

We increase our β from 0.1 to 1.3, and then decrease from 1.3 to 1.0, changing by 0.1 every 3 iterations. We also run an inexact parallel approximation of type-based MCMC in comparison with the non-parallel sampling to find out if parallel programming is feasible to speed up type-based MCMC sampling without affecting the performance greatly. We do not compare the PYP likelihood because the approximation renders it impossible to calculate the real PYP likelihood. We also calculate the BLEU score to compare the grammars extracted using each sampling schedule. We just report the BLEU result of grammars sampled using PYP as for all our schedules, since PYP always performs better than DP.

[1] The data are randomly sampled from various different sources (LDC2006E86, LDC2006E93, LDC2002E18, LDC2002L27, LDC2003E07, LDC2003E14, LDC2004T08, LDC2005T06, LDC2005T10, LDC2005T34, LDC2006E26, LDC2005E83, LDC2006E34, LDC2006E85, LDC2006E92, LDC2006E24, LDC2006E92, LDC2006E24) The language model is trained on the English side of entire data (1.65M sentences, which is 39.3M words.)

[2] They are from newswire portion of NIST MT evaluation data from 2004, 2005, and 2006.

As for parameter settings, we used $d = 0.5$ for the Pitman-Yor discount parameter. Though we have a separate PYP for each rule length, we used same $\alpha = 5$ for all rule sizes in all experiments, including experiments using DP. For rule length probability, a Poisson distribution where $\lambda = 2$ was used for all experiments.[3]

For each sentence sample, we initialize all the nodes in the forest to be cut sites and choose an incoming edge for each node uniformly. For each experiment, we run for 160 iterations. For each DP experiment, we draw the log likelihood graph for each sampling schedule before it finally converges. For each PYP experiment, we tried averaging the grammars from every 10th iteration to construct a single grammar and use this grammar for decoding. We tune the number of grammars included for averaging by comparing the BLEU score on the dev set and report the BLEU score result on the test with the same averaging of grammars.

As each tree fragment sampled from the forest represents a unique translation rule, we do not need to explicitly extract the rules; we merely need to collect them and count them. However, the fragments sampled include purely non-lexical rules that do not conform to the rule constraints of Hiero, and rules that are not useful for translation. In order to get rid of this type of rule, we prune every rule that has **scope** (Hopkins and Langmead, 2010) greater than two. Whereas Hiero does not allow two adjacent nonterminals in the source side, our pruning criterion allows some rules of scope two that are not allowed by Hiero. For example, the following rule (only source side shown) has scope two but is not allowed by Hiero:

$$X \rightarrow w_1 X_1 X_2 w_2 X_3$$

5.2 Experiment Results

Figure 4 shows the log likelihood result of our type-based MCMC sampling schedule and the baseline top-down sampling. We can see that type-based sampling converges to a much better result than non-type-based top-down sampling. This shows that type-based MCMC escapes some local optima that are hard for token-based methods to escape. This further strengthens the idea that sampling a block of strongly coupled variables jointly

[3]The priors are the same as the work of Chung et al. (2014). The priors are set to be the same because other priors turn out not to affect much of the final performance and add additional difficulty for tuning.

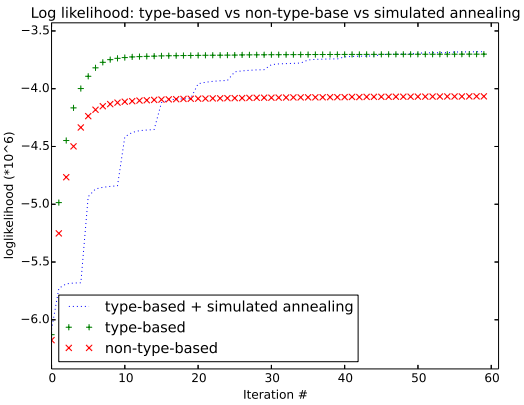


Figure 4: Log likelihood result of type-based MCMC sampling against non-type-based MCMC sampling, simulated annealing is used to verify if type-based MCMC converges to a good likelihood

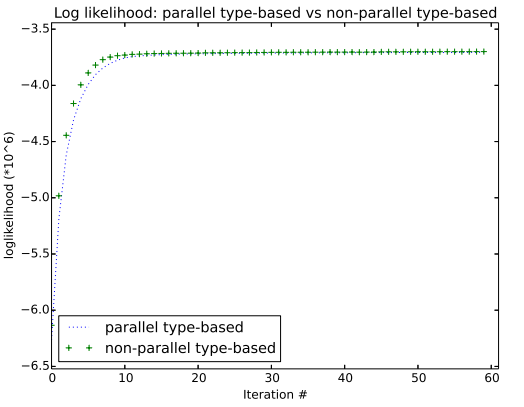


Figure 5: parallelization result for type-based MCMC

helps solve the slow mixing problem of token-based sampling methods. Another interesting observation is that, even though theoretically these two sampling methods should finally converge to the same point, in practice a worse sampling algorithm is prone to get trapped at local optima, and it will be hard for its Markov chain to escape it. We can also see from Figure 4 that the log likelihood result only improves slightly using simulated annealing. One possible explanation is that the Markov chain has already converged to a very good optimum point with type-based sampling and it is hard to search for a better optimum.

Figure 5 shows the parallelization result of type-based MCMC sampling when we run the program on five processors. We can see from the graph that when running on five processors, the likelihood fi-

Sampling Schedule	iteration	dev	test
Non-type-based	averaged (0-90)	25.62	24.98
Type-based	averaged (0-100)	25.88	25.20
Parallel Type-based	averaged (0-90)	25.75	25.04

Table 1: Comparisons of BLEU score results

nally converges to the same likelihood result as non-parallel type-based MCMC sampling. However, when we use more processors, the likelihood eventually becomes lower than with non-parallel sampling. This is because when we increase the number of processors, we split the dataset into very small subsets. As we maintain the bookkeeping for each subset separately and do not communicate the updates to each subset, the power of type-based sampling is weakened with bookkeeping for very few sites of each type. In the extreme case, when we use too many processors in parallel, the bookkeeping would have a singleton site for each type. In this case, the approximation would degrade to the scenario of approximating token-based sampling. By choosing a proper size of division of the dataset and by maintaining local bookkeeping for each subset, the parallel approximation can converge to almost the same point as non-parallel sampling. As shown in our experimental results, the speed up is very significant with the running time decreasing from thirty minutes per iteration to just seven minutes when running on five processors. Part of the speed up comes from the smaller bookkeeping since with fewer sites for each index, there is less mismatch or conflict of sites.

Table 1 shows the BLEU score results for type-based MCMC and the baseline. For non-type-based top-down sampling, the best BLEU score result on dev is achieved when averaging the grammars of every 10th iteration from the 0th to the 90th iteration, while our type-based method gets the best result by averaging over every 10th iteration from the 0th to the 100th iteration. We can see that the BLEU score on dev for type-based MCMC and the corresponding BLEU score on test are both better than the result for the non-type-based method, though not significantly. This shows that the better likelihood of our Markov Chain using type-based MCMC does result in better translation.

We have also done experiments calculating the BLEU score result of the inexact parallel implementation. We can see from Table 1 that, while the likelihood of the approximation does not change in comparison with the exact type-based MCMC, there is a gap between the BLEU score results. We think this difference might come from the inconsistency of the grammars sampled by each processor within each iteration, as they do not communicate the update within each iteration.

6 Conclusion

We presented a novel type-based MCMC algorithm for sampling tree fragments from phrase decomposition forests. While the hidden tree structure in our settings makes it difficult to maintain the constantly changing type information, we have come up with a compact way to store the type information of variables and proposed efficient ways to update the bookkeeping index. Under the additional hidden structure limitation, we have shown that type-based MCMC sampling still works and results in both better likelihood and BLEU score. We also came with techniques to speed up the type-based MCMC sampling schedule while not affecting the final sampling likelihood result. A remaining issue with parallelization is the inconsistency of the grammar within an iteration between processors. One possible solution would be using better averaging methods instead of simply averaging over every few iterations. Another interesting extension for our methods would be to also define types for the edge variables, and then sample both cut and edge types jointly.

Acknowledgments

We gratefully acknowledge the assistance of Licheng Fang and Tagyoung Chung. This work was partially funded by NSF grant IIS-0910611.

References

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A Gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*, pages 782–790, Singapore.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Tagyoung Chung, Licheng Fang, Daniel Gildea, and Daniel Štefankovič. 2014. Sampling tree fragments from forests. *Computational Linguistics*, 40:203–229.

Trevor Cohn and Phil Blunsom. 2010. Blocked inference in Bayesian tree substitution grammars. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 225–230, Uppsala, Sweden.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 548–556, Boulder, Colorado, June. Association for Computational Linguistics.

John DeNero, Alexandre Bouchard-Cote, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *Proceedings of EMNLP*, pages 314–323, Honolulu, HI.

Yang Feng and Trevor Cohn. 2013. A markov model of machine translation using non-parametric bayesian inference. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 333–342, Sofia, Bulgaria, August. Association for Computational Linguistics.

Mark Hopkins and Greg Langmead. 2010. SCFG decoding without binarization. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 646–655, Cambridge, MA, October.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July.

Songfang Huang and Steve Renals. 2010. Power law discounting for n-gram language models. In *Proc. IEEE International Conference on Acoustic, Speech, and Signal Processing (ICASSP'10)*, pages 5178–5181, Dallas, Texas, USA.

Scott Kirkpatrick, C. D. Gelatt, Jr., and Mario P. Vecchi. 1983. Optimization by Simulated Annealing. *Science*, 220(4598):671–680.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL-03*, pages 48–54, Edmonton, Alberta.

Abby Levenberg, Chris Dyer, and Phil Blunsom. 2012. A Bayesian model for learning SCFGs with discontiguous rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 223–232, Jeju Island, Korea.

Percy Liang, Michael I Jordan, and Dan Klein. 2010. Type-based mcmc. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 573–581. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL-02*, pages 311–318, Philadelphia, PA.

Jim Pitman and Marc Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25(2):855–900.

Matt Post and Daniel Gildea. 2009. Bayesian learning of a tree substitution grammar. In *Proc. Association for Computational Linguistics (short paper)*, pages 45–48, Singapore.

Elif Yamangil and Stuart M Shieber. 2013. Non-parametric bayesian inference and efficient parsing for tree-adjoining grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Association of Computational Linguistics.

Hao Zhang, Daniel Gildea, and David Chiang. 2008. Extracting synchronous grammar rules from word-level alignments in linear time. In *COLING-08*, pages 1081–1088, Manchester, UK.

Convolutional Neural Networks for Sentence Classification

Yoon Kim
New York University
yhk255@nyu.edu

Abstract

We report on a series of experiments with convolutional neural networks (CNN) trained on top of pre-trained word vectors for sentence-level classification tasks. We show that a simple CNN with little hyperparameter tuning and static vectors achieves excellent results on multiple benchmarks. Learning task-specific vectors through fine-tuning offers further gains in performance. We additionally propose a simple modification to the architecture to allow for the use of both task-specific and static vectors. The CNN models discussed herein improve upon the state of the art on 4 out of 7 tasks, which include sentiment analysis and question classification.

1 Introduction

Deep learning models have achieved remarkable results in computer vision (Krizhevsky et al., 2012) and speech recognition (Graves et al., 2013) in recent years. Within natural language processing, much of the work with deep learning methods has involved learning word vector representations through neural language models (Bengio et al., 2003; Yih et al., 2011; Mikolov et al., 2013) and performing composition over the learned word vectors for classification (Collobert et al., 2011). Word vectors, wherein words are projected from a sparse, 1-of-V encoding (here V is the vocabulary size) onto a lower dimensional vector space via a hidden layer, are essentially feature extractors that encode semantic features of words in their dimensions. In such dense representations, semantically close words are likewise close—in euclidean or cosine distance—in the lower dimensional vector space.

Convolutional neural networks (CNN) utilize layers with convolving filters that are applied to local features (LeCun et al., 1998). Originally invented for computer vision, CNN models have subsequently been shown to be effective for NLP and have achieved excellent results in semantic parsing (Yih et al., 2014), search query retrieval (Shen et al., 2014), sentence modeling (Kalchbrenner et al., 2014), and other traditional NLP tasks (Collobert et al., 2011).

In the present work, we train a simple CNN with one layer of convolution on top of word vectors obtained from an unsupervised neural language model. These vectors were trained by Mikolov et al. (2013) on 100 billion words of Google News, and are publicly available.[1] We initially keep the word vectors static and learn only the other parameters of the model. Despite little tuning of hyperparameters, this simple model achieves excellent results on multiple benchmarks, suggesting that the pre-trained vectors are 'universal' feature extractors that can be utilized for various classification tasks. Learning task-specific vectors through fine-tuning results in further improvements. We finally describe a simple modification to the architecture to allow for the use of both pre-trained and task-specific vectors by having multiple channels.

Our work is philosophically similar to Razavian et al. (2014) which showed that for image classification, feature extractors obtained from a pre-trained deep learning model perform well on a variety of tasks—including tasks that are very different from the original task for which the feature extractors were trained.

2 Model

The model architecture, shown in figure 1, is a slight variant of the CNN architecture of Collobert et al. (2011). Let $\mathbf{x}_i \in \mathbb{R}^k$ be the k-dimensional word vector corresponding to the i-th word in the sentence. A sentence of length n (padded where

[1] https://code.google.com/p/word2vec/

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

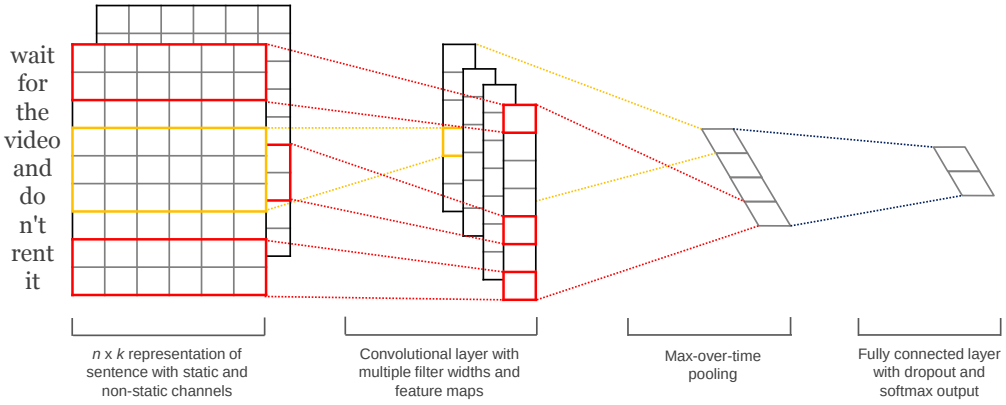


Figure 1: Model architecture with two channels for an example sentence.

necessary) is represented as

$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n, \quad (1)$$

where $\oplus$ is the concatenation operator. In general, let $\mathbf{x}_{i:i+j}$ refer to the concatenation of words $\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{i+j}$. A convolution operation involves a *filter* $\mathbf{w} \in \mathbb{R}^{hk}$, which is applied to a window of h words to produce a new feature. For example, a feature c_i is generated from a window of words $\mathbf{x}_{i:i+h-1}$ by

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b). \quad (2)$$

Here $b \in \mathbb{R}$ is a bias term and f is a non-linear function such as the hyperbolic tangent. This filter is applied to each possible window of words in the sentence $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$ to produce a *feature map*

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}], \quad (3)$$

with $\mathbf{c} \in \mathbb{R}^{n-h+1}$. We then apply a max-over-time pooling operation (Collobert et al., 2011) over the feature map and take the maximum value $\hat{c} = \max\{\mathbf{c}\}$ as the feature corresponding to this particular filter. The idea is to capture the most important feature—one with the highest value—for each feature map. This pooling scheme naturally deals with variable sentence lengths.

We have described the process by which *one* feature is extracted from *one* filter. The model uses multiple filters (with varying window sizes) to obtain multiple features. These features form the penultimate layer and are passed to a fully connected softmax layer whose output is the probability distribution over labels.

In one of the model variants, we experiment with having two 'channels' of word vectors—one that is kept static throughout training and one that is fine-tuned via backpropagation (section 3.2).[2] In the multichannel architecture, illustrated in figure 1, each filter is applied to both channels and the results are added to calculate c_i in equation (2). The model is otherwise equivalent to the single channel architecture.

2.1 Regularization

For regularization we employ dropout on the penultimate layer with a constraint on l_2-norms of the weight vectors (Hinton et al., 2012). Dropout prevents co-adaptation of hidden units by randomly dropping out—i.e., setting to zero—a proportion p of the hidden units during foward-backpropagation. That is, given the penultimate layer $\mathbf{z} = [\hat{c}_1, \ldots, \hat{c}_m]$ (note that here we have m filters), instead of using

$$y = \mathbf{w} \cdot \mathbf{z} + b \quad (4)$$

for output unit y in forward propagation, dropout uses

$$y = \mathbf{w} \cdot (\mathbf{z} \circ \mathbf{r}) + b, \quad (5)$$

where $\circ$ is the element-wise multiplication operator and $\mathbf{r} \in \mathbb{R}^m$ is a 'masking' vector of Bernoulli random variables with probability p of being 1. Gradients are backpropagated only through the unmasked units. At test time, the learned weight vectors are scaled by p such that $\hat{\mathbf{w}} = p\mathbf{w}$, and $\hat{\mathbf{w}}$ is used (without dropout) to score unseen sentences. We additionally constrain l_2-norms of the weight vectors by rescaling $\mathbf{w}$ to have $||\mathbf{w}||_2 = s$ whenever $||\mathbf{w}||_2 > s$ after a gradient descent step.

[2] We employ language from computer vision where a color image has red, green, and blue channels.

Data	c	l	N	$\|V\|$	$\|V_{pre}\|$	*Test*
MR	2	20	10662	18765	16448	CV
SST-1	5	18	11855	17836	16262	2210
SST-2	2	19	9613	16185	14838	1821
Subj	2	23	10000	21323	17913	CV
TREC	6	10	5952	9592	9125	500
CR	2	19	3775	5340	5046	CV
MPQA	2	3	10606	6246	6083	CV

Table 1: Summary statistics for the datasets after tokenization. c: Number of target classes. l: Average sentence length. N: Dataset size. $|V|$: Vocabulary size. $|V_{pre}|$: Number of words present in the set of pre-trained word vectors. *Test*: Test set size (CV means there was no standard train/test split and thus 10-fold CV was used).

3 Datasets and Experimental Setup

We test our model on various benchmarks. Summary statistics of the datasets are in table 1.

- **MR**: Movie reviews with one sentence per review. Classification involves detecting positive/negative reviews (Pang and Lee, 2005).[3]
- **SST-1**: Stanford Sentiment Treebank—an extension of MR but with train/dev/test splits provided and fine-grained labels (very positive, positive, neutral, negative, very negative), re-labeled by Socher et al. (2013).[4]
- **SST-2**: Same as SST-1 but with neutral reviews removed and binary labels.
- **Subj**: Subjectivity dataset where the task is to classify a sentence as being subjective or objective (Pang and Lee, 2004).
- **TREC**: TREC question dataset—task involves classifying a question into 6 question types (whether the question is about person, location, numeric information, etc.) (Li and Roth, 2002).[5]
- **CR**: Customer reviews of various products (cameras, MP3s etc.). Task is to predict positive/negative reviews (Hu and Liu, 2004).[6]
- **MPQA**: Opinion polarity detection subtask of the MPQA dataset (Wiebe et al., 2005).[7]

3.1 Hyperparameters and Training

For all datasets we use: rectified linear units, filter windows (h) of 3, 4, 5 with 100 feature maps each, dropout rate (p) of 0.5, l_2 constraint (s) of 3, and mini-batch size of 50. These values were chosen via a grid search on the SST-2 dev set.

We do not otherwise perform any dataset-specific tuning other than early stopping on dev sets. For datasets without a standard dev set we randomly select 10% of the training data as the dev set. Training is done through stochastic gradient descent over shuffled mini-batches with the Adadelta update rule (Zeiler, 2012).

3.2 Pre-trained Word Vectors

Initializing word vectors with those obtained from an unsupervised neural language model is a popular method to improve performance in the absence of a large supervised training set (Collobert et al., 2011; Socher et al., 2011; Iyyer et al., 2014). We use the publicly available `word2vec` vectors that were trained on 100 billion words from Google News. The vectors have dimensionality of 300 and were trained using the continuous bag-of-words architecture (Mikolov et al., 2013). Words not present in the set of pre-trained words are initialized randomly.

3.3 Model Variations

We experiment with several variants of the model.

- **CNN-rand**: Our baseline model where all words are randomly initialized and then modified during training.
- **CNN-static**: A model with pre-trained vectors from `word2vec`. All words—including the unknown ones that are randomly initialized—are kept static and only the other parameters of the model are learned.
- **CNN-non-static**: Same as above but the pre-trained vectors are fine-tuned for each task.
- **CNN-multichannel**: A model with two sets of word vectors. Each set of vectors is treated as a 'channel' and each filter is applied

[3] https://www.cs.cornell.edu/people/pabo/movie-review-data/

[4] http://nlp.stanford.edu/sentiment/ Data is actually provided at the phrase-level and hence we train the model on both phrases and sentences but only score on sentences at test time, as in Socher et al. (2013), Kalchbrenner et al. (2014), and Le and Mikolov (2014). Thus the training set is an order of magnitude larger than listed in table 1.

[5] http://cogcomp.cs.illinois.edu/Data/QA/QC/

[6] http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

[7] http://www.cs.pitt.edu/mpqa/

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

Table 2: Results of our CNN models against other methods. **RAE**: Recursive Autoencoders with pre-trained word vectors from Wikipedia (Socher et al., 2011). **MV-RNN**: Matrix-Vector Recursive Neural Network with parse trees (Socher et al., 2012). **RNTN**: Recursive Neural Tensor Network with tensor-based feature function and parse trees (Socher et al., 2013). **DCNN**: Dynamic Convolutional Neural Network with k-max pooling (Kalchbrenner et al., 2014). **Paragraph-Vec**: Logistic regression on top of paragraph vectors (Le and Mikolov, 2014). **CCAE**: Combinatorial Category Autoencoders with combinatorial category grammar operators (Hermann and Blunsom, 2013). **Sent-Parser**: Sentiment analysis-specific parser (Dong et al., 2014). **NBSVM, MNB**: Naive Bayes SVM and Multinomial Naive Bayes with uni-bigrams from Wang and Manning (2012). **G-Dropout, F-Dropout**: Gaussian Dropout and Fast Dropout from Wang and Manning (2013). **Tree-CRF**: Dependency tree with Conditional Random Fields (Nakagawa et al., 2010). **CRF-PR**: Conditional Random Fields with Posterior Regularization (Yang and Cardie, 2014). **SVM_S**: SVM with uni-bi-trigrams, wh word, head word, POS, parser, hypernyms, and 60 hand-coded rules as features from Silva et al. (2011).

to both channels, but gradients are back-propagated only through one of the channels. Hence the model is able to fine-tune one set of vectors while keeping the other static. Both channels are initialized with `word2vec`.

In order to disentangle the effect of the above variations versus other random factors, we eliminate other sources of randomness—CV-fold assignment, initialization of unknown word vectors, initialization of CNN parameters—by keeping them uniform within each dataset.

4 Results and Discussion

Results of our models against other methods are listed in table 2. Our baseline model with all randomly initialized words (CNN-rand) does not perform well on its own. While we had expected performance gains through the use of pre-trained vectors, we were surprised at the magnitude of the gains. Even a simple model with static vectors (CNN-static) performs remarkably well, giving competitive results against the more sophisticated deep learning models that utilize complex pooling schemes (Kalchbrenner et al., 2014) or require parse trees to be computed beforehand (Socher et al., 2013). These results suggest that the pre-trained vectors are good, 'universal' feature extractors and can be utilized across datasets. Fine-tuning the pre-trained vectors for each task gives still further improvements (CNN-non-static).

4.1 Multichannel vs. Single Channel Models

We had initially hoped that the multichannel architecture would prevent overfitting (by ensuring that the learned vectors do not deviate too far from the original values) and thus work better than the single channel model, especially on smaller datasets. The results, however, are mixed, and further work on regularizing the fine-tuning process is warranted. For instance, instead of using an additional channel for the non-static portion, one could maintain a single channel but employ extra dimensions that are allowed to be modified during training.

	Most Similar Words for	
	Static Channel	Non-static Channel
bad	*good*	*terrible*
	terrible	*horrible*
	horrible	*lousy*
	lousy	*stupid*
good	*great*	*nice*
	bad	*decent*
	terrific	*solid*
	decent	*terrific*
n't	*os*	*not*
	ca	*never*
	ireland	*nothing*
	wo	*neither*
!	*2,500*	*2,500*
	entire	*lush*
	jez	*beautiful*
	changer	*terrific*
,	*decasia*	*but*
	abysmally	*dragon*
	demise	*a*
	valiant	*and*

Table 3: Top 4 neighboring words—based on cosine similarity—for vectors in the static channel (left) and fine-tuned vectors in the non-static channel (right) from the multichannel model on the SST-2 dataset after training.

4.2 Static vs. Non-static Representations

As is the case with the single channel non-static model, the multichannel model is able to fine-tune the non-static channel to make it more specific to the task-at-hand. For example, *good* is most similar to *bad* in `word2vec`, presumably because they are (almost) syntactically equivalent. But for vectors in the non-static channel that were fine-tuned on the SST-2 dataset, this is no longer the case (table 3). Similarly, *good* is arguably closer to *nice* than it is to *great* for expressing sentiment, and this is indeed reflected in the learned vectors.

For (randomly initialized) tokens not in the set of pre-trained vectors, fine-tuning allows them to learn more meaningful representations: the network learns that exclamation marks are associated with effusive expressions and that commas are conjunctive (table 3).

4.3 Further Observations

We report on some further experiments and observations:

- Kalchbrenner et al. (2014) report much worse results with a CNN that has essentially the same architecture as our single channel model. For example, their Max-TDNN (Time Delay Neural Network) with randomly initialized words obtains 37.4% on the SST-1 dataset, compared to 45.0% for our model. We attribute such discrepancy to our CNN having much more capacity (multiple filter widths and feature maps).
- Dropout proved to be such a good regularizer that it was fine to use a larger than necessary network and simply let dropout regularize it. Dropout consistently added 2%–4% relative performance.
- When randomly initializing words not in `word2vec`, we obtained slight improvements by sampling each dimension from $U[-a, a]$ where a was chosen such that the randomly initialized vectors have the same variance as the pre-trained ones. It would be interesting to see if employing more sophisticated methods to mirror the distribution of pre-trained vectors in the initialization process gives further improvements.
- We briefly experimented with another set of publicly available word vectors trained by Collobert et al. (2011) on Wikipedia,[8] and found that `word2vec` gave far superior performance. It is not clear whether this is due to Mikolov et al. (2013)'s architecture or the 100 billion word Google News dataset.
- Adadelta (Zeiler, 2012) gave similar results to Adagrad (Duchi et al., 2011) but required fewer epochs.

5 Conclusion

In the present work we have described a series of experiments with convolutional neural networks built on top of `word2vec`. Despite little tuning of hyperparameters, a simple CNN with one layer of convolution performs remarkably well. Our results add to the well-established evidence that unsupervised pre-training of word vectors is an important ingredient in deep learning for NLP.

Acknowledgments

We would like to thank Yann LeCun and the anonymous reviewers for their helpful feedback and suggestions.

[8] http://ronan.collobert.com/senna/

References

Y. Bengio, R. Ducharme, P. Vincent. 2003. Neural Probabilitistic Language Model. *Journal of Machine Learning Research* 3:1137–1155.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuglu, P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research 12:2493–2537*.

J. Duchi, E. Hazan, Y. Singer. 2011 Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research,* 12:2121–2159.

L. Dong, F. Wei, S. Liu, M. Zhou, K. Xu. 2014. A Statistical Parsing Framework for Sentiment Classification. *CoRR, abs/1401.6330.*

A. Graves, A. Mohamed, G. Hinton. 2013. Speech recognition with deep recurrent neural networks. *In Proceedings of ICASSP 2013.*

G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR, abs/1207.0580.*

K. Hermann, P. Blunsom. 2013. The Role of Syntax in Vector Space Models of Compositional Semantics. *In Proceedings of ACL 2013.*

M. Hu, B. Liu. 2004. Mining and Summarizing Customer Reviews. *In Proceedings of ACM SIGKDD 2004.*

M. Iyyer, P. Enns, J. Boyd-Graber, P. Resnik 2014. Political Ideology Detection Using Recursive Neural Networks. *In Proceedings of ACL 2014.*

N. Kalchbrenner, E. Grefenstette, P. Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. *In Proceedings of ACL 2014.*

A. Krizhevsky, I. Sutskever, G. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. *In Proceedings of NIPS 2012.*

Q. Le, T. Mikolov. 2014. Distributed Represenations of Sentences and Documents. *In Proceedings of ICML 2014.*

Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. 1998. Gradient-based learning applied to document recognition. *In Proceedings of the IEEE*, 86(11):2278–2324, November.

X. Li, D. Roth. 2002. Learning Question Classifiers. *In Proceedings of ACL 2002.*

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *In Proceedings of NIPS 2013.*

T. Nakagawa, K. Inui, S. Kurohashi. 2010. Dependency tree-based sentiment classification using CRFs with hidden variables. *In Proceedings of ACL 2010.*

B. Pang, L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *In Proceedings of ACL 2004.*

B. Pang, L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *In Proceedings of ACL 2005.*

A.S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson 2014. CNN Features off-the-shelf: an Astounding Baseline. *CoRR, abs/1403.6382.*

Y. Shen, X. He, J. Gao, L. Deng, G. Mesnil. 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. *In Proceedings of WWW 2014.*

J. Silva, L. Coheur, A. Mendes, A. Wichert. 2011. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154.

R. Socher, J. Pennington, E. Huang, A. Ng, C. Manning. 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. *In Proceedings of EMNLP 2011.*

R. Socher, B. Huval, C. Manning, A. Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. *In Proceedings of EMNLP 2012.*

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, C. Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. *In Proceedings of EMNLP 2013.*

J. Wiebe, T. Wilson, C. Cardie. 2005. Annotating Expressions of Opinions and Emotions in Language. *Language Resources and Evaluation*, 39(2-3): 165–210.

S. Wang, C. Manning. 2012. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. *In Proceedings of ACL 2012.*

S. Wang, C. Manning. 2013. Fast Dropout Training. *In Proceedings of ICML 2013.*

B. Yang, C. Cardie. 2014. Context-aware Learning for Sentence-level Sentiment Analysis with Posterior Regularization. *In Proceedings of ACL 2014.*

W. Yih, K. Toutanova, J. Platt, C. Meek. 2011. Learning Discriminative Projections for Text Similarity Measures. *Proceedings of the Fifteenth Conference on Computational Natural Language Learning, 247–256.*

W. Yih, X. He, C. Meek. 2014. Semantic Parsing for Single-Relation Question Answering. *In Proceedings of ACL 2014.*

M. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR, abs/1212.5701.*

Sometimes Average is Best: The Importance of Averaging for Prediction using MCMC Inference in Topic Modeling

Viet-An Nguyen
Computer Science
University of Maryland
College Park, MD
vietan@cs.umd.edu

Jordan Boyd-Graber
Computer Science
University of Colorado
Boulder, CO
jbg@boydgraber.org

Philip Resnik
Linguistics and UMIACS
University of Maryland
College Park, MD
resnik@umd.edu

Abstract

Markov chain Monte Carlo (MCMC) approximates the posterior distribution of latent variable models by generating many samples and averaging over them. In practice, however, it is often more convenient to cut corners, using only a single sample or following a suboptimal averaging strategy. We systematically study different strategies for averaging MCMC samples and show empirically that averaging properly leads to significant improvements in prediction.

1 Introduction

Probabilistic topic models are powerful methods to uncover hidden thematic structures in text by projecting each document into a low dimensional space spanned by a set of *topics*, each of which is a distribution over words. Topic models such as latent Dirichlet allocation (Blei et al., 2003, LDA) and its extensions discover these topics from text, which allows for effective exploration, analysis, and summarization of the otherwise unstructured corpora (Blei, 2012; Blei, 2014).

In addition to exploratory data analysis, a typical goal of topic models is prediction. Given a set of unannotated training data, *unsupervised topic models* try to learn good topics that can generalize to unseen text. *Supervised topic models* jointly capture both the text and associated *metadata* such as a continuous response variable (Blei and McAuliffe, 2007; Zhu et al., 2009; Nguyen et al., 2013), single label (Rosen-Zvi et al., 2004; Lacoste-Julien et al., 2008; Wang et al., 2009) or multiple labels (Ramage et al., 2009; Ramage et al., 2011) to predict metadata from text.

Probabilistic topic modeling requires estimating the posterior distribution. Exact computation of the posterior is often intractable, which motivates approximate inference techniques (Asuncion et al., 2009). One popular approach is Markov chain Monte Carlo (MCMC), a class of inference algorithms to approximate the target posterior distribution. To make prediction, MCMC algorithms generate samples on training data to estimate corpus-level latent variables, and use them to generate samples to estimate document-level latent variables for test data. The underlying theory requires averaging on both training and test samples, but in practice it is often convenient to cut corners: either skip averaging entirely by using just the values of the last sample or use a single training sample and average over test samples.

We systematically study non-averaging and averaging strategies when performing predictions using MCMC in topic modeling (Section 2). Using popular unsupervised (LDA in Section 3) and supervised (SLDA in Section 4) topic models via thorough experimentation, we show empirically that cutting corners on averaging leads to consistently poorer prediction.

2 Learning and Predicting with MCMC

While reviewing all of MCMC is beyond the scope of this paper, we need to briefly review key concepts.[1] To estimate a target density $p(x)$ in a high-dimensional space $\mathcal{X}$, MCMC generates samples $\{x_t\}_{t=1}^{T}$ while exploring $\mathcal{X}$ using the Markov assumption. Under this assumption, sample x_{t+1} depends on sample x_t only, forming a *Markov chain*, which allows the sampler to spend more time in the most important regions of the density. Two concepts control sample collection:

Burn-in B: Depending on the initial value of the Markov chain, MCMC algorithms take time to reach the target distribution. Thus, in practice, samples before a burn-in period B are often discarded.

Sample-lag L: Averaging over samples to estimate the target distribution requires i.i.d. samples. However, future samples depend on the current samples (i.e., the Markov assumption). To avoid autocorrelation, we discard all but every L samples.

2.1 MCMC in Topic Modeling

As generative probabilistic models, topic models define a joint distribution over latent variables and observable evidence. In our setting, the latent variables consist of corpus-level *global* variables $\boldsymbol{g}$ and document-level *local* variables $\boldsymbol{l}$; while the evidence consists of words $\boldsymbol{w}$ and additional metadata $\boldsymbol{y}$—the latter omitted in unsupervised models.

During training, MCMC estimates the posterior $p(\boldsymbol{g}, \boldsymbol{l}^{\text{TR}} \mid \boldsymbol{w}^{\text{TR}}, \boldsymbol{y}^{\text{TR}})$ by generating a *training Markov chain* of T_{TR} samples.[2] Each training sample i provides a set of fully realized global latent variables $\hat{\boldsymbol{g}}(i)$, which can generate test data. During test time, given a

[1] For more details please refer to Neal (1993), Andrieu et al. (2003), Resnik and Hardisty (2010).

[2] We omit hyperparameters for clarity. We split data into training (TR) and testing (TE) folds, and denote the training iteration i and the testing iteration j within the corresponding Markov chains.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1752–1757,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

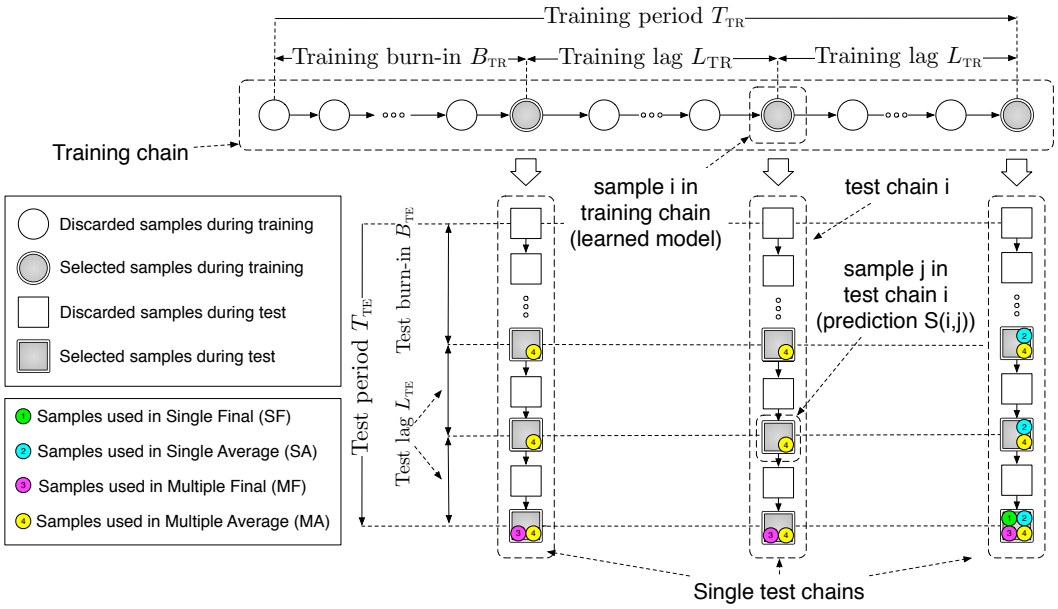


Figure 1: Illustration of training and test chains in MCMC, showing samples used in four prediction strategies studied in this paper: Single Final (SF), Single Average (SA), Multiple Final (MF), and Multiple Average (MA).

learned model from training sample i, we generate a *test Markov chain* of T_{TE} samples to estimate the local latent variables $p(\boldsymbol{l}^{\text{TE}} \mid \boldsymbol{w}^{\text{TE}}, \hat{\boldsymbol{g}}(i))$ of test data. Each sample j of test chain i provides a fully estimated local latent variables $\hat{\boldsymbol{l}}^{\text{TE}}(i,j)$ to make a prediction.

Figure 1 shows an overview. To reduce the effects of unconverged and autocorrelated samples, during training we use a burn-in period of B_{TR} and a sample-lag of L_{TR} iterations. We use $\mathcal{T}_{\text{TR}} = \{i \mid i \in (B_{\text{TR}}, T_{\text{TR}}] \wedge (i - B_{\text{TR}}) \mod L_{\text{TR}} = 0\}$ to denote the set of indices of the selected models. Similarly, B_{TE} and L_{TE} are the test burn-in and sample-lag. The set of indices of selected samples in test chains is $\mathcal{T}_{\text{TE}} = \{j \mid j \in (B_{\text{TE}}, T_{\text{TE}}] \wedge (j - B_{\text{TE}}) \mod L_{\text{TE}} = 0\}$.

2.2 Averaging Strategies

We use $S(i,j)$ to denote the prediction obtained from sample j of the test chain i. We now discuss different strategies to obtain the final prediction:

- **Single Final (SF)** uses the last sample of last test chain to obtain the predicted value,

$$S_{\text{SF}} = S(T_{\text{TR}}, T_{\text{TE}}). \tag{1}$$

- **Single Average (SA)** averages over multiple samples in the last test chain

$$S_{\text{SA}} = \frac{1}{|\mathcal{T}_{\text{TE}}|} \sum_{j \in \mathcal{T}_{\text{TE}}} S(T_{\text{TR}}, j). \tag{2}$$

This is a common averaging strategy in which we obtain a point estimate of the global latent variables at the end of the training chain. Then, a single test chain is generated on the test data and multiple samples of this test chain are averaged to obtain the final prediction (Chang, 2012; Singh et al., 2012; Jiang et al., 2012; Zhu et al., 2014).

- **Multiple Final (MF)** averages over the last samples of multiple test chains from multiple models

$$S_{\text{MF}} = \frac{1}{|\mathcal{T}_{\text{TR}}|} \sum_{i \in \mathcal{T}_{\text{TR}}} S(i, T_{\text{TE}}). \tag{3}$$

- **Multiple Average (MA)** averages over all samples of multiple test chains for distinct models,

$$S_{\text{MA}} = \frac{1}{|\mathcal{T}_{\text{TR}}|} \frac{1}{|\mathcal{T}_{\text{TE}}|} \sum_{i \in \mathcal{T}_{\text{TR}}} \sum_{j \in \mathcal{T}_{\text{TE}}} S(i,j), \tag{4}$$

3 Unsupervised Topic Models

We evaluate the predictive performance of the unsupervised topic model LDA using different averaging strategies in Section 2.

LDA: Proposed by Blei et al. in 2003, LDA posits that each document d is a multinomial distribution θ_d over K topics, each of which is a multinomial distribution ϕ_k over the vocabulary. LDA's generative process is:

1. For each topic $k \in [1, K]$
   (a) Draw word distribution $\phi_k \sim \text{Dir}(\beta)$
2. For each document $d \in [1, D]$
   (a) Draw topic distribution $\theta_d \sim \text{Dir}(\alpha)$
   (b) For each word $n \in [1, N_d]$
      i. Draw topic $z_{d,n} \sim \text{Mult}(\theta_d)$
      ii. Draw word $w_{d,n} \sim \text{Mult}(\phi_{z_{d,n}})$

In LDA, the global latent variables are topics $\{\phi_k\}_{k=1}^K$ and the local latent variables for each document d are topic proportions θ_d.

Train: During training, we use collapsed Gibbs sampling to assign each token in the training data with a topic (Steyvers and Griffiths, 2006). The probability of

assigning token n of training document d to topic k is $p(z^{\text{TR}}_{d,n} = k \mid \boldsymbol{z}^{\text{TR}}_{-d,n}, \boldsymbol{w}^{\text{TR}}_{-d,n}, w^{\text{TR}}_{d,n} = v) \propto$

$$\frac{N^{-d,n}_{\text{TR},d,k} + \alpha}{N^{-d,n}_{\text{TR},d,\cdot} + K\alpha} \cdot \frac{N^{-d,n}_{\text{TR},k,v} + \beta}{N^{-d,n}_{\text{TR},k,\cdot} + V\beta}, \tag{5}$$

where $N_{\text{TR},d,k}$ is the number of tokens in the training document d assigned to topic k, and $N_{\text{TR},k,v}$ is the number of times word type v assigned to topic k. Marginal counts are denoted by $\cdot$, and $^{-d,n}$ denotes the count excluding the assignment of token n in document d.

At each training iteration i, we estimate the distribution over words $\hat{\phi}_k(i)$ of topic k as

$$\hat{\phi}_{k,v}(i) = \frac{N_{\text{TR},k,v}(i) + \beta}{N_{\text{TR},k,\cdot}(i) + V\beta} \tag{6}$$

where the counts $N_{\text{TR},k,v}(i)$ and $N_{\text{TR},k,\cdot}(i)$ are taken at training iteration i.

Test: Because we lack explicit topic annotations for these data (c.f. Nguyen et al. (2012)), we use *perplexity*– a widely-used metric to measure the predictive power of topic models on held-old documents. To compute perplexity, we follow the *estimating* θ method (Wallach et al., 2009, Section 5.1) and evenly split each test document d into $\boldsymbol{w}^{\text{TE}_1}_d$ and $\boldsymbol{w}^{\text{TE}_2}_d$. We first run Gibbs sampling on $\boldsymbol{w}^{\text{TE}_1}_d$ to estimate the topic proportion $\hat{\theta}^{\text{TE}}_d$ of test document d. The probability of assigning topic k to token n in $\boldsymbol{w}^{\text{TE}_1}_d$ is $p(z^{\text{TE}_1}_{d,n} = k \mid \boldsymbol{z}^{\text{TE}_1}_{-d,n}, \boldsymbol{w}^{\text{TE}_1}, \hat{\phi}(i)) \propto$

$$\frac{N^{-d,n}_{\text{TE}_1,d,k} + \alpha}{N^{-d,n}_{\text{TE}_1,d,\cdot} + K\alpha} \cdot \hat{\phi}_{k,w^{\text{TE}_1}_{d,n}(i)} \tag{7}$$

where $N_{\text{TE}_1,d,k}$ is the number of tokens in $\boldsymbol{w}^{\text{TE}_1}_d$ assigned to topic k. At each iteration j in test chain i, we can estimate the topic proportion vector $\hat{\theta}^{\text{TE}}_d(i,j)$ for test document d as

$$\hat{\theta}^{\text{TE}}_{d,k}(i,j) = \frac{N_{\text{TE}_1,d,k}(i,j) + \alpha}{N_{\text{TE}_1,d,\cdot}(i,j) + K\alpha} \tag{8}$$

where both the counts $N_{\text{TE}_1,d,k}(i,j)$ and $N_{\text{TE}_1,d,\cdot}(i,j)$ are taken using sample j of test chain i.

Prediction: Given $\hat{\theta}^{\text{TE}}_d(i,j)$ and $\hat{\phi}(i)$ at sample j of test chain i, we compute the predicted likelihood for each unseen token $w^{\text{TE}_2}_{d,n}$ as $S(i,j) \equiv p(w^{\text{TE}_2}_{d,n} \mid \hat{\theta}^{\text{TE}}_d(i,j), \hat{\phi}(i)) = \sum_{k=1}^{K} \hat{\theta}^{\text{TE}}_{d,k}(i,j) \cdot \hat{\phi}_{k,w^{\text{TE}_2}_{d,n}}(i)$.

Using different strategies described in Section 2, we obtain the final predicted likelihood for each unseen token $p(w^{\text{TE}_2}_{d,n} \mid \hat{\theta}^{\text{TE}}_d, \hat{\phi})$ and compute the perplexity as $\exp\left(-(\sum_d \sum_n \log(p(w^{\text{TE}_2}_{d,n} \mid \hat{\theta}^{\text{TE}}_d, \hat{\phi})))/N^{\text{TE}_2}\right)$ where N^{TE_2} is the number of tokens in $\boldsymbol{w}^{\text{TE}_2}$.

Setup: We use three Internet review datasets in our experiment. For all datasets, we preprocess by tokenizing, removing stopwords, stemming, adding bigrams to the vocabulary, and we filter using TF-IDF to obtain a vocabulary of 10,000 words.[3] The three datasets are:

- HOTEL: 240,060 reviews of hotels from TripAdvisor (Wang et al., 2010).
- RESTAURANT: 25,459 reviews of restaurants from Yelp (Jo and Oh, 2011).
- MOVIE: 5,006 reviews of movies from Rotten Tomatoes (Pang and Lee, 2005)

We report cross-validated average performance over five folds, and use $K = 50$ topics for all datasets. To update the hyperparameters, we use slice sampling (Wallach, 2008, p. 62).[4]

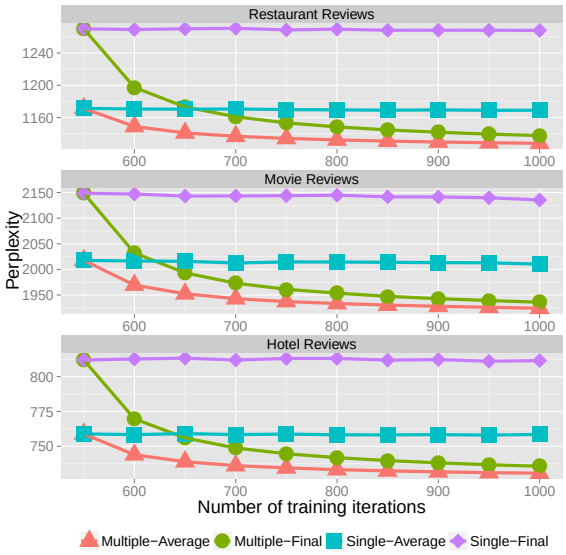


Figure 2: Perplexity of LDA using different averaging strategies with different number of training iterations T_{TR}. Perplexity generally decreases with additional training iterations, but the drop is more pronounced with multiple test chains.

Results: Figure 2 shows the perplexity of the four averaging methods, computed with different number of training iterations T_{TR}. SA outperforms SF, showing the benefits of averaging over multiple test samples from a single test chain. However, both multiple chain methods (MF and MA) significantly outperform these two methods.

This result is consistent with Asuncion et al. (2009), who run multiple training chains but a single test chain for each training chain and average over them. This is more costly since training chains are usually significantly longer than test chains. In addition, multiple training chains are sensitive to their initialization.

[3] To find bigrams, we begin with bigram candidates that occur at least 10 times in the corpus and use a χ^2 test to filter out those having a χ^2 value less than 5. We then treat selected bigrams as single word types and add them to the vocabulary.

[4] MCMC setup: $T_{\text{TR}} = 1,000$, $B_{\text{TR}} = 500$, $L_{\text{TR}} = 50$, $T_{\text{TE}} = 100$, $B_{\text{TE}} = 50$ and $L_{\text{TE}} = 5$.

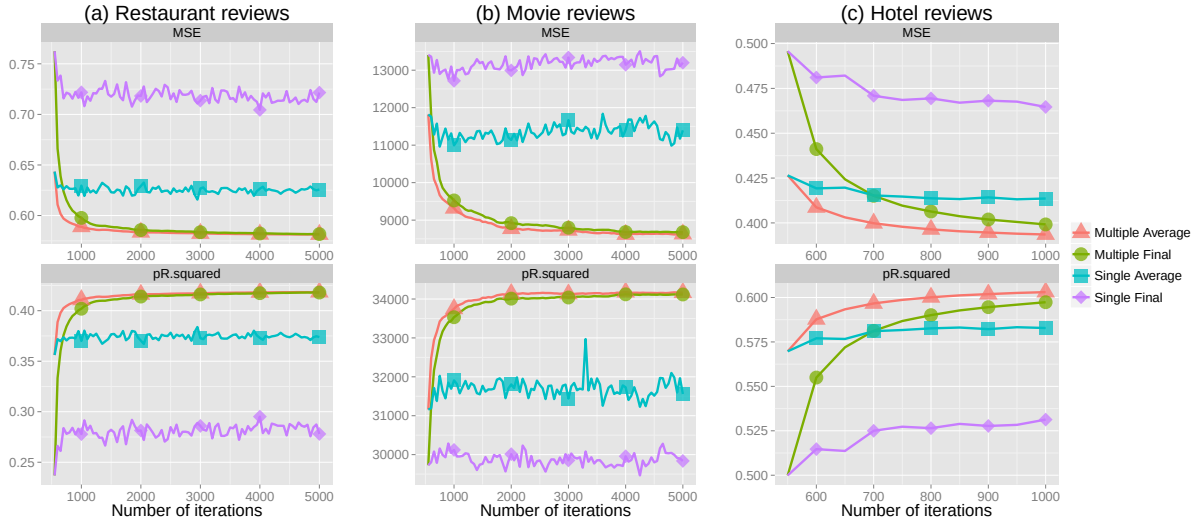


Figure 3: Performance of SLDA using different averaging strategies computed at each training iteration.

4 Supervised Topic Models

We evaluate the performance of different prediction methods using supervised latent Dirichlet allocation (SLDA) (Blei and McAuliffe, 2007) for sentiment analysis: predicting review ratings given review text. Each review text is the document $\boldsymbol{w}_d$ and the metadata $\boldsymbol{y}_d$ is the associated rating.

SLDA: Going beyond LDA, SLDA captures the relationship between latent topics and metadata by modeling each document's continuous response variable using a normal linear model, whose covariates are the document's empirical distribution of topics: $y_d \sim \mathcal{N}(\boldsymbol{\eta}^T \bar{\boldsymbol{z}}_d, \rho)$ where $\boldsymbol{\eta}$ is the regression parameter vector and $\bar{\boldsymbol{z}}_d$ is the empirical distribution over topics of document d. The generative process of SLDA is:

1. For each topic $k \in [1, K]$
   (a) Draw word distribution $\phi_k \sim \text{Dir}(\beta)$
   (b) Draw parameter $\eta_k \sim \mathcal{N}(\mu, \sigma)$
2. For each document $d \in [1, D]$
   (a) Draw topic distribution $\theta_d \sim \text{Dir}(\alpha)$
   (b) For each word $n \in [1, N_d]$
      i. Draw topic $z_{d,n} \sim \text{Mult}(\theta_d)$
      ii. Draw word $w_{d,n} \sim \text{Mult}(\phi_{z_{d,n}})$
   (c) Draw response $y_d \sim \mathcal{N}(\boldsymbol{\eta}^T \bar{\boldsymbol{z}}_d, \rho)$ where $\bar{z}_{d,k} = \frac{1}{N_d} \sum_{n=1}^{N_d} \mathbb{I}[z_{d,n} = k]$

where $\mathbb{I}[x] = 1$ if x is true, and 0 otherwise.

In SLDA, in addition to the K multinomials $\{\phi_k\}_{k=1}^K$, the global latent variables also contain the regression parameter η_k for each topic k. The local latent variables of SLDA resembles LDA's: the topic proportion vector θ_d for each document d.

Train: For posterior inference during training, following Boyd-Graber and Resnik (2010), we use stochastic EM, which alternates between (1) a Gibbs sampling step to assign a topic to each token, and (2) optimizing the regression parameters. The probability of assigning topic k to token n in the training document d is

$$p(z^{\text{TR}}_{d,n} = k \mid \boldsymbol{z}^{\text{TR}}_{-d,n}, \boldsymbol{w}^{\text{TR}}_{-d,n}, w^{\text{TR}}_{d,n} = v) \propto$$

$$\mathcal{N}(y_d; \mu_{d,n}, \rho) \cdot \frac{N^{-d,n}_{\text{TR},d,k} + \alpha}{N^{-d,n}_{\text{TR},d,\cdot} + K\alpha} \cdot \frac{N^{-d,n}_{\text{TR},k,v} + \beta}{N^{-d,n}_{\text{TR},k,\cdot} + V\beta} \quad (9)$$

where $\mu_{d,n} = (\sum_{k'=1}^K \eta_{k'} N^{-d,n}_{\text{TR},d,k'} + \eta_k)/N_{\text{TR},d}$ is the mean of the Gaussian generating y_d if $z^{\text{TR}}_{d,n} = k$. Here, $N_{\text{TR},d,k}$ is the number of times topic k is assigned to tokens in the training document d; $N_{\text{TR},k,v}$ is the number of times word type v is assigned to topic k; $\cdot$ represents marginal counts and $^{-d,n}$ indicates counts excluding the assignment of token n in document d.

We optimize the regression parameters $\boldsymbol{\eta}$ using L-BFGS (Liu and Nocedal, 1989) via the likelihood

$$\mathcal{L}(\boldsymbol{\eta}) = -\frac{1}{2\rho} \sum_{d=1}^{D} (y^{\text{TR}}_d - \boldsymbol{\eta}^T \bar{\boldsymbol{z}}^{\text{TR}}_d)^2 - \frac{1}{2\sigma} \sum_{k=1}^{K} (\eta_k - \mu)^2 \quad (10)$$

At each iteration i in the training chain, the estimated global latent variables include the a multinomial $\hat{\phi}_k(i)$ and a regression parameter $\hat{\eta}_k(i)$ for each topic k.

Test: Like LDA, at test time we sample the topic assignments for all tokens in the test data

$$p(z^{\text{TE}}_{d,n} = k \mid \boldsymbol{z}^{\text{TE}}_{-d,n}, \boldsymbol{w}^{\text{TE}}) \propto \frac{N^{-d,n}_{\text{TE},d,k} + \alpha}{N^{-d,n}_{\text{TE},d,\cdot} + K\alpha} \cdot \hat{\phi}_{k,w^{\text{TE}}_{d,n}} \quad (11)$$

Prediction: The predicted value $S(i, j)$ in this case is the estimated value of the metadata review rating

$$S(i,j) \equiv \hat{y}^{\text{TE}}_d(i,j) = \hat{\boldsymbol{\eta}}(i)^T \bar{z}^{\text{TE}}_d(i,j), \quad (12)$$

where the empirical topic distribution of test document d is $\bar{z}^{\text{TE}}_{d,k}(i,j) \equiv \frac{1}{N_{\text{TE},d}} \sum_{n=1}^{N_{\text{TE},d}} \mathbb{I}\left[z^{\text{TE}}_{d,n}(i,j) = k\right]$.

Figure 4: Performance of SLDA using different averaging strategies computed at the final training iteration T_{TR}, compared with two baselines MLR and SVR. Methods using multiple test chains (MF and MA) perform as well as or better than the two baselines, whereas methods using a single test chain (SF and SA) perform significantly worse.

Experimental setup: We use the same data as in Section 3. For all datasets, the metadata are the review rating, ranging from 1 to 5 stars, which is standardized using z-normalization. We use two evaluation metrics: mean squared error (MSE) and predictive R-squared (Blei and McAuliffe, 2007).

For comparison, we consider two baselines: (1) multiple linear regression (MLR), which models the metadata as a linear function of the features, and (2) support vector regression (Joachims, 1999, SVR). Both baselines use the normalized frequencies of unigrams and bigrams as features. As in the unsupervised case, we report average performance over five cross-validated folds. For all models, we use a development set to tune their parameter(s) and use the set of parameters that gives best results on the development data at test.[5]

Results: Figure 3 shows SLDA prediction results with different averaging strategies, computed at different training iterations.[6] Consistent with the unsupervised results in Section 3, SA outperforms SF, but both are outperformed significantly by the two methods using multiple test chains (MF and MA).

We also compare the performance of the four prediction methods obtained at the final iteration T_{TR} of the training chain with the two baselines. The results in Figure 4 show that the two baselines (MLR and SVR) outperform significantly the SLDA using only a single test chains (SF and SA). Methods using multiple test chains (MF and MA), on the other hand, match the baseline [7] (HOTEL) or do better (RESTAURANT and MOVIE).

5 Discussion and Conclusion

MCMC relies on averaging multiple samples to approximate target densities. When used for prediction, MCMC needs to generate and average over both training samples to learn from training data and test samples to make prediction. We have shown that simple averaging—not more aggressive, *ad hoc* approximations like taking the final sample (either training or test)—is not just a question of theoretical aesthetics, but an important factor in obtaining good prediction performance.

Compared with SVR and MLR baselines, SLDA using multiple test chains (MF and MA) performs as well as or better, while SLDA using a single test chain (SF and SA) falters. This simple experimental setup choice can determine whether a model improves over reasonable baselines. In addition, better prediction with shorter training is possible with multiple test chains. Thus, we conclude that averaging using multiple chains produces above-average results.

Acknowledgments

We thank Jonathan Chang, Ke Zhai and Mohit Iyyer for helpful discussions, and thank the anonymous reviewers for insightful comments. This research was supported in part by NSF under grant #1211153 (Resnik) and #1018625 (Boyd-Graber and Resnik). Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

[5] For MLR we use a Gaussian prior $\mathcal{N}(0, 1/\lambda)$ with $\lambda = a \cdot 10^b$ where $a \in [1, 9]$ and $b \in [1, 4]$; for SVR, we use SVM^{light} (Joachims, 1999) and vary $C \in [1, 50]$, which trades off between training error and margin; for SLDA, we fix $\sigma = 10$ and vary $\rho \in \{0.1, 0.5, 1.0, 1.5, 2.0\}$, which trades off between the likelihood of words and response variable.

[6] MCMC setup: $T_{\text{TR}} = 5,000$ for RESTAURANT and MOVIE and $1,000$ for HOTEL; for all datasets $B_{\text{TR}} = 500$, $L_{\text{TR}} = 50$, $T_{\text{TE}} = 100$, $B_{\text{TE}} = 20$ and $L_{\text{TE}} = 5$.

[7] This gap is because SLDA has not converged after 1,000 training iterations (Figure 3).

References

Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. 2003. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43.

Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. 2009. On smoothing and inference for topic models. In *UAI*.

David M. Blei and Jon D. McAuliffe. 2007. Supervised topic models. In *NIPS*.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *JMLR*, 3.

David M. Blei. 2012. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, April.

David M. Blei. 2014. Build, compute, critique, repeat: Data analysis with latent variable models. *Annual Review of Statistics and Its Application*, 1(1):203–232.

Jordan Boyd-Graber and Philip Resnik. 2010. Holistic sentiment analysis across languages: Multilingual supervised latent Dirichlet allocation. In *EMNLP*.

Jonathan Chang. 2012. lda: Collapsed Gibbs sampling methods for topic models. `http://cran.r-project.org/web/packages/lda/index.html`. [Online; accessed 02-June-2014].

Qixia Jiang, Jun Zhu, Maosong Sun, and Eric P. Xing. 2012. Monte Carlo methods for maximum margin supervised topic models. In *NIPS*.

Yohan Jo and Alice H. Oh. 2011. Aspect and sentiment unification model for online review analysis. In *WSDM*.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*, chapter 11. Cambridge, MA.

Simon Lacoste-Julien, Fei Sha, and Michael I. Jordan. 2008. DiscLDA: Discriminative learning for dimensionality reduction and classification. In *NIPS*.

D. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Prog.*

Radford M. Neal. 1993. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto.

Viet-An Nguyen, Jordan Boyd-Graber, and Philip Resnik. 2012. SITS: A hierarchical nonparametric model using speaker identity for topic segmentation in multiparty conversations. In *ACL*.

Viet-An Nguyen, Jordan Boyd-Graber, and Philip Resnik. 2013. Lexical and hierarchical topic regression. In *Neural Information Processing Systems*.

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

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *EMNLP*.

Daniel Ramage, Christopher D. Manning, and Susan Dumais. 2011. Partially labeled topic models for interpretable text mining. In *KDD*, pages 457–465.

Philip Resnik and Eric Hardisty. 2010. Gibbs sampling for the uninitiated. Technical Report UMIACS-TR-2010-04, University of Maryland. http://drum.lib.umd.edu//handle/1903/10058.

Michal Rosen-Zvi, Thomas L. Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *UAI*.

Sameer Singh, Michael Wick, and Andrew McCallum. 2012. Monte Carlo MCMC: Efficient inference by approximate sampling. In *EMNLP*, pages 1104–1113.

Mark Steyvers and Tom Griffiths. 2006. Probabilistic topic models. In T. Landauer, D. Mcnamara, S. Dennis, and W. Kintsch, editors, *Latent Semantic Analysis: A Road to Meaning.* Laurence Erlbaum.

Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009. Evaluation methods for topic models. In Leon Bottou and Michael Littman, editors, *ICML*.

Hanna M Wallach. 2008. *Structured Topic Models for Language*. Ph.D. thesis, University of Cambridge.

Chong Wang, David Blei, and Li Fei-Fei. 2009. Simultaneous image classification and annotation. In *CVPR*.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: A rating regression approach. In *SIGKDD*, pages 783–792.

Jun Zhu, Amr Ahmed, and Eric P. Xing. 2009. MedLDA: maximum margin supervised topic models for regression and classification. In *ICML*.

Jun Zhu, Ning Chen, Hugh Perkins, and Bo Zhang. 2014. Gibbs max-margin topic models with data augmentation. *Journal of Machine Learning Research*, 15:1073–1110.

Large-scale Reordering Model for Statistical Machine Translation using Dual Multinomial Logistic Regression

Abdullah Alrajeh[ab] **and Mahesan Niranjan**[b]
[a]Computer Research Institute, King Abdulaziz City for Science and Technology (KACST)
Riyadh, Saudi Arabia, asrajeh@kacst.edu.sa
[b]School of Electronics and Computer Science, University of Southampton
Southampton, United Kingdom, {asar1a10, mn}@ecs.soton.ac.uk

Abstract

Phrase reordering is a challenge for statistical machine translation systems. Posing phrase movements as a prediction problem using contextual features modeled by maximum entropy-based classifier is superior to the commonly used lexicalized reordering model. However, Training this discriminative model using large-scale parallel corpus might be computationally expensive. In this paper, we explore recent advancements in solving large-scale classification problems. Using the dual problem to multinomial logistic regression, we managed to shrink the training data while iterating and produce significant saving in computation and memory while preserving the accuracy.

1 Introduction

Phrase reordering is a common problem when translating between two grammatically different languages. Analogous to speech recognition systems, statistical machine translation (SMT) systems relied on language models to produce more fluent output. While early work penalized phrase movements without considering reorderings arising from vastly differing grammatical structures across language pairs like Arabic-English (Koehn, 2004a), many researchers considered lexicalized reordering models that attempted to learn orientation based on the training corpus (Tillmann, 2004; Kumar and Byrne, 2005; Koehn et al., 2005).

Building on this, some researchers have borrowed powerful ideas from the machine learning literature, to pose the phrase movement problem as a prediction problem using contextual input features whose importance is modeled as weights of a linear classifier trained by entropic criteria. The approach (so called maximum entropy classifier or simply MaxEnt) is a popular choice (Zens and Ney, 2006; Xiong et al., 2006; Nguyen et al., 2009; Xiang et al., 2011). Max-margin structure classifiers were also proposed (Ni et al., 2011). Alternatively, Cherry (2013) proposed recently using sparse features optimize the translation quality with the decoder instead of training a classifier independently.

While large-scale parallel corpus is advantageous for improving such reordering model, this improvement comes at a price of computational complexity. This issue is particularly pronounced when discriminative models are considered such as maximum entropy-based model due to the required iterative learning.

Advancements in solving large-scale classification problems have been shown to be effective such as dual coordinate descent method for linear support vector machines (Hsieh et al., 2008). Similarly, Yu et al. (2011) proposed a two-level dual coordinate descent method for maximum entropy classifier.

In this work we explore the dual problem to multinomial logistic regression for building large-scale reordering model (section 3). One of the main advantages of solving the dual problem is providing a mechanism to shrink the training data which is a serious issue in building such large-scale system. We present empirical results comparing between the primal and the dual problems (section 4). Our approach is shown to be fast and memory-efficient.

2 Baseline System

In statistical machine translation, the most likely translation $\mathbf{e}_{\text{best}}$ of an input sentence $\mathbf{f}$ can be found by maximizing the probability $p(\mathbf{e}|\mathbf{f})$, as follows:

$$\mathbf{e}_{\text{best}} = \arg\max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}). \quad (1)$$

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1758–1763,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

A log-linear combination of different models (features) is used for direct modeling of the posterior probability $p(\mathbf{e}|\mathbf{f})$ (Papineni et al., 1998; Och and Ney, 2002):

$$\mathbf{e}_{\text{best}} = \arg\max_{\mathbf{e}} \sum_{i=1}^{n} \lambda_i h_i(\mathbf{f}, \mathbf{e}) \qquad (2)$$

where the feature $h_i(\mathbf{f}, \mathbf{e})$ is a score function over sentence pairs. The translation model and the language model are the main features in any system although additional features $h(.)$ can be integrated easily (such as word penalty). State-of-the-art systems usually have around ten features.

The language model, which ensures fluent translation, plays an important role in reordering; however, it has a bias towards short translations (Koehn, 2010). Therefore, a need for developing a specific model for the reordering problem.

2.1 Lexicalized Reordering Model

Adding a lexicalized reordering model consistently improved the translation quality for several language pairs (Koehn et al., 2005). Reordering modeling involves formulating phrase movements as a classification problem where each phrase position considered as a class (Tillmann, 2004). Some researchers classified phrase movements into three categories (monotone, swap, and discontinuous) but the classes can be extended to any arbitrary number (Koehn and Monz, 2005). In general, the distribution of phrase orientation is:

$$p(o_k|\bar{f}_i, \bar{e}_i) = \frac{1}{Z}\, h(\bar{f}_i, \bar{e}_i, o_k)\,. \qquad (3)$$

This lexicalized reordering model is estimated by relative frequency where each phrase pair $(\bar{f}_i, \bar{e}_i)$ with such an orientation (o_k) is counted and then normalized to yield the probability as follows:

$$p(o_k|\bar{f}_i, \bar{e}_i) = \frac{\text{count}(\bar{f}_i, \bar{e}_i, o_k)}{\sum_o \text{count}(\bar{f}_i, \bar{e}_i, o)}. \qquad (4)$$

The orientation of a current phrase pair is defined with respect to the previous target phrase. Galley and Manning (2008) extended the model to tackle long-distance reorderings. Their hierarchical model enables phrase movements that are more complex than swaps between adjacent phrases.

3 Multinomial Logistic Regression

Multinomial logistic regression (MLR), also known as maximum entropy classifier (Zens and Ney, 2006), is a probabilistic model for the multiclass problem. The class probability is given by:

$$p(o_k|\bar{f}_i, \bar{e}_i) = \frac{\exp(\mathbf{w}_k^\top \phi(\bar{f}_i, \bar{e}_i))}{\sum_{k'} \exp(\mathbf{w}_{k'}^\top \phi(\bar{f}_i, \bar{e}_i))}, \qquad (5)$$

where $\phi(\bar{f}_i, \bar{e}_i)$ is the feature vector of the i-th phrase pair. An equivalent notation to $\mathbf{w}_k^\top \phi(\bar{f}_i, \bar{e}_i)$ is $\mathbf{w}^\top f(\phi(\bar{f}_i, \bar{e}_i), o_k)$ where $\mathbf{w}$ is a long vector composed of all classes parameters (i.e. $\mathbf{w}^\top = [\mathbf{w}_1^\top \dots \mathbf{w}_K^\top]$) and $f(.,.)$ is a joint feature vector decomposed via the orthogonal feature representation (Rousu et al., 2006). This representation simply means there is no crosstalk between two different feature vectors. For example, $f(\phi(\bar{f}_i, \bar{e}_i), o_1)^\top = [\phi(\bar{f}_i, \bar{e}_i)^\top\, 0 \dots 0]$.

The model's parameters can be estimated by minimizing the following regularized negative log-likelihood $\mathcal{P}(\mathbf{w})$ as follows (Bishop, 2006):

$$\min_{\mathbf{w}} \; \frac{1}{2\sigma^2} \sum_{k=1}^{K} \|\mathbf{w}_k\|^2 - \sum_{i=1}^{N} \sum_{k=1}^{K} \tilde{p}_{ik} \log p(o_k|\bar{f}_i, \bar{e}_i) \qquad (6)$$

Here σ is a penalty parameter and $\tilde{p}$ is the empirical distribution where $\tilde{p}_{ik}$ equals zero for all $o_k \neq o_i$.

Solving the primal optimization problem (6) using the gradient:

$$\frac{\partial \mathcal{P}(\mathbf{w})}{\partial \mathbf{w}_k} = \frac{\mathbf{w}_k}{\sigma^2} - \sum_{i=1}^{N} \big(\tilde{p}_{ik} - p(o_k|\bar{f}_i, \bar{e}_i)\big)\, \phi(\bar{f}_i, \bar{e}_i), \qquad (7)$$

do not constitute a closed-form solution. In our experiments, we used stochastic gradient decent method (i.e. online learning) to estimate $\mathbf{w}$ which is shown to be fast and effictive for large-scale problems (Bottou, 2010). The method approximates (7) by a gradient at a single randomly picked phrase pair. The update rule is:

$$\mathbf{w}'_k = \mathbf{w}_k - \eta_i \nabla_k \mathcal{P}_i(\mathbf{w}), \qquad (8)$$

where η_i is a positive learning rate.

3.1 The Dual Problem

Lebanon and Lafferty (2002) derived an equivalent dual problem to (6). Introducing Lagrange multipliers $\boldsymbol{\alpha}$, the dual becomes

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2\sigma^2}\sum_{k=1}^{K}\|\mathbf{w}_k(\boldsymbol{\alpha})\|^2 + \sum_{i=1}^{N}\sum_{k=1}^{K}\alpha_{ik}\log\alpha_{ik}, \\ \text{s.t.} \quad & \sum_{k=1}^{K}\alpha_{ik} = 1 \ \text{ and } \ \alpha_{\text{ik}} \geq 0\,, \forall \text{i}, \text{k}, \end{aligned} \tag{9}$$

where

$$\mathbf{w}_k(\boldsymbol{\alpha}) = \sigma^2\sum_{i=1}^{N}(\tilde{p}_{ik} - \alpha_{ik})\phi(\bar{f}_i, \bar{e}_i) \tag{10}$$

As mentioned in the introduction, Yu et al. (2011) proposed a two-level dual coordinate descent method to minimize $\mathcal{D}(\boldsymbol{\alpha})$ in (9) but it has some numerical difficulties. Collins et al. (2008) proposed simple exponentiated gradient (EG) algorithm for Conditional Random Feild (CRF). The algorithm is applicable to our problem, a special case of CRF. The rule update is:

$$\alpha'_{ik} = \frac{\alpha_{ik}\exp(-\eta_i\,\nabla_{ik}\mathcal{D}(\boldsymbol{\alpha}))}{\sum_{k'}\alpha_{ik'}\exp(-\eta_i\,\nabla_{ik'}\mathcal{D}(\boldsymbol{\alpha}))} \tag{11}$$

where

$$\begin{aligned} \nabla_{ik}\mathcal{D}(\boldsymbol{\alpha}) &\equiv \frac{\partial\mathcal{D}(\boldsymbol{\alpha})}{\partial\alpha_{ik}} = 1 + \log\alpha_{ik} \\ &+ \left(\mathbf{w}_y(\boldsymbol{\alpha})^\top\phi(\bar{f}_i, \bar{e}_i) - \mathbf{w}_k(\boldsymbol{\alpha})^\top\phi(\bar{f}_i, \bar{e}_i)\right). \end{aligned} \tag{12}$$

Here y represents the true class (i.e. $o_y = o_i$). To improve the convergence, η_i is adaptively adjusted for each example. If the objective function (9) did not decrease, η_i is halved for number of trials (Collins et al., 2008). Calculating the function difference below is the main cost in EG algorithm,

$$\begin{aligned} \mathcal{D}(\boldsymbol{\alpha}') - \mathcal{D}(\boldsymbol{\alpha}) &= \sum_{k=1}^{K}\left(\alpha'_{ik}\log\alpha'_{ik} - \alpha_{ik}\log\alpha_{ik}\right) \\ &- \sum_{k=1}^{K}(\alpha'_{ik} - \alpha_{ik})\mathbf{w}_k(\boldsymbol{\alpha})^\top\phi(\bar{f}_i, \bar{e}_i) \\ &+ \frac{\sigma^2}{2}\|\phi(\bar{f}_i, \bar{e}_i)\|^2\sum_{k=1}^{K}(\alpha'_{ik} - \alpha_{ik})^2. \end{aligned} \tag{13}$$

Clearly, the cost is affordable because $\mathbf{w}_k(\boldsymbol{\alpha})$ is maintained throughout the algorithm as follows:

$$\mathbf{w}_k(\boldsymbol{\alpha}') = \mathbf{w}_k(\boldsymbol{\alpha}) - \sigma^2(\alpha'_{ik} - \alpha_{ik})\phi(\bar{f}_i, \bar{e}_i) \tag{14}$$

Following Yu et al. (2011), we initialize α_{ik} as follows:

$$\alpha_{ik} = \begin{cases} (1-\epsilon) & \text{if } o_k = o_i; \\ \frac{\epsilon}{K-1} & \text{else.} \end{cases} \tag{15}$$

where ϵ is a small positive value. This is because the objective function (9) is not well defined at $\alpha_{ik} = 0$ due to the logarithm appearance.

Finally, the optimal dual variables are achieved when the following condition is satisfied for all examples (Yu et al., 2011):

$$\max_k \nabla_{ik}\mathcal{D}(\boldsymbol{\alpha}) = \min_k \nabla_{ik}\mathcal{D}(\boldsymbol{\alpha}) \tag{16}$$

This condition is the key to accelerate EG algorithm. Unlike the primal problem (6), the dual variables α_{ik} are associated with each example (i.e. phrase pair) therefore a training example can be disregarded once its optimal dual variables obtained. More data shrinking can be achieved by tolerating a small difference between the two values in (16). Algorithm 1 presents the overall procedure (shrinking step is from line 6 to 9).

Algorithm 1 Shrinking stochastic exponentiated gradient method for training the dual problem

Require: training set $S = \{\phi(\bar{f}_i, \bar{e}_i), o_i\}_{i=1}^N$

1: Given $\boldsymbol{\alpha}$ and the corresponding $\mathbf{w}(\boldsymbol{\alpha})$
2: **repeat**
3: Randomly pick i from S
4: Claculate $\nabla_{ik}\mathcal{D}(\boldsymbol{\alpha})\ \forall k$ by (12)
5: $v_i = \max_k \nabla_{ik}\mathcal{D}(\boldsymbol{\alpha}) - \min_k \nabla_{ik}\mathcal{D}(\boldsymbol{\alpha})$
6: **if** $v_i \leq \epsilon$ **then**
7: Remove i from S
8: Continue from line 3
9: **end if**
10: $\eta = 0.5$
11: **for** $t = 1$ **to** maxTrial **do**
12: Calculate $\alpha'_{ik}\ \forall k$ by (11)
13: **if** $\mathcal{D}(\boldsymbol{\alpha}') - \mathcal{D}(\boldsymbol{\alpha}) \leq 0$ **then**
14: Update $\boldsymbol{\alpha}$ and $\mathbf{w}(\boldsymbol{\alpha})$ by (14)
15: Break
16: **end if**
17: $\eta = 0.5\,\eta$
18: **end for**
19: **until** $v_i \leq \epsilon \quad \forall i$

4 Experiments

We used MultiUN which is a large-scale parallel corpus extracted from the United Nations website (Eisele and Chen, 2010). We have used Arabic and English portion of MultiUN where the English side is about 300 million words.

We simplify the problem by classifying phrase movements into three categories (monotone, swap, discontinuous). To train the reordering models, we used GIZA++ to produce word alignments (Och and Ney, 2000). Then, we used the `extract` tool that comes with the Moses toolkit (Koehn et al., 2007) in order to extract phrase pairs along with their orientation classes.

As shown in Table 1, each extracted phrase pair is represented by linguistic features as follows:

- Aligned source and target words in a phrase pair. Each word alignment is a feature.
- Words within a window around the source phrase to capture the context. We choose adjacent words of the phrase boundary.

The extracted phrase pairs after filtering are 47,227,789. The features that occur more than 10 times are 670,154.

Sentence pair:

$\mathbf{f}$: $f_1\ f_2\ {}_1\ f_3\ f_4\ f_5\ {}_2\ f_6\ {}_3$.

$\mathbf{e}$: $e_1\ {}_1\ e_2\ e_3\ {}_3\ e_4\ e_5\ {}_2$.

Extracted phrase pairs $(\bar{\mathbf{f}}, \bar{\mathbf{e}})$:

$\bar{f}_i$	$\bar{e}_i$	o_i	alignment	context
$f_1\ f_2$	e_1	mono	0-0 1-0	f_3
$f_3\ f_4\ f_5$	$e_4\ e_5$	swap	0-1 2-0	$f_2\ f_6$
f_6	$e_2\ e_3$	other	0-0 0-1	f_5

All linguistic features:

1. f_1&e_1 2. f_2&e_1 3. f_3 4. f_3&e_5 5. f_5&e_4
6. f_2 7. f_6 8. f_6&e_2 9. f_6&e_3 10. f_5

Bag-of-words representation:

a phrase pair is represented as a vector where each feature is a discrete number (0=not exist).

$\phi(\bar{f}_i, \bar{e}_i)$	1	2	3	4	5	6	7	8	9	10
$\phi(\bar{f}_1, \bar{e}_1) =$	1	1	1	0	0	0	0	0	0	0
$\phi(\bar{f}_2, \bar{e}_2) =$	0	0	0	1	1	1	1	0	0	0
$\phi(\bar{f}_3, \bar{e}_3) =$	0	0	0	0	0	0	1	1	1	1

Table 1: A generic example of the process of phrase pair extraction and representation.

4.1 Classification

We trained our reordering models by both primal and dual classifiers for 100 iterations. For the dual MLR, different shrinking levels have been tried by varying the parameter (ϵ) in Algorithm 1. Table 2 reports the training time and classification error rate of these models.

Training the dual MLR with moderate shrinking level (i.e. $\epsilon = 0.1$) is almost four times faster than training the primal one. Choosing larger value for (ϵ) leads to faster training but might harm the performance as shown below.

Classifier	Training Time	Error Rate
Primal MLR	1 hour 9 mins	17.81%
Dual MLR ϵ:0.1	18 minutes	17.95%
Dual MLR ϵ:1.0	13 minutes	21.13%
Dual MLR ϵ:0.01	22 minutes	17.89%

Table 2: Performance of the primal and dual MLR based on held-out data.

Figure 1 shows the percentage of active set during training dual MLR with various shrinking levels. Interestingly, the dual MLR could disregard more than 99% of the data after a couple of iterations. For very large corpus, the data might not fit in memory and training primal MLR will take long time due to severe disk-swapping. In this situation, using dual MLR is very beneficial.

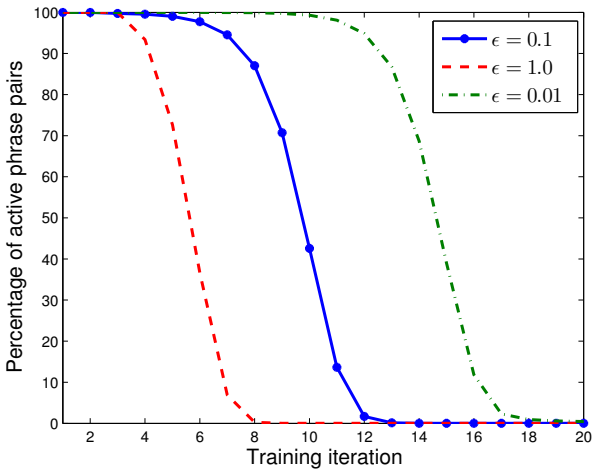


Figure 1: Percentage of active set in dual MLR. As the data size decreases, each iteration takes far less computation time (see Table 2 for total time).

4.2 Translation

We used the Moses toolkit (Koehn et al., 2007) with its default settings to build three phrase-based translation systems. They differ in how their reordering models were estimated. The language model is a 5-gram with interpolation and Kneser-Ney smoothing (Kneser and Ney, 1995). We tuned the system by using MERT technique (Och, 2003).

As commonly used in statistical machine translation, we evaluated the translation performance by BLEU score (Papineni et al., 2002). The test sets are NIST MT06 and MT08 where the English sides are 35,481 words (1056 sentences) and 116,840 words (3252 sentences), respectively. Table 3 shows the BLEU scores for the translation systems. We also computed statistical significance for the models using the *paired bootstrap resampling* method (Koehn, 2004b).

Translation System	MT06	MT08
Baseline + Lexical. model	30.86	34.22
Baseline + Primal MLR	31.37*	34.85*
Baseline + Dual MLR ϵ:0.1	31.36*	34.87*

Table 3: BLEU scores for Arabic-English translation systems with different reordering models (*: better than the lexicalized model with at least 95% statistical significance).

5 Conclusion

In training such system with large data sizes and big dimensionality, computational complexity become a serious issue. In SMT, maximum entropy-based reordering model is often introduced as a better alternative to the commonly used lexicalized one. However, training this discriminative model using large-scale corpus might be computationally expensive due to the iterative learning.

In this paper, we propose training the model using the dual MLR with shrinking method. It is almost four times faster than the primal MLR (also know as MaxEnt) and much more memory-efficient. For very large corpus, the data might not fit in memory and training primal MLR will take long time due to severe disk-swapping. In this situation, using dual MLR is very beneficial. The proposed method is also useful for many classification problems in natural language processing that require large-scale data.

References

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187, Paris, France, August. Springer.

Colin Cherry. 2013. Improved reordering for phrase-based translation using sparse features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 22–31, Atlanta, Georgia, June. Association for Computational Linguistics.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. 2008. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, 9:1775–1822, June.

Andreas Eisele and Yu Chen. 2010. Multiun: A multilingual corpus from united nation documents. In Daniel Tapias, Mike Rosner, Stelios Piperidis, Jan Odjik, Joseph Mariani, Bente Maegaard, Khalid Choukri, and Nicoletta Calzolari (Conference Chair), editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation*, pages 2868–2872. European Language Resources Association (ELRA), 5.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Hawaii, October. Association for Computational Linguistics.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. 2008. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 408–415.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 181–184.

Philipp Koehn and Christof Monz. 2005. Shared task: Statistical machine translation between european languages. In *Proceedings of ACL Workshop on Building and Using Parallel Texts*, pages 119–124. Association for Computational Linguistics.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description

for the 2005 IWSLT speech translation evaluation. In *Proceedings of International Workshop on Spoken Language Translation*, Pittsburgh, PA.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Christopher J. Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 177–180.

Philipp Koehn. 2004a. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of 6th Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 115—124, Washington DC.

Philipp Koehn. 2004b. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press.

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 161–168, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Guy Lebanon and John D. Lafferty. 2002. Boosting and maximum likelihood for exponential models. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 447–454. MIT Press.

Vinh Van Nguyen, Akira Shimazu, Minh Le Nguyen, and Thai Phuong Nguyen. 2009. Improving a lexicalized hierarchical reordering model using maximum entropy. In *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*. International Association for Machine Translation.

Yizhao Ni, Craig Saunders, Sandor Szedmak, and Mahesan Niranjan. 2011. Exploitation of machine learning techniques in modelling phrase movements for machine translation. *Journal of Machine Learning Research*, 12:1–30, February.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association of Computational Linguistics (ACL)*.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, and Todd Ward. 1998. Maximum likelihood and discriminative training of direct translation models. In *Proceedings of ICASSP*, pages 189–192.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. 2006. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, pages 1601–1626.

Christoph Tillmann. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL: Short Papers*, pages 101–104.

Bing Xiang, Niyu Ge, and Abraham Ittycheriah. 2011. Improving reordering for statistical machine translation with smoothed priors and syntactic features. In *Proceedings of SSST-5, Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 61–69, Portland, Oregon, USA. Association for Computational Linguistics.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 521–528, Sydney, July. Association for Computational Linguistics.

Hsiang-Fu Yu, Fang-Lan Huang, and Chih-Jen Lin. 2011. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1-2):41–75, October.

Richard Zens and Hermann Ney. 2006. Discriminative reordering models for statistical machine translation. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 55–63, New York City, June. Association for Computational Linguistics.

Improved Decipherment of Homophonic Ciphers

Malte Nuhn and **Julian Schamper** and **Hermann Ney**
Human Language Technology and Pattern Recognition
Computer Science Department, RWTH Aachen University, Aachen, Germany
<surname>@cs.rwth-aachen.de

Abstract

In this paper, we present two improvements to the beam search approach for solving homophonic substitution ciphers presented in Nuhn et al. (2013): An improved rest cost estimation together with an optimized strategy for obtaining the order in which the symbols of the cipher are deciphered reduces the beam size needed to successfully decipher the Zodiac-408 cipher from several million down to less than one hundred: The search effort is reduced from several hours of computation time to just a few seconds on a single CPU. These improvements allow us to successfully decipher the second part of the famous Beale cipher (see (Ward et al., 1885) and e.g. (King, 1993)): Having 182 different cipher symbols while having a length of just 762 symbols, the decipherment is way more challenging than the decipherment of the previously deciphered Zodiac-408 cipher (length 408, 54 different symbols). To the best of our knowledge, this cipher has not been deciphered automatically before.

1 Introduction

State-of-the-art statistical machine translation systems use large amounts of parallel data to estimate translation models. However, parallel corpora are expensive and not available for every domain.

Decipherment uses only monolingual data to train a translation model: Improving the core decipherment algorithms is an important step for making decipherment techniques useful for training practical machine translation systems.

In this paper we present improvements to the beam search algorithm for deciphering homophonic substitution ciphers as presented in Nuhn et al. (2013). We show significant improvements in computation time on the Zodiac-408 cipher and show the first decipherment of part two of the Beale ciphers.

2 Related Work

Regarding the decipherment of 1:1 substitution ciphers, various works have been published: Most older papers do not use a statistical approach and instead define some heuristic measures for scoring candidate decipherments. Approaches like Hart (1994) and Olson (2007) use a dictionary to check if a decipherment is useful. Clark (1998) defines other suitability measures based on n-gram counts and presents a variety of optimization techniques like simulated annealing, genetic algorithms and tabu search. On the other hand, statistical approaches for 1:1 substitution ciphers are published in the natural language processing community: Ravi and Knight (2008) solve 1:1 substitution ciphers optimally by formulating the decipherment problem as an integer linear program (ILP) while Corlett and Penn (2010) solve the problem using A^* search. Ravi and Knight (2011) report the first automatic decipherment of the Zodiac-408 cipher. They use a combination of a 3-gram language model and a word dictionary. As stated in the previous section, this work can be seen as an extension of Nuhn et al. (2013). We will therefore make heavy use of their definitions and approaches, which we will summarize in Section 3.

3 General Framework

In this Section we recap the beam search framework introduced in Nuhn et al. (2013).

3.1 Notation

We denote the ciphertext with $f_1^N = f_1 \dots f_j \dots f_N$ which consists of cipher

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1764–1768,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

tokens $f_j \in V_f$. We denote the plaintext with $e_1^N = e_1 \dots e_i \dots e_N$ (and its vocabulary V_e respectively). We define $e_0 = f_0 = e_{N+1} = f_{N+1} = \$$ with "$" being a special sentence boundary token. Homophonic substitutions are formalized with a general function $\phi : V_f \rightarrow V_e$. Following (Corlett and Penn, 2010), cipher functions ϕ, for which not all $\phi(f)$'s are fixed, are called partial cipher functions. Further, ϕ' is said to extend ϕ, if for all $f \in V_f$ that are fixed in ϕ, it holds that f is also fixed in ϕ' with $\phi'(f) = \phi(f)$. The cardinality of ϕ counts the number of fixed f's in ϕ. When talking about partial cipher functions we use the notation for relations, in which $\phi \subseteq V_f \times V_e$.

3.2 Beam Search

The main idea of (Nuhn et al., 2013) is to structure all partial ϕ's into a search tree: If a cipher contains N unique symbols, then the search tree is of height N. At each level a decision about the n-th symbol is made. The leaves of the tree form full hypotheses. Instead of traversing the whole search tree, beam search descents the tree top to bottom and only keeps the most promising candidates at each level. Practically, this is done by keeping track of all partial hypotheses in two arrays H_s and H_t. During search all allowed extensions of the partial hypotheses in H_s are generated, scored and put into H_t. Here, the function EXT_ORDER (see Section 5) chooses which cipher symbol is used next for extension, EXT_LIMITS decides which extensions are allowed, and SCORE (see Section 4) scores the new partial hypotheses. PRUNE then selects a subset of these hypotheses. Afterwards the array H_t is copied to H_s and the search process continues with the updated array H_s. Figure 1 shows the general algorithm.

```
1: function BEAM_SEARCH(EXT_ORDER)
2:     init sets H_s, H_t
3:     CARDINALITY = 0
4:     H_s.ADD((∅, 0))
5:     while CARDINALITY < |V_f| do
6:         f = EXT_ORDER[CARDINALITY]
7:         for all φ ∈ H_s do
8:             for all e ∈ V_e do
9:                 φ' := φ ∪ {(e, f)}
10:                if EXT_LIMITS(φ') then
11:                    H_t.ADD(φ',SCORE (φ'))
12:                end if
13:            end for
14:        end for
15:        PRUNE(H_t)
16:        CARDINALITY = CARDINALITY + 1
17:        H_s = H_t
18:        H_t.CLEAR()
19:    end while
20:    return best scoring cipher function in H_s
21: end function
```

Figure 1: The general structure of the beam search algorithm for decipherment of substitution ciphers as presented in Nuhn et al. (2013). This paper improves the functions SCORE and EXT_ORDER.

4 Score Estimation

The score estimation function is crucial to the search procedure: It predicts how good or bad a partial cipher function ϕ might become, and therefore, whether it's worth to keep it or not.

To illustrate how we can calculate these scores, we will use the following example with vocabularies $V_f = \{A, B, C, D\}$, $V_e = \{a, b, c, d\}$, extension order (B, C, A, D), and cipher text[1]

$$f_1^N = \texttt{\$ ABDD CABC DADC ABDC \$}$$

[1]We include blanks only for clarity reasons.

and partial hypothesis $\phi = \{(A, a), (B, b)\}$. This yields the following partial decipherment

$$\phi(f_1^N) = \texttt{\$ ab.. .ab. .a.. ab.. \$}$$

The score estimation function can only use this partial decipherment to calculate the hypothesis' score, since there are not yet any decisions made about the other positions.

4.1 Baseline

Nuhn et al. (2013) present a very simple rest cost estimator, which calculates the hypothesis' score based only on fully deciphered n-grams, i.e. those parts of the partial decipherment that form a contiguous chunk of n deciphered symbols. For all other n-grams containing not yet deciphered symbols, a trivial estimate of probability 1 is assumed, making it an admissible heuristic. For the above example, this baseline yields the probability $p(a|\$) \cdot p(b|a) \cdot 1^4 \cdot p(b|a) \cdot 1^6 \cdot p(b|a) \cdot 1^2$. The more symbols are fixed, the more contiguous n-grams become available. While being easy and efficient to compute, it can be seen that for example the single "a" is not involved in the computation of

the score at all. In practical decipherment, like e.g. the Zodiac-408 cipher, this forms a real problem: While making the first decisions—i.e. traversing the first levels of the search tree—only very few terms actually contribute to the score estimation, and thus only give a very coarse score. This makes the beam search "blind" when not many symbols are deciphered yet. This is the reason, why Nuhn et al. (2013) need a large beam size of several million hypotheses in order to not lose the right hypothesis during the first steps of the search.

4.2 Improved Rest Cost Estimation

The rest cost estimator we present in this paper solves the problem mentioned in the previous section by also including lower order n-grams: In the example mentioned before, we would also include unigram scores into the rest cost estimate, yielding a score of $p(a|\$) \cdot p(b|a) \cdot 1^3 \cdot p(a) \cdot p(b|a) \cdot 1^2 \cdot p(a) 1^2 \cdot p(a) \cdot p(b|a) \cdot 1^2$. Note that this is not a simple linear interpolation of different n-gram trivial scores: Each symbol is scored only using the maximum amount of context available. This heuristic is non-admissible, since an increased amount of context can always lower the probabilty of some symbols. However, experiments show that this score estimation function works great.

5 Extension Order

Besides having a generally good scoring function, also the order in which decisions about the cipher symbols are made is important for obtaining reliable cost estimates. Generally speaking we want an extension order that produces partial decipherments that contain useful information to decide whether a hypothesis is worth being kept or not as early as possible.

It is also clear that the choice of a good extension order is dependent on the score estimation function SCORE. After presenting the previous state of the art, we introduce a new extension order optimized to work together with our previously introduced rest cost estimator.

5.1 Baseline

In (Nuhn et al., 2013), two strategies are presented: One which at each step chooses the most frequent remaining cipher symbol, and another, which greedily chooses the next symbol to maximize the number of contiguously fixed n-grams in the ciphertext.

LM order	Perplexity	
	Zodiac-408	Beale Pt. 2
1	19.49	18.35
2	14.09	13.96
3	12.62	11.81
4	11.38	10.76
5	11.19	9.33
6	10.13	8.49
7	10.15	8.27
8	9.98	8.27

Table 1: Perplexities of the correct decipherment of Zodiac-408 and part two of the Beale ciphers using the character based language model used in beam search. The language model was trained on the English Gigaword corpus.

5.2 Improved Extension Order

Each partial mapping ϕ defines a partial decipherment. We want to choose an extension order such that *all possible* partial decipherments following this extension order are as informative as possible: Due to that, we can only use information about *which* symbols will be deciphered, not their actual decipherment. Since our heuristic is based on n-grams of different orders, it seems natural to evaluate an extension order by counting how many contiguously deciphered n-grams are available: Our new strategy tries to find an extension order optimizing the weighted sum of contiguously deciphered n-gram counts[2]

$$\sum_{n=1}^{N} w_n \cdot \#_n.$$

Here n is the n-gram order, w_n the weight for order n, and $\#_n$ the number of positions whose maximum context is of size n.

We perform a beam search over all possible enumerations of the cipher vocabulary: We start with fixing only the first symbol to decipher. We then continue with the second symbol and evaluate all resulting extension orders of length 2. In our experiments, we prune these candidates to the 100 best ones and continue with length 3, and so on.

Suitable values for the weights w_n have to be chosen. We try different weights for the different

[2]If two partial extension orders have the same score after fixing n symbols, we fall back to comparing the scores of the partial extension orders after fixing only the first $n-1$ symbols.

i_{02}	h_{08}	a_{03}	v_{01}	e_{05}	d_{09}	e_{07}	p_{03}	o_{07}	s_{10}	i_{11}	t_{03}	e_{14}	d_{03}	i_{03}	n_{05}	t_{06}	h_{01}	e_{13}	c_{04}	o_{10}	u_{01}	n_{01}	t_{04}	y_{01}
o_{12}	f_{04}	b_{04}	e_{15}	d_{09}	f_{03}	o_{04}	r_{06}	d_{04}	a_{07}	b_{07}	o_{09}	u_{03}	t_{13}	f_{01}	o_{01}	u_{08}	r_{05}	m_{03}	i_{08}	l_{09}	e_{14}	s_{06}	f_{01}	r_{05}
o_{07}	m_{04}	b_{06}	u_{02}	f_{04}	o_{10}	r_{07}	d_{01}	s_{11}	i_{03}	n_{02}	a_{06}	n_{03}	e_{05}	x_{01}	c_{03}	a_{01}	v_{01}	a_{03}	t_{10}	i_{13}	o_{03}	n_{05}	o_{08}	r_{06}
v_{01}	a_{08}	u_{03}	l_{01}	t_{11}	s_{12}	i_{04}	x_{01}	f_{01}	e_{01}	e_{03}	t_{02}	b_{06}	e_{07}	l_{02}	o_{11}	w_{06}	t_{08}	h_{08}	e_{15}	s_{06}	u_{04}	r_{06}	f_{04}	a_{10}
											$\vdots$												
p_{04}	a_{14}	p_{01}	e_{07}	r_{05}	n_{02}	u_{02}	m_{02}	b_{01}	e_{14}	r_{05}	o_{03}	n_{05}	e_{15}	d_{10}	e_{01}	s_{01}	c_{01}	r_{01}	i_{03}	b_{05}	e_{06}	s_{08}	t_{01}	h_{08}
c_{04}	e_{10}	x_{01}	a_{14}	c_{07}	t_{02}	l_{09}	o_{12}	c_{02}	a_{04}	l_{09}	i_{13}	t_{02}	y_{01}	o_{02}	f_{03}	t_{07}	h_{02}	e_{11}	v_{01}	a_{10}	r_{07}	l_{07}	t_{11}	s_{09}
o_{04}	t_{01}	h_{03}	a_{06}	t_{04}	n_{03}	o_{06}	d_{05}	i_{13}	f_{02}	f_{03}	i_{03}	c_{04}	u_{07}	l_{09}	t_{02}	y_{01}	w_{04}	i_{12}	l_{01}	l_{02}	b_{03}	e_{01}	h_{02}	a_{09}
d_{10}	i_{07}	n_{06}	f_{01}	i_{13}	n_{01}	d_{10}	i_{03}	n_{05}	g_{04}	i_{03}	t_{05}												

Table 2: Beginning and end of part two of the Beale cipher. Here we show a relabeled version of the cipher, which encodes knowledge of the gold decipherment to assign reasonable names to all homophones. The original cipher just consists of numbers.

orders on the Zodiac-408 cipher with just a beam size of 26. With such a small beam size, the extension order plays a crucial role for a successful decipherment: Depending on the choice of the different weights w_n we can observe decipherment runs with 3 out of 54 correct mappings, up to 52 out of 54 mappings correct. Even though the choice of weights is somewhat arbitrary, we can see that generally giving higher weights to higher n-gram orders yields better results.

We use the weights $w_1^8 = (0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 2.0, 3.0)$ for the following experiments. It is interesting to compare these weights to the perplexities of the correct decipherment measured using different n-gram orders (Table 5). However, at this point we do not see any obvious connection between perplexities and weights w_n, and leave this as a further research direction.

6 Experimental Evaluation

6.1 Zodiac Cipher

Using our new algorithm we are able to decipher the Zodiac-408 with just a beam size of 26 and a language model order of size 8. By keeping track of the gold hypothesis while performing the beam search, we can see that the gold decipherment indeed always remains within the top 26 scoring hypotheses. Our new algorithm is able to decipher the Zodiac-408 cipher in less than 10s on a single CPU, as compared to 48h of CPU time using the previously published heuristic, which required a beam size of several million. Solving a cipher with such a small beam size can be seen as "reading off the solution".

6.2 Beale Cipher

We apply our algorithm to the second part of the Beale ciphers with a 8-gram language model.

Compared to the Zodiac-408, which has length 408 while having 54 different symbols (7.55 observations per symbol), part two of the Beale ciphers has length 762 while having 182 different symbols (4.18 observations per symbol). Compared to the Zodiac-408, this is both, in terms of redundancy, as well as in size of search space, a way more difficult cipher to break.

Here we run our algorithm with a beam size of 10M and achieve a decipherment accuracy of 157 out of 185 symbols correct yielding a symbol error rate of less than 5.4%. The gold decipherment is pruned out of the beam after 35 symbols have been fixed.

We also ran our algorithm on the other parts of the Beale ciphers: The first part has a length 520 and contains 299 different cipher symbols (1.74 observations per symbol), while part three has length 618 and has 264 symbols which is 2.34 observations per mapping. However, our algorithm does not yield any reasonable decipherments. Since length and number of symbols indicate that deciphering these ciphers is again more difficult than for part two, it is not clear whether the other parts are not a homophonic substitution cipher at all, or whether our algorithm is still not good enough to find the correct decipherment.

7 Conclusion

We presented two extensions to the beam search method presented in (Nuhn et al., 2012), that reduce the search effort to decipher the Zodiac-408 enormously. These improvements allow us to automatically decipher part two of the Beale ciphers. To the best of our knowledge, this has not been

done before. This algorithm might prove useful when applied to word substitution ciphers and to learning translations from monolingual data.

Acknowledgements

The authors thank Mark Kozek from the Department of Mathematics at Whittier College for challenging us with a homophonic cipher he created. Working on his cipher led to developing the methods presented in this paper.

References

Andrew J. Clark. 1998. *Optimisation heuristics for cryptology*. Ph.D. thesis, Faculty of Information Technology, Queensland University of Technology.

Eric Corlett and Gerald Penn. 2010. An exact A* method for deciphering letter-substitution ciphers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1040–1047, Uppsala, Sweden, July. The Association for Computer Linguistics.

George W. Hart. 1994. To decode short cryptograms. *Communications of the Association for Computing Machinery (CACM)*, 37(9):102–108, September.

John C. King. 1993. A reconstruction of the key to beale cipher number two. *Cryptologia*, 17(3):305–317.

Malte Nuhn, Arne Mauser, and Hermann Ney. 2012. Deciphering foreign language by combining language models and context vectors. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 156–164, Jeju, Republic of Korea, July. Association for Computational Linguistics.

Malte Nuhn, Julian Schamper, and Hermann Ney. 2013. Beam search for solving substitution ciphers. In *Annual Meeting of the Assoc. for Computational Linguistics*, pages 1569–1576, Sofia, Bulgaria, August.

Edwin Olson. 2007. Robust dictionary attack of short simple substitution ciphers. *Cryptologia*, 31(4):332–342, October.

Sujith Ravi and Kevin Knight. 2008. Attacking decipherment problems optimally with low-order n-gram models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 812–819, Honolulu, Hawaii. Association for Computational Linguistics.

Sujith Ravi and Kevin Knight. 2011. Bayesian inference for Zodiac and other homophonic ciphers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 239–247, Portland, Oregon, June. Association for Computational Linguistics.

James B Ward, Thomas Jefferson Beale, and Robert Morriss. 1885. *The Beale Papers*.

Cipher Type Detection

Malte Nuhn
Human Language Technology
and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
nuhn@cs.rwth-aachen.de

Kevin Knight
Information Sciences Institute
University of Southern California
knight@isi.edu

Abstract

Manual analysis and decryption of enciphered documents is a tedious and error prone work. Often—even after spending large amounts of time on a particular cipher—no decipherment can be found. Automating the decryption of various types of ciphers makes it possible to sift through the large number of encrypted messages found in libraries and archives, and to focus human effort only on a small but potentially interesting subset of them. In this work, we train a classifier that is able to predict which encipherment method has been used to generate a given ciphertext. We are able to distinguish 50 different cipher types (specified by the American Cryptogram Association) with an accuracy of 58.5%. This is a 11.2% absolute improvement over the best previously published classifier.

1 Introduction

Libraries and archives contain a large number of encrypted messages created throughout the centuries using various encryption methods. For the great majority of the ciphers an analysis has not yet been conducted, simply because it takes too much time to analyze each cipher individually, or because it is too hard to decipher them. Automatic methods for analyzing and classifying given ciphers makes it possible to sift interesting messages and by that focus the limited amount of human resources to a promising subset of ciphers.

For specific types of ciphers, there exist automated tools to decipher encrypted messages. However, the publicly available tools often depend on a more or less educated guess which type of encipherment has been used. Furthermore, they often still need human interaction and are only restricted to analyzing very few types of ciphers. In practice however, there are many different types of ciphers which we would like to analyze in a fully automatic fashion: Bauer (2010) gives a good overview over historical methods that have been used to encipher messages in the past. Similarly, the American Cryptogram Association (ACA) specifies a set of 56 different methods for enciphering a given plaintext:

Each encipherment method M_i can be seen as a function that transforms a given plaintext into a ciphertext using a given key, or short:

$$\text{cipher} = M_i(\text{plain}, \text{key})$$

When analyzing an unknown ciphertext, we are interested in the original plaintext that was used to generate the ciphertext, i.e. the opposite direction:

$$\text{plain} = M_i^{-1}(\text{cipher}, \text{key})$$

Obtaining the plaintext from an enciphered message is a difficult problem. We assume that the decipherment of a message can be separated into solving three different subproblems:

1. Find the encipherment method M_i that was used to create the cipher
$$\text{cipher} \quad \rightarrow \quad M_i$$
2. Find the key that was used together with the method M_i to encipher the plaintext to obtain $cipher = M_i(plain, key)$.
3. Decode the message using M_i and key
$$\text{cipher} \quad \rightarrow \quad M_i^{-1}(\text{cipher}, \text{key})$$

Thus, an intermediate step to deciphering an unknown ciphertext is to find out which encryption method was used. In this paper, we present a classifier that is able to predict just that: Given an unknown ciphertext, it can predict what kind of encryption method was most likely used to generate

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1769–1773,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

- Type: CMBIFID
- Plaintext:
  WOMEN NSFOO TBALL ISGAI
  NINGI NPOPU LARIT YANDT
  HETOU RNAME
- Key:
  LEFTKEY='IACERATIONS'
  RIGHTKEY='KNORKOPPING'
  PERIOD=3, LROUTE=1
  RROUTE=1, USE6X6=0
- Ciphertext:
  WTQNG GEEBQ BPNQP VANEN
  KDAOD GAHQS PKNVI PTAAP
  DGMGR PCSGN

Figure 1: Example "CMBIFID" cipher: Text is grouped in five character chunks for readability.

it. The results of our classifier are a valuable input to human decipherers to make a first categorization of an unknown ciphertext.

2 Related Work

Central to this work is the list of encryption methods provided by the American Cipher Association[1]. This list contains detailed descriptions and examples of each of the cipher types, allowing us to implement them. Figure 3 lists these methods.

We compare our work to the only previously published cipher type classifier for classical ciphers[2]. This classifier is trained on $16,800$ ciphertexts and is implemented in javascript to run in the web browser: The user can provide the ciphertext as input to a web page that returns the classifier's predictions. The source code of the classifier is available online. Our work includes a reimplementation of the features used in that classifier.

As examples for work that deals with the automated decipherment of cipher texts, we point to (Ravi and Knight, 2011), and (Nuhn et al., 2013). These publications develop specialized algorithms for solving simple and homophonic substitution ciphers, which are just two out of the 56 cipher types defined by the ACA. We also want to mention (de Souza et al., 2013), which presents a cipher type classifier for the finalist algorithms of the Advanced Encryption Standard (AES) contest.

[1] http://cryptogram.org/cipher_types.html

[2] See http://bionsgadgets.appspot.com/gadget_forms/refscore_extended.html and https://sites.google.com/site/bionspot/cipher-id-tests

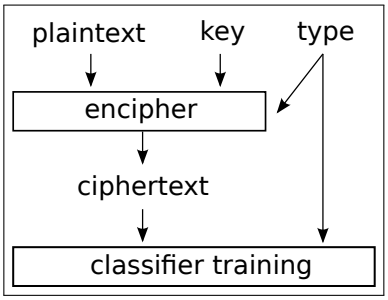


Figure 2: Overview over the data generation and training of the classifier presented in this work.

3 General Approach

Given a ciphertext, the task is to find the right encryption method. Our test set covers 50 out of 56 cipher types specified by ACA, as listed in Figure 3. We are going to take a machine learning approach which is based on the observation that we can generate an infinite amount of training data.

3.1 Data Flow

The training procedure is depicted in Figure 2: Based upon a large English corpus, we first choose possible plaintext messages. Then, for each encipherment method, we choose a random key and encipher each of the plaintext messages using the encipherment method and key. By doing this, we can obtain (a theoretically infinite) amount of labeled data of the form $(\text{type, ciphertext})$. We can then train a classifier on this data and evaluate it on some held out data.

Figure 1 shows that in general the key can consist of more than just a codeword: In this case, the method uses two codewords, a period length, two different permutation parameters, and a general decision whether to use a special "6×6" variant of the cipher or not. If not defined otherwise, we choose random settings for these parameters. If the parameters are integers, we choose random values from a uniform distribution (in a sensible range). In case of codewords, we choose the 450k most frequent words from an English dictionary. We train on cipher texts of random length.

3.2 Classifiers

The previous state-of-the-art classifier by BION uses a random forest classifier (Breiman, 2001). The version that is available online, uses 50 ran-

- 6x6bifid
- 6x6playfair
- amsco
- bazeries
- beaufort
- bifid6
- bifid7
- (cadenus)
- cmbifid
- columnar
- digrafid
- dbl chckrbrd
- four square
- fracmorse
- grandpre
- (grille)
- gromark
- gronsfeld
- homophonic
- mnmedinome
- morbit
- myszkowski
- nicodemus
- nihilistsub
- (nihilisttransp)
- patristocrat
- period 7 vig.
- periodic gromark
- phillips
- plaintext
- playfair
- pollux
- porta
- portax
- progkey beaufort
- progressivekey
- quagmire2
- quagmire3
- quagmire4
- ragbaby
- randomdigit
- randomtext
- redefence
- (route transp)
- runningkey
- seriatedpfair
- swagman
- tridigital
- trifid
- trisquare
- trisquare hr
- two square
- two sq. spiral
- vigautokey
- (vigenere)
- (vigslidefair)

Figure 3: Cipher types specified by ACA. Our classifier is able to recognize 50 out of these 56 ciphers. The braced cipher types are not covered in this work.

dom decision trees. The features used by this classifier are described in Section 4.

Further, we train a support vector machine using the libSVM toolkit (Chang and Lin, 2011). This is feasible for up to 100k training examples. Beyond this point, training times become too large. We perform multi class classification using ν-SVC and a polynomial kernel. Multi class classification is performed using one-against-one binary classification. We select the SVM's free parameters using a small development set of 1k training examples.

We also use Vowpal Wabbit (Langford et al., 2007) to train a linear classifier using stochastic gradient descent. Compared to training SVMs, Vowpal Wabbit is extremely fast and allows using a lot of training examples. We use a squared loss function, adaptive learning rates and don't employ any regularization. We train our classifier with up to 1M training examples. The best performing settings use one-against-all classification, 20 passes over the training data and the default learning rate. Quadratic features resulted in much slower training, while not providing any gains in accuracy.

4 Features

We reimplemented all of the features used in the BION classifier, and add three newly developed sets of features, resulting in a total of 58 features.

In order to further structure these features, we group these features as follows: We call the set of features that relate to the length of the cipher `LEN`. This set contains binary features firing when the cipher length is a multiple of 2, 3, 5, 25, any of 4-15, and any of 4-30. We call the set of features that are based on the fact that the ciphertext contains a specific symbol `HAS`. This set contains binary features firing when the cipher contains a digit, a letter (A-Z), the "#" symbol, the letter "j", the digit "0". We also introduce another set of features called `DGT` that contains two features, firing when the cipher is starting or ending with a digit. The set `VIG` contains 5 features: The feature score is based on the best possible bigram LM perplexity of a decipherment compatible with the decipherment process of the cipher types Autokey, Beaufort, Porta, Slidefair and Vigenere. Further, we also include the features `IC`, `MIC`, `MKA`, `DIC`, `EDI`, `LR`, `ROD` and `LDI`, `DBL`, `NOMOR`, `RDI`, `PTX`, `NIC`, `PHIC`, `BDI`, `CDD`, `SSTD`, `MPIC`, `SERP`, which were introduced in the BION classifier[3]. Thus, the first 22 data points in Figure 4 are based on previously known features by BION. We further present the following additional features.

4.1 Repetition Feature (`REP`)

This set of features is based on how often the ciphertext contains symbols that are repeated exactly n times in a row: For example the ciphertext shown in Figure 1 contains two positions with repetitions of length $n = 2$, because the ciphertext contains EE, as well as AA. Beyond length 2, there are no repeats. These numbers are then normalized by dividing them by the total number of repeats of length $2 \leq n \leq 5$.

4.2 Amsco Feature (`AMSC`)

The idea of the AMSCO cipher is to fill consecutive chunks of one and two plaintext characters into n columns of a grid (see Table 1). Then a permutation of the columns is performed, and the resulting permuted plaintext is read of line by line and forms the final ciphertext. This feature reads the ciphertext into a similar grid of up to 5 columns

[3] See `http://home.comcast.net/~acabion/acarefstats.html`

Plaintext	w oo	om t	e ba	ns l	f li
Permutation	3	5	1	4	2

Table 1: Example grid used for AMSCO ciphers.

and then tries all possible permutations to retain the original plaintext. The result of this operation is then scored with a bigram language model. Depending on whether the difference in perplexity between ciphertext and deciphered text exceeds a given threshold, this binary feature fires.

4.3 Variant Feature (VAR)

In the variant cipher, the plaintext is written into a block under a key word. All letters in the first column are enciphered by shifting them using the first key letter of the key word, the second column uses the second key letter, etc. For different periods (i.e. lengths of key words), the ciphertext is structured into n columns and unigram statistics for each column are calculated. The frequency profile of each column is compared to the unigram frequency profile using a perplexity measure. This binary feature fires when the resulting perplexities are lower than a specific threshold.

5 Results

Figure 4 shows the classification accuracy for the BION baseline, as well as our SVM and VW based classifiers for a test set of 305 ciphers that have been published in the ACA. The classifiers shown in this figure are trained on cipher texts of random length. We show the contribution of all the features we used in the classifier on the x-axis. Furthermore we also vary the amount of training data we use to train the classifiers from 10k to 1M training examples. It can be seen that when using the same features as BION, our prediction accuracy is compatible with the BION classifier. The main improvement of our classifier stems from the REP, AMSC and VAR features. Our best classifier is more than 11% more accurate than previous state-of-the-art BION classifier.

We identified the best classifier on a held-out set of 1000 ciphers, i.e. 20 ciphers for each cipher type. Here the three new features improve the VW-1M classifier from 50.9% accuracy to 56.0% accuracy, and the VW-100k classifier from 48.9% to 54.6%. Note that this held-out set is based on the exact same generator that we used to create the training data with. However, we also report the results of our method on the completely independently created ACA test set in Figure 4.

6 Conclusion

We presented a state-of-the art classifier for cipher type detection. The approach we present is easily extensible to cover more cipher types and allows incorporating new features.

Acknowledgements

We thank Taylor Berg-Kirkpatrick, Shu Cai, Bill Mason, Beáta Megyesi, Julian Schamper, and Megha Srivastava for their support and ideas. This work was supported by ARL/ARO (W911NF-10-1-0533) and DARPA (HR0011-12-C-0014).

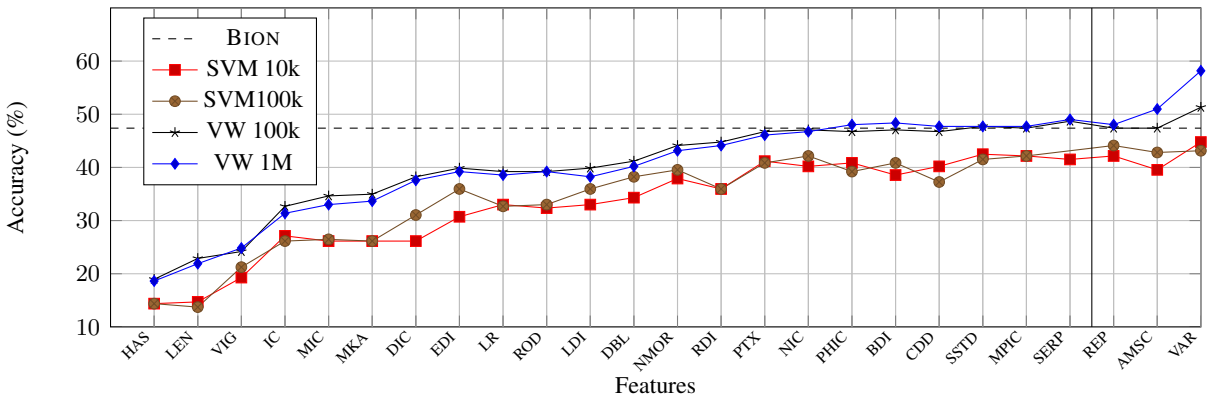


Figure 4: Classifier accuracy vs. training data and set of features used. From left to right more and more features are used, the x-axis shows which features are added. The feature names are described in Section 4. The features right of the vertical line are presented in this paper. The horizontal line shows the previous state-of-the art accuracy (BION) of 47.3%, we achieve 58.49%.

References

F.L. Bauer. 2010. *Decrypted Secrets: Methods and Maxims of Cryptology*. Springer.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32, October.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

William AR de Souza, Allan Tomlinson, and Luiz MS de Figueiredo. 2013. Cipher identification with a neural network.

John Langford, Lihong Li, and Alex Strehl. 2007. Vowpal Wabbit. `https://github.com/JohnLangford/vowpal_wabbit/wiki`.

Malte Nuhn, Julian Schamper, and Hermann Ney. 2013. Beam search for solving substitution ciphers. In *ACL (1)*, pages 1568–1576.

Sujith Ravi and Kevin Knight. 2011. Bayesian Inference for Zodiac and Other Homophonic Ciphers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 239–247, Stroudsburg, PA, USA, June. Association for Computational Linguistics.

Joint Learning of Chinese Words, Terms and Keywords

Ziqiang Cao[1] **Sujian Li**[1] **Heng Ji**[2]
[1]Key Laboratory of Computational Linguistics, Peking University, MOE, China
[2]Computer Science Department, Rensselaer Polytechnic Institute, USA
{ziqiangyeah, lisujian}@pku.edu.cn jih@rpi.edu

Abstract

Previous work often used a pipelined framework where Chinese word segmentation is followed by term extraction and keyword extraction. Such framework suffers from error propagation and is unable to leverage information in later modules for prior components. In this paper, we propose a four-level Dirichlet Process based model (DP-4) to jointly learn the word distributions from the corpus, domain and document levels simultaneously. Based on the DP-4 model, a sentence-wise Gibbs sampler is adopted to obtain proper segmentation results. Meanwhile, terms and keywords are acquired in the sampling process. Experimental results have shown the effectiveness of our method.

1 Introduction

For Chinese language which does not contain explicitly marked word boundaries, word segmentation (WS) is usually the first important step for many Natural Language Processing (NLP) tasks including term extraction (TE) and keyword extraction (KE). Generally, Chinese terms and keywords can be regarded as words which are representative of one domain or one document respectively. Previous work of TE and KE normally used the pipelined approaches which first conducted WS and then extracted important word sequences as terms or keywords.

It is obvious that the pipelined approaches are prone to suffer from error propagation and fail to leverage information for word segmentation from later stages. Here, we provide one example in the *disease* domain, to demonstrate the common problems in current pipelined approaches and propose the basic idea of our joint learning of words, terms and keywords.

Example: 血小板减少症(thrombocytopenia) 同(with) 类肝素(heparinoid) 有(have) 关系(relation).

This is a correctly segmented Chinese sentence. The document containing the example sentence mainly talks about the property of "类肝素(heparinoid)" which can be regarded as one keyword of the document. At the same time, the word 血小板减少症(thrombocytopenia) appears frequently in the *disease* domain and can be treated as a domain-specific term.

However, for such a simple sentence, current segmentation tools perform poorly. The segmentation result with the state-of-the-art Conditional Random Fields (CRFs) approach (Zhao et al., 2006) is as follows:

血小板(blood platelet) 减少(reduction) 症(symptom)
同类(of same kind) 肝(liver) 素有(always) 关系(relation)

where 血小板减少症 is segmented into three common Chinese words and 类肝素 is mixed with its neighbors.

In a text processing pipeline of WS, TE and KE, it is obvious that imprecise WS results will make the overall system performance unsatisfying. At the same time, we can hardly make use of domain-level and document-level information collected in TE and KE to promote the performance of WS. Thus, one question comes to our minds: can words, terms and keywords be jointly learned with consideration of all the information from the corpus, domain, and document levels?

Recently, the hierarchical Dirichlet process (HDP) model has been used as a smoothed bigram model to conduct word segmentation (Goldwater et al., 2006; Goldwater et al., 2009). Meanwhile, one strong point of the HDP based models is that they can model the diversity and commonality in multiple correlated corpora (Ren et al., 2008; Xu et al., 2008; Zhang et al., 2010; Li et al., 2012; Chang et al., 2014). Inspired by such existing work, we propose a four-level DP based model,

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1774–1778,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

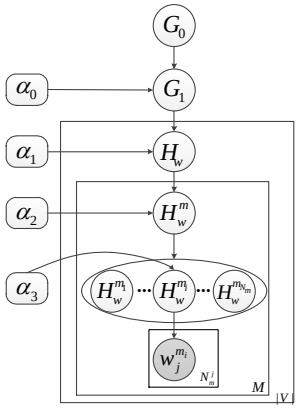


Figure 1: DP-4 Model

named DP-4, to adapt to three levels: corpus, domain and document. In our model, various DPs are designed to reflect the smoothed word distributions in the whole corpus, different domains and different documents. Same as the DP based segmentation models, our model can be easily used as a semi-supervised framework, through exerting on the corpus level the word distributions learned from the available segmentation results. Referring to the work of Mochihashi et al. (2009), we conduct word segmentation using a sentence-wise Gibbs sampler, which combines the Gibbs sampling techniques with the dynamic programming strategy. During the sampling process, the importance values of segmented words are measured in domains and documents respectively, and words, terms and keywords are jointly learned.

2 DP-4 Model

Goldwater et al. (2006) applied the HDP model on the word segmentation task. In essence, Goldwater's model can be viewed as a bigram language model with a unigram back-off. With the language model, word segmentation is implemented by a character-based Gibbs sampler which repeatedly samples the possible word boundary positions between two neighboring words, conditioned on the current values of all other words. However, Goldwater's model can be deemed as modeling the whole corpus only, and does not distinguish between domains and documents. To jointly learn the word information from the corpus, domain and document levels, we extend Goldwater's model by adding two levels (domain level and document level) of DPs, as illustrated in Figure 1.

2.1 Model Description

M DPs (H_w^m;$1 \leq m \leq M$) are designed specifically to word w to model the bigram distributions in each domain and these DPs share an overall base measure H_w, which is drawn from $DP(\alpha_0, G_1)$ and gives the bigram distribution for the whole corpus. Assuming the m^{th} domain includes N_m documents, we use $H_w^{m_j}$ ($1 \leq j \leq N_m$) to model the bigram distribution of the i^{th} document in the domain. Usually, given a domain, the bigram distributions of different documents are not conditionally independent and similar documents exhibit similar bigram distributions. Thus, the bigram distribution of one document is generated according to both the bigram distribution of the domain and the bigram distributions of other documents in the same domain. That is, $H_w^{m_j} \sim g(\alpha_3, H_w^m, H_w^{m-j})$ where H_w^{m-j} represents the bigram distributions of the documents in the m^{th} domain except the j^{th} document. Assuming the j^{th} document in the m^{th} domain contains N_m^j words, each word is drawn according to $H_w^{m_j}$. That is, $w_i^{m_j} \sim H_w^{m_j} (1 \leq i \leq N_m^j)$. Thus, our four-level DP model can be summarized formally as follows:

$$G_1 \sim DP(\alpha_0, G_0); H_w \sim DP(\alpha_1, G_1)$$
$$H_w^m \sim DP(\alpha_2, H_w); H_w^{m_j} \sim g\left(\alpha_3, H_w^m, H_w^{m-j}\right)$$
$$w_i^{m_j} | w_{i-1} = w \sim H_w^d$$

Here, we provide for our model the Chinese Restaurant Process (CRP) metaphor, which can create a partition of items into groups. In our model, the word type of the previous word w_{i-1} corresponds to a restaurant and the current word w_i corresponds to a customer. Each domain is analogous to a floor in a restaurant and a room denotes a document. Now, we can see that there are $|V|$ restaurants and each restaurant consists of M floors. The m^{th} floor contains N_m rooms and each room has an infinite number of tables with infinite seating capacity. Customers enter a specific room on a specific floor of one restaurant and seat themselves at a table with the label of a word type. Different from the standard HDP, each customer sits at an occupied table with probability proportional to both the numbers of customers already seated there and the numbers of customers with the same word type seated in the neighboring rooms, and at an unoccupied table with probability proportional to both the constant α_3 and the probability that the

customers with the same word type are seated on the same floor.

2.2 Model Inference

It is important to build an accurate G_0 which determines the prior word distribution $p_0(w)$. Similar to the work of Mochihashi et al. (2009), we consider the dependence between characters and calculate the prior distribution of a word w_i using the string frequency statistics (Krug, 1998):

$$p_0(w_i) = \frac{n_s(w_i)}{\sum n_s(.)} \tag{1}$$

where $n_s(w_i)$ counts the character string composed of w_i and the symbol "." represents any word in the vocabulary V.

Then, with the CRP metaphor, we can obtain the expected word unigram and bigram distributions on the corpus level according to G_1 and H_w:

$$p_1(w_i) = \frac{n(w_i) + \alpha_0 p_0(w_i)}{\sum n(.) + \alpha_0} \tag{2}$$

$$p_2(w_i|w_{i-1} = w) = \frac{n_w(w_i) + \alpha_1 p_1(w_i)}{\sum n_w(.) + \alpha_1} \tag{3}$$

where the subscript numbers indicate the corresponding DP levels. $n(w_i)$ denotes the number of w_i and $n_w(w_i)$ denotes the number of the bigram $< w, w_i >$ occurring in the corpus. Next, we can easily get the bigram distribution on the domain level by extending to the third DP.

$$p_3^m(w_i|w_{i-1} = w) = \frac{n_w^m(w_i) + \alpha_2 p_2(w_i|w_{i-1})}{\sum n_w^m(.) + \alpha_2} \tag{4}$$

where $n_w^m(w_i)$ is the number of the bigram $< w, w_i >$ occurring in the m^{th} domain.

To model the bigram distributions on the document level, it is beneficial to consider the influence of related documents in the same domain (Wan and Xiao, 2008). Here, we only consider the influence from the K most similar documents with a simple similarity metric $s(d_1, d_2)$ which calculates the Chinese character overlap ratio of two documents d_1 and d_2. Let d_m^j denote the j^{th} document in the m^{th} domain and $d_m^j[k](1 \le k \le K)$ the K most similar documents. d_m^j can be deemed to be "lengthened" by $d_m^j[k](1 \le k \le K)$. Therefore, we estimate the count of w_i in d_m^j as:

$$t_w^{d_m^j}(w_i) = n_w^{d_m^j}(w_i) + \sum_k s(d_m^j[k], d_m^j) n_w^{d_m^j[k]}(w_i) \tag{5}$$

where $n_w^{d_m^j[k]}(w_i)$ denotes the count of the bigram $< w, w_i >$ occurring in $d_m^j[k]$. Next, we model the bigram distribution in d_m^j as a DP with the base measure H_w^m :

$$p_4^{d_m^j}(w_i|w_{i-1} = w) = \frac{t_w^{d_m^j}(w_i) + \alpha_3 p_3^m(w_i|w_{i-1})}{\sum t_w^{d_m^j}(.) + \alpha_3} \tag{6}$$

With CRP, we can also easily estimate the unigram probabilities $p_3^m(w_i)$ and $p_4^{d_m^j}(w_i)$ respectively on the domain and document levels, through combining all the restaurants.

To measure whether a word is eligible to be a term, the score function $TH^m(\cdot)$ is defined as:

$$TH^m(w_i) = \frac{p_3^m(w_i)}{p_1(w_i)} \tag{7}$$

This equation is inspired by the work of Nazar (2011), which extracts terms with consideration of both the frequency in the domain corpus and the frequency in the general reference corpus. Similar to Eq. 7, we define the function $KH^{d_m^j}(\cdot)$ to judge whether w_i is an appropriate keyword.

$$KH^{d_m^j}(w_i) = \frac{p_4^{d_m^j}(w_i)}{p_1(w_i)} \tag{8}$$

During each sampling, we make use of Eqs. (7) and (8) to identify the most possible terms and keywords. Once a word is identified as a term or keyword, it will drop out of the sampling process in the following iterations. Its CRP explanation is that some customers (terms and keywords) find their proper tables and keep sitting there afterwards.

2.3 Sentence-wise Gibbs Sampler

The character-based Gibbs sampler for word segmentation (Goldwater et al., 2006) is extremely slow to converge, since there exists high correlation between neighboring words. Here, we introduce the sentence-wise Gibbs sampling technique as well as efficient dynamic programming strategy proposed by Mochihashi et al. (2009). The basic idea is that we randomly select a sentence in each sampling process and use the Viterbi algorithm (Viterbi, 1967) to find the optimal segmentation results according to the word distributions derived from other sentences. Different from Mochihashi's work, once terms or keywords are

identified, we do not consider them in the segmentation process. Due to space limitation, the algorithm is not detailed here and can be referred in (Mochihashi et al., 2009).

3 Experiment

3.1 Data and Setting

It is indeed difficult to find a standard evaluation corpus for our joint tasks, especially in different domains. As a result, we spent a lot of time to collect and annotate a new corpus [1] composed of ten domains (including *Physics*, *Computer*, *Agriculture*, *Sports*, *Disease*, *Environment*, *History*, *Art*, *Politics* and *Economy*) and each domain is composed of 200 documents. On average each document consists of about 4800 Chinese characters. For these 2000 documents, three annotators have manually checked the segmented words, terms and keywords as the gold standard results for evaluation. As we know, there exists a large amount of manually-checked segmented text for the general domain, which can be used as the training data for further segmentation. As with other nonparametric Bayesian models (Goldwater et al., 2006; Mochihashi et al., 2009), our DP-4 model can be easily amenable to semi-supervised learning by imposing the word distributions of the segmented text on the corpus level. The news texts provided by Peking University (named PKU corpus)[2] is used as the training data. This corpus contains about 1,870,000 Chinese characters and has been manually segmented into words.

In our experiments, the concentration coefficient (α_0) is finally set to 20 and the other three ($\alpha_{1\sim3}$) are set to 15. The parameter K which controls the number of similar documents is set to 3.

3.2 Performance Evaluation

The following baselines are implemented for comparison of segmentation results: (1) Forward maximum matching (FMM) algorithm with a vocabulary compiled from the PKU corpus; (2) Reverse maximum matching (RMM) algorithm with the compiled vocabulary; (3) Conditional Random Fields (CRFs)[3] based supervised algorithm trained from the PKU corpus; (4) HDP based semi-supervised algorithm (Goldwater et al., 2006) using the PKU corpus. The strength of Mochihashi et al. (2009)'s NPYLM based segmentation model is its speed due to the sentence-wise sampling technique, and its performance is similar to Goldwater et al. (2006)'s model. Thus, we do not consider the NPYLM based model for comparison here. Then, the segmentation results of FMM, RMM, CRF, and HDP methods are used respectively for further extracting terms and keywords. We use the mutual information to identify the candidate terms or keywords composed of more than two segmented words. As for DP-4, this recognition process has been done implicitly during sampling. To measure the candidate terms or keywords, we refer to the metric in Nazar (2011) to calculate their importance in some specific domain or document.

The metrics of F_1 and the out-of-vocabulary Recall (OOV-R) are used to evaluate the segmentation results, referring to the gold standard results. The second and third columns of Table 1 show the F_1 and OOV-R scores averaged on the 10 domains for all the compared methods. Our method significantly outperforms FMM, RMM and HDP according to t-test (p-value ≤ 0.05). From the segmentation results, we can see that the FMM and RMM methods are highly dependent on the compiled vocabulary and their identified OOV words are mainly the ones composed of a single Chinese character. The HDP method is heavily influenced by the segmented text, but it also exhibits the ability of learning new words. Our method only shows a slight advantage over the CRF approach. We check our segmentation results and find that the performance of the DP-4 model is depressed by the identified terms and keywords which may be composed of more than two words in the gold standard results, because the DP-4 model always treats the term or keyword as a single word. For example, in the gold standard, "岭南文化((Lingnan Culture)" is segmented into two words "岭南" and "文化", "数据接口(data interface)" is segmented into "数据" and "接口" and so on. In fact, our segmentation results correctly treat "岭南文化" and "数据接口" as words.

To evaluate the TE and KE performance, the top 50 (TE-50) and 100 (TE-100) accuracy are measured for the identified terms of one domain, while the top 5 (KE-5) and 10 (KE-10) accuracy for the keywords in one document, are shown in the right four columns of Table 1. We can see that DP-

[1] Nine domains are from http://www.datatang.com/data/44139 and we add an extra *Disease* domain.

[2] http://icl.pku.edu.cn

[3] We adopt CRF++(http://crfpp.googlecode.com/svn/trunk/doc/index.html)

4 performs significantly better than all the other methods in TE and KE results.

As for the ten domains, we find our approach behaves much better than the other approaches on the following three domains: *Disease*, *Physics* and *Computer*. It is because the language of these three domains is much different from that of the general domain (PKU corpus), while the rest domains are more similar to the general domain.

Method	F1	OOV-R	TE-50	TE-100	KE-5	KE-10
FMM	0.796	0.136	0.420	0.360	0.476	0.413
RMM	0.794	0.136	0.424	0.352	0.478	0.414
HDP	0.808	0.356	0.672	0.592	0.552	0.506
CRF	0.817	0.330	0.624	0.560	0.543	0.511
DP-4	**0.821**	**0.374**	**0.704**	**0.640**	**0.571**	**0.545**

Table 1: Comparison of WS, TE and KE Performance (averaged on the 10 domains).

4 Conclusion

This paper proposes a four-level DP based model to construct the word distributions from the corpus, domain and document levels simultaneously, through which Chinese words, terms and keywords can be learned jointly and effectively. In the future, we plan to explore how to combine more features such as part-of-speech tags into our model.

Acknowledgments

We thank the three anonymous reviewers for their helpful comments. This work was partially supported by National High Technology Research and Development Program of China (No. 2012AA011101), National Key Basic Research Program of China (No. 2014CB340504), National Natural Science Foundation of China (No. 61273278), and National Key Technology R&D Program (No: 2011BAH10B04-03). The contact author of this paper, according to the meaning given to this role by Peking University, is Sujian Li.

References

Baobao Chang, Wenzhe Pei, and Miaohong Chen. 2014. Inducing word sense with automatically learned hidden concepts. In *Proceedings of COLING 2014*, pages 355–364, Dublin, Ireland, August.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 673–680.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.

Manfred Krug. 1998. String frequency: A cognitive motivating factor in coalescence, language processing, and linguistic change. *Journal of English Linguistics*, 26(4):286–320.

Jiwei Li, Sujian Li, Xun Wang, Ye Tian, and Baobao Chang. 2012. Update summarization using a multi-level hierarchical dirichlet process model. In *Proceedings of Coling 2012*, pages 1603–1618, Mumbai, India.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 100–108.

Rogelio Nazar. 2011. A statistical approach to term extraction. *IJES, International Journal of English Studies*, 11(2):159–182.

Lu Ren, David B. Dunson, and Lawrence Carin. 2008. The dynamic hierarchical dirichlet process. In *Proceedings of the 25th international conference on Machine learning*, pages 824–831.

Andrew J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, pages 260–269.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, volume 8, pages 855–860.

Tianbing Xu, Zhongfei Zhang, Philip S. Yu, and Bo Long. 2008. Dirichlet process based evolutionary clustering. In *ICDM'08*, pages 648–657.

Jianwen Zhang, Yangqiu Song, Changshui Zhang, and Shixia Liu. 2010. Evolutionary hierarchical dirichlet processes for multiple correlated time-varying corpora. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1079–1088, New York, NY, USA.

Hai Zhao, Chang-Ning Huang, and Mu Li. 2006. An improved chinese word segmentation system with conditional random field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, volume 1082117.

Cross-Lingual Part-of-Speech Tagging through Ambiguous Learning

Guillaume Wisniewski Nicolas Pécheux Souhir Gahbiche-Braham François Yvon

Université Paris Sud
LIMSI-CNRS
91 403 ORSAY CEDEX, France
{wisniews,pecheux,souhir,yvon}@limsi.fr

Abstract

When Part-of-Speech annotated data is scarce, e.g. for under-resourced languages, one can turn to cross-lingual transfer and crawled dictionaries to collect partially supervised data. We cast this problem in the framework of *ambiguous learning* and show how to learn an accurate history-based model. Experiments on ten languages show significant improvements over prior state of the art performance.

1 Introduction

In the past two decades, supervised Machine Learning techniques have established new performance standards for many NLP tasks. Their success however crucially depends on the availability of annotated in-domain data, a not so common situation. This means that for many application domains and/or less-resourced languages, alternative ML techniques need to be designed to accommodate unannotated or partially annotated data.

Several attempts have recently been made to mitigate the lack of annotated corpora using parallel data pairing a (source) text in a resource-rich language with its counterpart in a less-resourced language. By transferring labels from the source to the target, it becomes possible to obtain noisy, yet useful, annotations that can be used to train a model for the target language in a *weakly supervised* manner. This research trend was initiated by Yarowsky et al. (2001), who consider the transfer of POS and other syntactic information, and further developed in (Hwa et al., 2005; Ganchev et al., 2009) for syntactic dependencies, in (Padó and Lapata, 2009; Kozhevnikov and Titov, 2013; van der Plas et al., 2014) for semantic role labeling and in (Kim et al., 2012) for named-entity recognition, to name a few.

Assuming that labels can actually be projected across languages, these techniques face the issue of extending standard supervised techniques with partial and/or uncertain labels in the presence of alignment noise. In comparison to the early approach of Yarowsky et al. (2001) in which POS are directly transferred, subject to heuristic filtering rules, recent works consider the integration of softer constraints using expectation regularization techniques (Wang and Manning, 2014), the combination of alignment-based POS transfer with additional information sources such as dictionaries (Li et al., 2012; Täckström et al., 2013) (Section 2), or even the simultaneous use of both techniques (Ganchev and Das, 2013).

In this paper, we reproduce the weakly supervised setting of Täckström et al. (2013). By recasting this setting in the framework of ambiguous learning (Bordes et al., 2010; Cour et al., 2011) (Section 3), we propose an alternative learning methodology and show that it improves the state of the art performance on a large array of languages (Section 4). Our analysis of the remaining errors suggests that in cross-lingual settings, improvements of error rates can have multiple causes and should be looked at with great care (Section 4.2).

All tools and resources used in this study are available at `http://perso.limsi.fr/wisniews/ambiguous`.

2 Projecting Labels across Aligned Corpora

Projecting POS information across languages relies on a rather strong assumption that morpho-syntactic categories in the source language can be directly related to the categories in the target language, which might not always be warranted (Evans and Levinson, 2009; Broschart, 2009). The universal reduced POS tagset proposed by Petrov et al. (2012) defines an operational, albeit rather empirical, ground to perform this mapping. It is made of the following 12 categories: NOUN (nouns), VERB (verbs), ADJ (ad-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1779–1785,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

	ar	cs	de	el	es	fi	fr	id	it	sv
% of test covered tokens (type)	83.2	93.2	95.6	97.4	96.7	83.0	98.3	90.5	95.8	95.3
% of test correctly covered token (type)	72.9	94.2	93.7	92.9	93.8	93.6	92.1	89.6	93.6	94.1
avg. number of labels per token (type)	2.1	1.3	1.3	1.3	1.3	1.4	1.3	1.2	1.3	1.3
avg. number of labels per token (type+token)	1.7	1.1	1.1	1.1	1.1	1.2	1.2	1.1	1.1	1.1
% of aligned tokens	53.0	77.8	66.7	69.3	74.0	73.1	64.7	81.6	72.2	79.9
% of token const. violating type const.	2.5	16.0	15.8	21.4	16.9	14.3	16.1	19.3	17.5	13.6
% informative token const.	79.7	27.5	15.7	29.8	21.3	36.0	25.5	16.2	28.2	26.4

Table 1: Interplay between token and type constraints on our training parallel corpora. 'Informative' token constraints correspond to tokens for which (a) a POS is actually transfered and (b) type constraints do not disambiguate the label, but type+token constraints do.

jectives), ADV (adverbs), PRON (pronouns), DET (determiners and articles), ADP (prepositions and postpositions), NUM (numerals), CONJ (conjunctions), PRT (particles), '.' (punctuation marks) and X (a catch-all for other categories). These labels have been chosen for their stability across languages and for their usefulness in various multilingual applications. In the rest of this work, all annotations are mapped to this universal tagset.

Transfer-based methods have shown to be very effective, even if projected labels only deliver a noisy supervision, due to tagging (of the source language) and other alignment errors (Yarowsky et al., 2001). While this uncertainty can be addressed in several ways, recent works have proposed to combine projected labels with monolingual information in order to filter out invalid label sequences (Das and Petrov, 2011; Täckström et al., 2013). In this work we follow Täckström et al. (2013) and use two families of constraints:

Token constraints rely on word alignments to project labels of source words to target words through alignment links. Table 1 shows that, depenening on the language, only $50-80\%$ of the target tokens would benefit from label transfer.

Type constraints rely on a tag dictionary to define the set of possible tags for each word type. Type constraints reduce the possible labels for a given word and help filtering out cross-lingual transfer errors (up to 20%, as shown in Table 1). As in (Täckström et al., 2013), we consider two different dictionaries. The first one is extracted automatically from `Wiktionary`,[1] using the method of (Li et al., 2012). The second tag dictionary is built by using for each word the two most frequently projected POS labels from the training data.[2] In contrast to Täckström et al. (2013) we use the intersection[3] of the two type constraints instead of their union. Table 1 shows the precision and recall of the resulting constraints on the test data.

These two information sources are merged according to the rules of Täckström et al. (2013). These rules assume that type constraints are more reliable than token constraints and should take precedence: by default, a given word is associated to the set of possible tags licensed type constraints; additionally, when a POS tag can be projected through alignment *and* also satisfies the type constraints, then it is actually projected, thereby providing a full (yet noisy) supervision.

As shown in Table 1, token and type constraints complement each other effectively and greatly reduce label ambiguity. However, the transfer method sketched above associates each target word with a set of possible labels, of which only one is true. This situation is less favorable than standard supervised learning in which one unique gold label is available for each occurrence. We describe in the following section how to learn from this *ambiguous supervision* information.

3 Modeling Sequences under Ambiguous Supervision

We use a history-based model (Black et al., 1992) with a LaSO-like training method (Daumé and Marcu, 2005). History-based models reduce structured prediction to a sequence of multi-class classification problems. The prediction of a complex structure (here, a sequence of POS tags) is thus modeled as a sequential decision problem: at each

[1] `http://www.wiktionary.org/`

[2] This heuristic is similar to the way Täckström et al. (2013) filter the tag distribution with a threshold to build the projected type constraints.

[3] If the intersection is empty we use the constraints from `Wiktionary` first, if also empty, the projected constraints then, and by default the whole tag set.

position in the sequence, a multiclass classifier is used to make a decision, using features that describe both the input structure and the history of past decisions (i.e. the partially annotated sequence).

Let $\mathbf{x} = (\mathbf{x}_i)_{i=1}^n$ denote the observed sequence and $\mathcal{Y}$ be the set of possible labels (in our case the 12 universal POS tags). Inference consists in predicting labels one after the other using, for instance, a linear model:

$$y_i^* = \arg\max_{y \in \mathcal{Y}} \langle \mathbf{w} | \phi(\mathbf{x}, i, y, h_i) \rangle \quad (1)$$

where $\langle \cdot | \cdot \rangle$ is the standard dot product operation, y_i^* the predicted label for position i, $\mathbf{w}$ the weight vector, $h_i = y_1^*, ..., y_{i-1}^*$ the history of past decisions and ϕ a joint feature map. Inference can therefore be seen as a greedy search in the space of the $\#\{\mathcal{Y}\}^n$ possible labelings of the input sequence. Trading off the global optimality of inference for additional flexibility in the design of features and long range dependencies between labels has proved useful for many sequence labeling tasks in NLP (Tsuruoka et al., 2011).

The training procedure, sketched in Algorithm 1, consists in performing inference on each input sentence and correcting the weight vector each time a wrong decision is made. Importantly (Ross and Bagnell, 2010), the history used during training has to be made of the previous predicted labels so that the training samples reflect the fact that the history will be imperfectly known at test time.

This *reduction* of sequence labeling to multiclass classification allows us to learn a sequence model in an ambiguous setting by building on the theoretical results of Bordes et al. (2010) and Cour et al. (2011). The decision about the correctness of a prediction and the weight updates can be adapted to the amount of supervision information that is available.

Full Supervision In a fully supervised setting, the correct label is known for each word token: a decision is thus considered wrong when this gold label is not predicted. In this case, a standard perceptron update is performed:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \phi(\mathbf{x}, i, y_i^*, h_i) + \phi(\mathbf{x}, i, \hat{y}_i, h_i) \quad (2)$$

where y_i^* and $\hat{y}_i$ are the predicted and the gold label, respectively. This update is a stochastic gradient step that increases the score of the gold label while decreasing the score of the predicted label.

Ambiguous Supervision During training, each observation i is now associated with a set of possible labels, denoted by $\hat{\mathcal{Y}}_i$. In this case, a decision is considered wrong when the predicted label is not in $\hat{\mathcal{Y}}_i$ and the weight vector is updated as follows:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \phi(\mathbf{x}, i, y_i^*, h_i) + \sum_{\hat{y}_i \in \hat{\mathcal{Y}}_i} \phi(\mathbf{x}, i, \hat{y}_i, h_i) \quad (3)$$

Compared to (2), this rule uniformly increases the scores of all the labels in $\hat{\mathcal{Y}}_i$.

It can be shown (Bordes et al., 2010; Cour et al., 2011), under mild assumptions (namely that two labels never systematically co-occur in the supervision information), that the update rule (3) enables to learn a classifier in an ambiguous setting, as if the gold labels were known. Intuitively, as long as two labels are not systematically co-occurring in $\hat{\mathcal{Y}}$, updates will reinforce the correct labels more often than the spurious ones; at the end of training, the highest scoring label should therefore be the correct one.

Algorithm 1 Training algorithm. In the ambiguous setting, $\hat{\mathcal{Y}}_i$ contains all possible labels; in the supervised setting, it only contains the gold label.

$\mathbf{w}_0 \leftarrow \mathbf{0}$
for $t \in [\![1, T]\!]$ **do**
  Randomly pick example $\mathbf{x}, \hat{\mathbf{y}}$
  $h \leftarrow$ empty list
  for $i \in [\![1, n]\!]$ **do**
    $y_i^* = \arg\max_{y \in \mathcal{Y}} \langle \mathbf{w}_t | \phi(\mathbf{x}, i, y, h_i) \rangle$
    if $y_i^* \notin \hat{\mathcal{Y}}_i$ **then**
      $\mathbf{w}_{t+1} \leftarrow \text{update}(\mathbf{w}_t, \mathbf{x}, i, \hat{\mathcal{Y}}_i, y_i^*, h_i)$
    end if
    push(y_i^*, h)
  end for
end for
return $\frac{1}{T} \sum_{t=1}^T \mathbf{w}_t$

4 Empirical Study

Datasets Our approach is evaluated on 10 languages that present very different characteristics and cover several language families.[4] In all our experiments we use English as the source language. Parallel sentences[5] are aligned with the standard

[4] Resources considered in the related works are not freely available, which prevents us from presenting a more complete comparison.

[5] All resources and features used in our experiments are thoroughly documented in the supplementary material.

	ar	cs	de	el	es	fi	fr	id	it	sv
HBAL	**27.9**	**10.4**	**8.8**	**8.1**	**8.2**	13.3	**10.2**	**11.3**	**9.1**	**10.1**
Partially observed CRF	33.9	11.6	12.2	10.9	10.7	**12.9**	11.6	16.3	10.4	11.6
HBSL	—	1.5	5.0	—	2.4	5.9	3.5	4.8	2.8	3.8
HBAL + matched POS	24.1	7.6	8.0	7.3	7.4	12.2	7.4	9.8	8.3	8.8
(Ganchev and Das, 2013)	49.9	19.3	9.6	9.4	12.8	—	12.5	—	10.1	10.8
(Täckström et al., 2013)	—	18.9	9.5	10.5	10.9	—	11.6	—	10.2	11.1
(Li et al., 2012)	—	—	14.2	20.8	13.6	—	—	—	13.5	13.9

Table 2: Error rate (in %) achieved by the method described in Sec. 3 trained in an ambiguous (HBAL) or in a supervised setting (HBSL), a partially observed CRF and different state-of-the-art results.

MOSES pipeline, using the intersection heuristic that only retains the most reliable alignment links.

The English side of the bitext is tagged using a standard linear CRF trained on the Penn Treebank. Tags are then transferred to the target language using the procedure described in Section 2. For each language, we train a tagger using the method described in Section 3 with $T = 100\,000$ iterations[6] using a feature set similar to the one of Li et al. (2012) and Täckström et al. (2013). The baseline system is our reimplementation of the partially observed CRF model of Täckström et al. (2013). Evaluation is carried out on the test sets of treebanks for which manual gold tags are known. For Czech and Greek, we use the CoNLL'07 Shared Task on Dependency Parsing; for Arabic, the Arabic Treebank; and otherwise the data of the Universal Dependency Treebank Project (McDonald et al., 2013). Tagging performance is evaluated with the standard error rate.

4.1 Results

Table 2 summarizes the performance achieved by our method trained in the ambiguous setting (HBAL) and by our re-implementation of the partially supervised CRF baseline. As an upper bound, we also report the score of our method when trained in a supervised (HBSL) settings considering the training part of the various treebanks, when it is available.[7] For the sake of comparison, we also list the *best scores* of previous studies. Note, however, that a direct comparison with these results is not completely fair as these systems were not trained and evaluated with the same exact resources (corpora,[8] type constraints, alignments, etc). Also note that the state-of-the-art scores have been achieved by different models, which have been selected based on their scores on the test set and not on a validation set.[9]

Experimental results show that HBAL significantly outperforms, on all considered languages but one, the partially observed CRF that was trained and tested in the same setting.

4.2 Discussion

The performance of our new method still falls short of the performance of a fully supervised POS tagger: for instance, in Spanish, full supervision reduces the error rate by a factor of 4. A fine-grained error analysis shows that many errors of HBAL directly result from the fact that, contrary to the fully supervised learner HBSL, our ambiguous setting suffers from a train/test mismatch, which has two main consequences. First, the train and test sets do not follow exactly the same normalization and tokenization conventions, which is an obvious source of mistakes. Second, and more importantly, many errors are caused by systematic differences between the test tags and the supervised tags (i.e. the English side of the bitext and `Wiktionary`). While some of these differences are linguistically well-justified and reflect fundamental differences in the language structure and usage, others seem to be merely due to arbitrary annotation conventions.

For instance, in Greek, proper names are labeled

[6] Preliminary experiments showed that increasing the number of iterations T in Algorithm 1 has no significant impact.

[7] In this setting, HBSL implements an averaged perceptron, and achieves results that are similar to those obtained with standard linear CRF.

[8] The test sets are only the same for Czech, Greek and Swedish.

[9] The partially observed CRF is the best model in (Täckström et al., 2013) only for German (de), Greek (el) and Swedish (sv), and uses only type constraints extracted from `Wiktionary`.

either as X (when they refer to a foreigner *and* are not transliterated) or as NOUN (in all other cases), while they are always labeled as NOUN in English. In French and in Greek, contractions of a preposition and a determiner such as '*στο*' ('*σε το*', meaning 'to the') or '*aux*' ('*à les*' also meaning 'to the') are labeled as ADP in the Universal Dependency Treebank but as DET in `Wiktionary` and are usually aligned with a determiner in the parallel corpora. In the Penn Treebank, quantifiers like 'few' or 'little' are generally used in conjunction with a determiner ('a few years', 'a little parable', ...) and labeled as ADJ; the corresponding Spanish constructions lack an article ('*mucho tempio*', '*pocos años*', ...) and the quantifiers are therefore labeled as DET. Capturing such subtle differences is hardly possible without prior knowledge and specifically tailored features.

This annotation mismatch problem is all the more important in settings like ours, that rely on several, independently designed, information sources, which follow contradictory annotation conventions and for which the mapping to the universal tagset is actually error-prone (Zhang et al., 2012). To illustrate this point, we ran three additional experiments to assess the impact of the train/test mismatch.

We first designed a control experiment in which the type constraints were manually completed with the gold labels of the most frequent errors of HBAL. These errors generally concern function words and can be assumed to result from systematic differences in the annotations rather than prediction errors. For instance, for French the type constraints for '*du*', '*des*', '*au*' and '*aux*' were corrected from DET to ADP. The resulting model, denoted 'HBAL + matched POS' in Table 2, significantly outperforms HBAL, stressing the divergence in the different annotation conventions.

Additionally, in order to approximate the ambiguous setting train/test mismatch, we learn two fully supervised Spanish taggers on the same training data as HBAL, using two different strategies to obtain labeled data. We first use HBSL (which was trained on the treebank) to automatically label the target side of the parallel corpus. In this setting, the POS tagger is trained with data from a different domain, but labeled with the same annotation scheme as a the test set. Learning with this fully supervised data yields an error rate of 4.2% for Spanish, almost twice as much as HBSL, bringing into light the impact of domain shift. We then use a generic tagger, FREELING,[10] to label the training data, this time with possible additional inconsistent annotations. The corresponding error rate for Spanish was 6.1%, to be compared with the 8.2% achieved by HBAL. The last two control experiments show that many of the remaining labeling errors seem to be due to domain and convention mismatches rather to the transfer/ambiguous setting, as supervised models also suffer from very similar conditions.

These observations show that the evaluation of transfer-based methods suffer from several biases. Their results must therefore be interpreted with great care.

5 Conclusion

In this paper, we have presented a novel learning methodology to learn from ambiguous supervision information, and used it to train several POS taggers. Using this method, we have been able to achieve performance that surpasses the best reported results, sometimes by a wide margin. Further work will attempt to better analyse these results, which could be caused by several subtle differences between HBAL and the baseline system. Nonetheless, these experiments confirm that cross-lingual projection of annotations have the potential to help in building very efficient POS taggers with very little monolingual supervision data. Our analysis of these results also suggests that, for this task, additional gains might be more easily obtained by fixing systematic biases introduced by conflicting mappings between tags or by train/test domain mismatch than by designing more sophisticated weakly supervised learners.

Acknowledgments

We wish to thank Thomas Lavergne and Alexandre Allauzen for early feedback and for providing us with the partially observed CRF implementation. We also thank the anonymous reviewers for their helpful comments.

References

Ezra Black, Fred Jelinek, John Lafferty, David M. Magerman, Robert Mercer, and Salim Roukos. 1992. Towards history-based grammars: Using

[10] `http://nlp.lsi.upc.edu/freeling/`

richer models for probabilistic parsing. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, pages 134–139, Stroudsburg, PA, USA. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, and Jason Weston. 2010. Label ranking under ambiguous supervision for learning semantic correspondences. In *ICML*, pages 103–110.

Jürgen Broschart. 2009. Why Tongan does it differently: Categorial distinctions in a language without nouns and verbs. *Linguistic Typology*, 1:123–166, 10.

Timothee Cour, Ben Sapp, and Ben Taskar. 2011. Learning from partial labels. *Journal of Machine Learning Research*, 12:1501–1536, July.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 600–609, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hal Daumé, III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 169–176, New York, NY, USA. ACM.

Nicholas Evans and Stephen C. Levinson. 2009. The myth of language universals: Language diversity and its importance for cognitive science. *Behavioral and Brain Sciences*, 32:429–448, 10.

Kuzman Ganchev and Dipanjan Das. 2013. Cross-lingual discriminative learning of sequence models with posterior regularization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1996–2006, Seattle, Washington, USA, October. Association for Computational Linguistics.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 369–377, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Nat. Lang. Eng.*, 11(3):311–325, September.

Sungchul Kim, Kristina Toutanova, and Hwanjo Yu. 2012. Multilingual named entity recognition using parallel data and metadata from wikipedia. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 694–702, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mikhail Kozhevnikov and Ivan Titov. 2013. Cross-lingual transfer of semantic role labeling models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1190–1200, Sofia, Bulgaria, August. Association for Computational Linguistics.

Shen Li, João V. Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1389–1398, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria, August. Association for Computational Linguistics.

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection of semantic roles. *J. Artif. Int. Res.*, 36(1):307–340, September.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Stéphane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *AISTATS*, pages 661–668.

Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Kazama. 2011. Learning with lookahead: Can history-based models rival globally optimized models? In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 238–246, Portland, Oregon, USA, June. Association for Computational Linguistics.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

Lonneke van der Plas, Marianna Apidianaki, and Chenhua Chen. 2014. Global methods for cross-lingual semantic role and predicate labelling. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1279–1290, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Mengqiu Wang and Christopher D. Manning. 2014. Cross-lingual projected expectation regularization for weakly supervised learning. *Transactions of the ACL*, 2:55–66, February.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yuan Zhang, Roi Reichart, Regina Barzilay, and Amir Globerson. 2012. Learning to map into a universal pos tagset. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1368–1378, Stroudsburg, PA, USA. Association for Computational Linguistics.

Comparing Representations of Semantic Roles for String-To-Tree Decoding

Marzieh Bazrafshan and **Daniel Gildea**
Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

We introduce new features for incorporating semantic predicate-argument structures in machine translation (MT). The methods focus on the completeness of the semantic structures of the translations, as well as the order of the translated semantic roles. We experiment with translation rules which contain the core arguments for the predicates in the source side of a MT system, and observe that using these rules significantly improves the translation quality. We also present a new semantic feature that resembles a language model. Our results show that the language model feature can also significantly improve MT results.

1 Introduction

In recent years, there have been increasing efforts to incorporate semantics in statistical machine translation (SMT), and the use of predicate-argument structures has provided promising improvements in translation quality. Wu and Fung (2009) showed that shallow semantic parsing can improve the translation quality in a machine translation system. They introduced a two step model, in which they used a semantic parser to rerank the translation hypotheses of a phrase-based system. Liu and Gildea (2010) used semantic features for a tree-to-string syntax based SMT system. Their features modeled deletion and reordering for source side semantic roles, and they improved the translation quality. Xiong et al. (2012) incorporated the semantic structures into phrase-based SMT by adding syntactic and semantic features to their translation model. They proposed two discriminative models which included features for predicate translation and argument reordering from source to target side. Bazrafshan and Gildea (2013) used semantic structures in a string-to-tree translation system by extracting translation rules enriched with semantic information, and showed that this can improve the translation quality. Li et al. (2013) used predicate-argument structure reordering models for hierarchical phrase-based translation, and they used linguistically motivated constraints for phrase translation.

In this paper, we experiment with methods for incorporating semantics in a string-to-tree MT system. These methods are designed to model the order of translation, as well as the completeness of the semantic structures. We extract translation rules that include the complete semantic structure in the source side, and compare that with using semantic rules for the target side predicates. We present a method for modeling the order of semantic role sequences that appear spread across multiple syntax-based translation rules, in order to overcome the problem that a rule representing the entire semantic structure of a predicate is often too large and too specific to apply to new sentences during decoding. For this method, we compare the verb-specific roles of PropBank and the more general thematic roles of VerbNet.

These essential arguments of a verbal predicate are called the *core* arguments. Standard syntax-based MT is incapable of ensuring that the target translation includes all of the core arguments of a predicate that appear in the source sentence. To encourage the translation of the likely core arguments, we follow the work of Bazrafshan and Gildea (2013), who use special translation rules with complete semantic structures of the predicates in the target side of their MT system. Each of these rules includes a predicate and all of its core arguments. Instead of incorporating only the target side semantic rules, we extract the special rules for both the source and the target sides, and compare the effectiveness of adding these rules to

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1786–1791,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

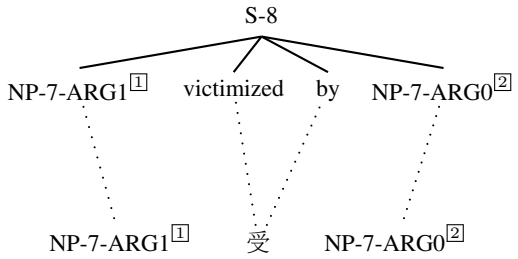


Figure 1: A complete semantic rule (Bazrafshan and Gildea (2013)).

the system separately and simultaneously.

Besides the completeness of the arguments, it is also important for the arguments to appear in the correct order. Our second method is designed to encourage correct order of translation for both the core and the non-core roles in the target sentence. We designed a new feature that resembles the language model feature in a standard MT system. We train a n-gram language model on sequences of semantic roles, by treating the semantic roles as the *words* in what we call the *semantic language*. Our experimental results show that the language model feature significantly improves translation quality.

Semantic Role Labeling (SRL): We use semantic role labelers to annotate the training data that we use to extract the translation rules. For target side SRL, the role labels are attached to the nonterminal nodes in the syntactic parse of each sentence. For source side SRL, the labels annotate the spans from the source sentence that they cover. We train our semantic role labeler using two different standards: Propbank (Palmer et al., 2005) and VerbNet (Kipper Schuler, 2005).

PropBank annotates the Penn Treebank with predicate-argument structures.It use generic labels (such as Arg0, Arg1, etc.) which are defined specifically for each verb. We trained a semantic role labeler on the annotated Penn Treebank data and used the classifier to tag our training data.

VerbNet is a verb lexicon that categorizes English verbs into hierarchical classes, and annotates them with thematic roles for the arguments that they accept. Since the thematic roles use more meaningful labels (e.g. Agent, Patient, etc.), a language model trained on VerbNet labels may be more likely to generalize across verbs than one trained on PropBank labels. It may also provide more information, since VerbNet has a larger set of labels than PropBank. To train the semantic role labeler on VerbNet, we used the mappings

$$\begin{array}{rl} A \rightarrow BC & c_0 \\ \quad [B,i,j] & c_1 \\ \quad [C,j,k] & c_2 \\ \hline \quad [A,i,k] & c_0 + c_1 + c_2 \end{array}$$

Figure 2: A deduction step in our baseline decoder

provided by the SemLink project (Palmer, 2009) to annotate the Penn Treebank with the VerbNet roles. These mappings map the roles in PropBank to the thematic roles of VerbNet. When there is no mapping for a role, we keep the role from Propbank.

2 Using Semantics in Machine Translation

In this section, we present our techniques for incorporating semantics in MT: source side semantic rules, and the semantic language model.

2.1 Source Side Semantic Rules

Bazrafshan and Gildea (2013) extracted translation rules that included a predicate and all of its arguments from the target side, and added those rules to the baseline rules of their string-to-tree MT system. Figure 1 shows an example of such rules, which we refer to as *complete semantic rules*. The new rules encourage the decoder to generate translations that include all of the semantic roles that appear in the source sentence.

In this paper, we use the same idea to extract rules from the semantic structures of the source side. The complete semantic rules consist of the smallest fragments of the combination of GHKM (Galley et al., 2004) rules that include one predicate and all of its core arguments that appear in the sentence. Rather than keeping the predicate and argument labels attached to the non-terminals, we remove those labels from our extracted semantic rules, to keep the non-terminals in the semantic rules consistent with the non-terminals of the baseline GHKM rules. This is also important when using both the source and the target semantic rules (i.e. Chinese and English rules), as it has been shown that there are cross lingual mismatches between Chinese and English semantic roles in bilingual sentences (Fung et al., 2007).

We extract a complete semantic rule for each verbal predicate of each sentence pair in the training data. To extract the target side complete semantic rules, using the target side SRL anno-

$A \rightarrow BC$ *to space*	c_0 (x1 x2 Destination)
$[B, i, j, (Agent,)]$	c_1
$[C, j, k, (PRED\_bring, Theme,)]$	c_2
$[A, i, k,$ (Agent, PRED_bring,-*-, Theme, Destination)$]$	$c_0 + c_1 + c_+$ + LMcost(Agent, PRED_bring,-*-, Theme, Destination)

Figure 3: A deduction step in the semantic language model method.

tated training data, we follow the general GHKM method, and modify it to ensure that each *frontier node* (Galley et al., 2004) in a rule includes either all or none of the semantic role labels (i.e. the predicate and all of its present core arguments) in its descendants in the target side tree. The resulting rule then includes the predicate and all of its arguments. We use the source side SRL annotated training data to extract the source side semantic rules. Since the annotations specify the spans of the semantic roles, we extract the semantic rules by ensuring that the span of the root (in the target side) of the extracted rule covers all of the spans of the roles in the predicate-argument structure.

The semantic rules are then used together with the original GHKM rules. We add a binary feature to distinguish the semantic rules from the rest. We experiment with adding the semantic rules from the source side, and compare that with adding semantic rules of both the source and the target side.

In all of the experiments in this paper, we use a string-to-tree decoder which uses a CYK style parser (Yamada and Knight, 2002). Figure 2 depicts a deduction step in the baseline decoder. The CFG rule in the first line is used to generate a new item A with span (i, k) using items B and C, which have spans (i, j) and (j, k) respectively. The cost of each item is shown on the right. For experimenting with complete semantic rules, in addition having more rules, the only other modification made to the baseline system is extending the feature vector to include the new feature. We do not modify the decoder in any significant way.

2.2 Semantic Language Model

The semantic language model is designed to encourage the correct order of translation for the semantic roles. While the complete translation rules of Section 2.1 contain the order of the translation for core semantic roles, they do not include the non-core semantic roles, that is, semantic roles which are not essential for the verbal predicates, but do contribute to the meaning of the predicate. In addition, the semantic LM can help in cases where no specific complete semantic rule can apply, which makes the system more flexible.

The semantic language model resembles a regular language model, but instead of words, it defines a probability distribution over sequences of semantic roles. For this method we also use a semantic role labeler on our training data, and use the labeled data to train a tri-gram semantic language model.

The rules are extracted using the baseline rule extraction method. As opposed to the previous method, the rules for this method are not derived by combining GHKM rules, but rather are regular GHKM rules which are annotated with semantic roles. We make a new field in each rule to keep the ordered list of the semantic roles in that rule. We also include the nonterminals of the right-hand-side of the rule in that ordered list, to be able to substitute the semantic roles from the input translation items in the correct order. The decoder uses this new field to save the semantic roles in the translation items, and propagates the semantic LM *states* in the same way that the regular language model states are propagated by the decoder.

We define a new feature for the semantic language model, and score the semantic states in each translation item, again analogously to a regular language model. Figure 3 depicts how the deduction for this method is different from our baseline. In this example, the semantic roles "Agent", "PRED_bring" and "Theme" come from the input items, and the role "Destination" (which tags the terminals "to space") comes from the translation rule.

We stemmed the verbs for training this feature, and also annotated our rules with stemmed verbal predicates. The stemming helps the training since the argument types of a verb are normally independent of its inflected variants.

	avg. BLEU Score		
	dev	test	p-value
Baseline	26.01	25.00	-
Source	26.44	25.17	0.048
Source and target	26.39	**25.63**	$< 10^{-10}$
Propbank LM	26.38	25.08	0.108
VerbNet LM	26.58	**25.23**	0.025

Table 1: Comparisons of the methods with the baseline. The BLEU scores are calculated on the top 3 results from 15 runs MERT for each experiments. The p-values are calculated by comparing each method against the baseline system.

3 Experiments

3.1 Experimental Setup

The data that we used for training the MT system was a Chinese-English corpus derived from newswire text from LDC.[1] The data consists of 250K sentences, which is 6.3M words in the English side. Our language model was trained on the English side of the entire data, which consisted of 1.65M sentences (39.3M words). Our development and test sets are from the newswire portion of NIST evaluations (2004, 2005, 2006). We used 392 sentences for the development set and 428 sentences for the test set. These sentences have lengths smaller than 30, and they each have 4 reference translations. We used our in-house string-to-tree decoder that uses Earley parsing. Other than the features that we presented for our new methods, we used a set of nine standard features. The rules for the baseline system were extracted using the GHKM method. Our baseline GHKM rules also include composed rules, where larger rules are constructed by combining two levels of the regular GHKM rules. We exclude any unary rules (Chung et al., 2011), and only keep rules that have scope up to 3 (Hopkins and Langmead, 2010). For the semantic language model, we used the SRILM package (Stolcke, 2002) and trained a tri-gram language model with the default Good-Turing smoothing.

Our target side semantic role labeler uses a maximum entropy classifier to label parsed sentences. We used Sections 02-22 of the Penn TreeBank to train the labeler, and sections 24 and 23 for development set and training set respectively. The labeler has a precision of 90% and a recall of 88%. We used the Chinese semantic role labeler of Wu and Palmer (2011) for source side SRL, which uses the LIBLINEAR (Fan et al., 2008) as a classifier. Minimum Error Rate Training (MERT) (Och, 2003) was used for tuning the feature weights. For all of our experiments, we ran 15 instances of MERT with random initial weight vectors, and used the weights of the top 3 results on the development set to test the systems on the test set. We chose to use the top 3 runs (rather than the best run) of each system to account for the instability of MERT (Clark et al., 2011). This method is designed to reflect the average performance of the MT system when trained with random restarts of MERT: we wish to discount runs in which the optimizer is stuck in a poor region of the weight space, but also to average across several good runs in order not to be mislead by the high variance of the single best run. For each of our MT systems, we merged the results of the top 3 runs on the test set into one file, and ran a statistical significance test, comparing it to the merged top 3 results from our baseline system. The 3 runs were merged by duplicating each run 3 times, and arranging them in the file so that the significance testing compares each run with all the runs of the baseline. We performed significance testing using paired bootstrap resampling (Koehn, 2004). The difference is considered statistically significant if $p < 0.05$ using 1000 iterations of paired bootstrap resampling.

3.2 Results

Our results are shown in Table 1. The second and the third columns contain the average BLEU score (Papineni et al., 2002) on the top three results on the development and test sets. The fourth column is the p-value for statistical significance testing against the baseline. The first row shows the results for our baseline. The second row contains the results for using the source (Chinese) side complete semantic rules of Section 2.1, and the third row is the results for combining both the source and the target side complete semantic rules. As noted before, in both of these experiments we also use the regular GHKM rules. The result show that the source side complete semantic rules improve the system ($p = 0.048$), and as we expected, combining the source and the tar-

[1]The data was randomly selected from the following sources: LDC2006E86, LDC2006E93, LDC2002E18, LDC2002L27, LDC2003E07, LDC2003E14, LDC2004T08, LDC2005T06, LDC2005T10, LDC2005T34, LDC2006E26, LDC2005E83, LDC2006E34, LDC2006E85, LDC2006E92, LDC2006E24, LDC2006E92, LDC2006E24

Source Sentence	因此 , 保护 儿童 免 受 武装冲突 的 伤害 是 国际 社会 重要 的 职责 .
Reference	therefore , it is the international community 's responsibility to protect the children from harms resulted from armed conflicts .
Baseline	the armed conflicts will harm the importance of the international community the responsibilities . therefore , from child protection
Verbet LM	therefore , the importance of the international community is to protect children from the harm affected by the armed conflicts .

Source Sentence	同 去年 的 会议 相比 , 今年 会议 的 火药 味 消失 了 , 双方 的 立场 在 靠近 .
Reference	compared with last year 's meeting , the smell of gunpowder has disappeared in this year 's meeting and the two sides ' standpoints are getting closer .
Baseline	disappears on gunpowder , near the stance of the two sides compared with last year 's meeting , the meeting of this year .
Verbet LM	the smells of gunpowder has disappeared , the position in the two sides approach . compared with last year 's meeting , this meeting

(a) Comparison of the language model method (using VerbNet) and the baseline system.

Source Sentence	科学家 曾 大胆 预料 , 这 艘 英国 的 太空 船 可能 陷 在 坑 洞 中 .
Reference	scientists have boldly predicted that the british spacecraft might have been stuck in a hole .
Baseline	scientists boldly expected , this vessel uk may have in the space ship in hang tung .
Semantic Rules	scientists have boldly expected this vessel and the possible settlement of the space ship in hang tung .

Source Sentence	美国 政府 应 以 善意 对待 朝鲜 的 这 一 立场 .
Reference	the us government should show goodwills to north korea 's stand .
Baseline	this position of the government of the united states to goodwill toward the dprk .
Semantic Rules	this position that the us government should use goodwill toward the dprk .

(b) Comparison of the experiments with source and target side semantic rules and the baseline system.

Figure 4: Comparison of example translations from our semantic methods and the baseline system.

get side rules improves the system even more significantly ($p < 10^{-10}$). To measure the effect of combining the rules, in a separate experiment we replicated the complete semantic rules experiments of Bazrafshan and Gildea (2013), and ran statistical significance tests comparing the combination of the source and target rules with using only the source or the target semantic rules separately. The results showed that combining the semantic rules outperforms both of the experiments that used rules from only one side (with $p < 0.05$ in both cases).

The results for the language model feature are shown in the last two rows of the table. Using Propbank for language model training did not change the system in any significant way ($p = 0.108$), but using VerbNet significantly improved the results ($p = 0.025$). Figure 4(a) contains an example comparing the baseline system with the VerbNet language model. We can see how the VerbNet language model helps the decoder translate the argument in the correct order. The baseline system has also generated the correct arguments, but the output is in the wrong order. Figure 4(b) compares the experiment with semantic rules of both target and source side and the baseline system. Translation of the word "use" by our semantic rules is a good example showing how the decoder uses these semantic rules to generate a more complete predicate-argument structure.

4 Conclusions

We experimented with two techniques for incorporating semantics in machine translation. The models were designed to help the decoder translate semantic roles in the correct order, as well as generating complete predicate-argument structures. We observed that using a semantic language model can significantly improve the translations, and help the decoder to generate the semantic roles in the correct order. Adding translation rules with complete semantic structures also improved our MT system. We experimented with using source side *complete semantic rules*, as well as using rules for both the source and the target sides. Both of our experiments showed improvements over the baseline, and as expected, the second one had a higher improvement.

Acknowledgments

Partially funded by NSF grant IIS-0910611.

References

Marzieh Bazrafshan and Daniel Gildea. 2013. Semantic roles for string to tree machine translation. In *Association for Computational Linguistics (ACL-13) short paper*.

Tagyoung Chung, Licheng Fang, and Daniel Gildea. 2011. Issues concerning decoding with synchronous context-free grammar. In *Proceedings of the ACL 2011 Conference Short Papers*, Portland, Oregon. Association for Computational Linguistics.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 176–181, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June.

Pascale Fung, Zhaojun Wu, Yongsheng Yang, and Dekai Wu. 2007. Learning Bilingual Semantic Frames: Shallow Semantic Parsing vs. Semantic Role Projection. In *TMI-2007: Proceedings of the 11 th International Conference on Theoretical and Methodological Issues in Machine Translation*, Skövde, Sweden.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of NAACL-04*, pages 273–280, Boston, MA.

Mark Hopkins and Greg Langmead. 2010. SCFG decoding without binarization. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 646–655, Cambridge, MA, October.

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylvania.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395, Barcelona, Spain, July.

Junhui Li, Philip Resnik, and Hal Daumé III. 2013. Modeling syntactic and semantic structures in hierarchical phrase-based translation. In *HLT-NAACL*, pages 540–549.

Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *COLING-10*, Beijing.

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of ACL-03*, pages 160–167, Sapporo, Japan.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Martha. Palmer. 2009. SemLink: Linking PropBank, VerbNet and FrameNet. In *Proceedings of the Generative Lexicon ConferenceGenLex-09*, Pisa, Italy, Sept.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL-02*, pages 311–318, Philadelphia, PA.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Dekai Wu and Pascale Fung. 2009. Semantic roles for smt: A hybrid two-pass model. In *Proceedings of the HLT-NAACL 2009: Short Papers*, Boulder, Colorado.

Shumin Wu and Martha Palmer. 2011. Semantic mapping using automatic word alignment and semantic role labeling. In *Proceedings of the Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation*, SSST-5, pages 21–30, Stroudsburg, PA, USA. Association for Computational Linguistics.

Deyi Xiong, Min Zhang, and Haizhou Li. 2012. Modeling the translation of predicate-argument structure for smt. In *ACL (1)*, pages 902–911.

Kenji Yamada and Kevin Knight. 2002. A decoder for syntax-based statistical MT. In *Proceedings of ACL-02*, pages 303–310, Philadelphia, PA.

Detecting Non-compositional MWE Components using Wiktionary

Bahar Salehi,♣♡ Paul Cook♡ and Timothy Baldwin♣♡
♣ NICTA Victoria Research Laboratory
♡ Department of Computing and Information Systems
The University of Melbourne
Victoria 3010, Australia
bsalehi@student.unimelb.edu.au, paulcook@unimelb.edu.au, tb@ldwin.net

Abstract

We propose a simple unsupervised approach to detecting non-compositional components in multiword expressions based on Wiktionary. The approach makes use of the definitions, synonyms and translations in Wiktionary, and is applicable to any type of MWE in any language, assuming the MWE is contained in Wiktionary. Our experiments show that the proposed approach achieves higher F-score than state-of-the-art methods.

1 Introduction

A multiword expression (MWE) is a combination of words with lexical, syntactic or semantic idiosyncrasy (Sag et al., 2002; Baldwin and Kim, 2009). An MWE is considered (semantically) "non-compositional" when its meaning is not predictable from the meaning of its components. Conversely, compositional MWEs are those whose meaning is predictable from the meaning of the components. Based on this definition, a component is compositional within an MWE, if its meaning is reflected in the meaning of the MWE, and it is non-compositional otherwise.

Understanding which components are non-compositional within an MWE is important in NLP applications in which semantic information is required. For example, when searching for *spelling bee*, we may also be interested in documents about *spelling*, but not those which contain only *bee*. For *research project*, on the other hand, we are likely to be interested in documents which contain either *research* or *project* in isolation, and for *swan song*, we are only going to be interested in documents which contain the phrase *swan song*, and not just *swan* or *song*.

In this paper, we propose an unsupervised approach based on Wikitionary for predicting which components of a given MWE have a compositional usage. Experiments over two widely-used datasets show that our approach outperforms state-of-the-art methods.

2 Related Work

Previous studies which have considered MWE compositionality have focused on either the identification of non-compositional MWE token instances (Kim and Baldwin, 2007; Fazly et al., 2009; Forthergill and Baldwin, 2011; Muzny and Zettlemoyer, 2013), or the prediction of the compositionality of MWE types (Reddy et al., 2011; Salehi and Cook, 2013; Salehi et al., 2014). The identification of non-compositional MWE tokens is an important task when a word combination such as *kick the bucket* or *saw logs* is ambiguous between a compositional (generally non-MWE) and non-compositional MWE usage. Approaches have ranged from the unsupervised learning of type-level preferences (Fazly et al., 2009) to supervised methods specific to particular MWE constructions (Kim and Baldwin, 2007) or applicable across multiple constructions using features similar to those used in all-words word sense disambiguation (Forthergill and Baldwin, 2011; Muzny and Zettlemoyer, 2013). The prediction of the compositionality of MWE types has traditionally been couched as a binary classification task (compositional or non-compositional: Baldwin et al. (2003), Bannard (2006)), but more recent work has moved towards a regression setup, where the degree of the compositionality is predicted on a continuous scale (Reddy et al., 2011; Salehi and Cook, 2013; Salehi et al., 2014). In either case, the modelling has been done either over the whole MWE (Reddy et al., 2011; Salehi and Cook, 2013), or relative to each component within the MWE (Baldwin et al., 2003; Bannard, 2006). In this paper, we focus on the binary classification of MWE types relative to each component of the

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1792–1797,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

MWE.

The work that is perhaps most closely related to this paper is that of Salehi and Cook (2013) and Salehi et al. (2014), who use translation data to predict the compositionality of a given MWE relative to each of its components, and then combine those scores to derive an overall compositionality score. In both cases, translations of the MWE and its components are sourced from PanLex (Baldwin et al., 2010; Kamholz et al., 2014), and if there is greater similarity between the translated components and MWE in a range of languages, the MWE is predicted to be more compositional. The basis of the similarity calculation is unsupervised, using either string similarity (Salehi and Cook, 2013) or distributional similarity (Salehi et al., 2014). However, the overall method is supervised, as training data is used to select the languages to aggregate scores across for a given MWE construction. To benchmark our method, we use two of the same datasets as these two papers, and repurpose the best-performing methods of Salehi and Cook (2013) and Salehi et al. (2014) for classification of the compositionality of each MWE component.

3 Methodology

Our basic method relies on analysis of lexical overlap between the component words and the definitions of the MWE in Wiktionary, in the manner of Lesk (1986). That is, if a given component can be found in the definition, then it is inferred that the MWE carries the meaning of that component. For example, the Wiktionary definition of *swimming pool* is "An artificially constructed pool of water used for swimming", suggesting that the MWE is compositional relative to both *swimming* and *pool*. If the MWE is not found in Wiktionary, we use Wikipedia as a backoff, and use the first paragraph of the (top-ranked) Wikipedia article as a proxy for the definition.

As detailed below, we further extend the basic method to incorporate three types of information found in Wiktionary: (1) definitions of each word in the definitions, (2) synonyms of the words in the definitions, and (3) translations of the MWEs and components.

3.1 Definition-based Similarity

The basic method uses Boolean lexical overlap between the target component of the MWE and a definition. A given MWE will often have multiple definitions, however, begging the question of how to combine across them, for which we propose the following three methods.

First Definition (FIRSTDEF): Use only the first-listed Wiktionary definition for the MWE, based on the assumption that this is the predominant sense.

All Definitions (ALLDEFS): In the case that there are multiple definitions for the MWE, calculate the lexical overlap for each independently and take a majority vote; in the case of a tie, label the component as non-compositional.

Idiom Tag (ITAG): In Wiktionary, there is facility for users to tag definitions as idiomatic.[1] If, for a given MWE, there are definitions tagged as idiomatic, use only those definitions; if there are no such definitions, use the full set of definitions.

3.2 Synonym-based Definition Expansion

In some cases, a component is not explicitly mentioned in a definition, but a synonym does occur, indicating that the definition is compositional in that component. In order to capture synonym-based matches, we optionally look for synonyms of the component word in the definition,[2] and expand our notion of lexical overlap to include these synonyms.

For example, for the MWE *china clay*, the definition is *kaolin*, which includes neither of the components. However, we find the component word *clay* in the definition for *kaolin*, as shown below.

> A fine clay, rich in kaolinite, used in ceramics, paper-making, etc.

This method is compatible with the three definition-based similarity methods described above, and indicated by the +SYN suffix (e.g. FIRSTDEF+SYN is FIRSTDEF with synonym-based expansion).

3.3 Translations

A third information source in Wiktionary that can be used to predict compositionality is sense-level translation data. Due to the user-generated nature of Wiktionary, the set of languages for which

[1] Although the recall of these tags is low (Muzny and Zettlemoyer, 2013).

[2] After removing function words, based on a stopword list.

	ENC	EVPC
WordNet	91.1%	87.5%
Wiktionary	96.7%	96.2%
Wiktionary+Wikipedia	100.0%	96.2%

Table 1: Lexical coverage of WordNet, Wiktionary and Wiktionary+Wikipedia over our two datasets.

translations are provided varies greatly across lexical entries. Our approach is to take whatever translations happen to exist in Wiktionary for a given MWE, and where there are translations in that language for the component of interest, use the LCS-based method of Salehi and Cook (2013) to measure the string similarity between the translation of the MWE and the translation of the components. Unlike Salehi and Cook (2013), however, we do not use development data to select the optimal set of languages in a supervised manner, and instead simply take the average of the string similarity scores across the available languages. In the case of more than one translation in a given language, we use the maximum string similarity for each pairing of MWE and component translation.

Unlike the definition and synonym-based approach, the translation-based approach will produce real rather than binary values. To combine the two approaches, we discretise the scores given by the translation approach. In the case of disagreement between the two approaches, we label the given MWE as non-compositional. This results in higher recall and lower precision for the task of detecting compositionality.

3.4 An Analysis of Wiktionary Coverage

A dictionary-based method is only as good as the dictionary it is applied to. In the case of MWE compositionality analysis, our primary concern is lexical coverage in Wiktionary, i.e., what proportion of a representative set of MWEs is contained in Wiktionary. We measure lexical coverage relative to the two datasets used in this research (described in detail in Section 4), namely 90 English noun compounds (ENCs) and 160 English verb particle constructions (EVPCs). In each case, we calculated the proportion of the dataset that is found in Wiktionary, Wiktionary+Wikipedia (where we back off to a Wikipedia document in the case that a MWE is not found in Wiktionary) and WordNet (Fellbaum, 1998). The results are found in Table 1, and indicate perfect coverage in Wiktionary+Wikipedia for the ENCs, and very high coverage for the EVPCs. In both cases, the coverage of WordNet is substantially lower, although still respectable, at around 90%.

4 Datasets

As mentioned above, we evaluate our method over the same two datasets as Salehi and Cook (2013) (which were later used, in addition to a third dataset of German noun compounds, in Salehi et al. (2014)): (1) 90 binary English noun compounds (ENCs, e.g. *spelling bee* or *swimming pool*); and (2) 160 English verb particle constructions (EVPCs, e.g. *stand up* and *give away*). Our results are not directly comparable with those of Salehi and Cook (2013) and Salehi et al. (2014), however, who evaluated in terms of a regression task, modelling the overall compositionality of the MWE. In our case, the task setup is a binary classification task relative to each of the two components of the MWE.

The ENC dataset was originally constructed by Reddy et al. (2011), and annotated on a continuous $[0, 5]$ scale for both overall compositionality and the component-wise compositionality of each of the modifier and head noun. The sampling was random in an attempt to make the dataset balanced, with 48% of compositional English noun compounds, of which 51% are compositional in the first component and 60% are compositional in the second component. We generate discrete labels by discretising the component-wise compositionality scores based on the partitions $[0, 2.5]$ and $(2.5, 5]$. On average, each NC in this dataset has 1.4 senses (definitions) in Wiktionary.

The EVPC dataset was constructed by Bannard (2006), and manually annotated for compositionality on a binary scale for each of the head verb and particle. For the 160 EVPCs, 76% are verb-compositional and 48% are particle-compositional. On average, each EVPC in this dataset has 3.0 senses (definitions) in Wiktionary.

5 Experiments

The baseline for each dataset takes the form of looking for a user-annotated idiom tag in the Wiktionary lexical entry for the MWE: if there is an idiomatic tag, both components are considered to be non-compositional; otherwise, both components are considered to be compositional. We expect this method to suffer from low precision for two

Method	First Component			Second Component		
	Precision	Recall	F-score	Precision	Recall	F-score
Baseline	66.7	68.2	67.4	66.7	83.3	74.1
LCS	60.0	77.7	67.7	81.6	68.1	64.6
DS	62.1	88.6	73.0	80.5	86.4	71.2
DS+DSL2	62.5	92.3	74.5	78.4	89.4	70.6
LCS+DS+DSL2	66.3	87.5	75.4	82.1	80.6	70.1
FIRSTDEF	59.4	93.2	72.6	54.2	88.9	67.4
ALLDEFS	59.5	**100.0**	74.6	52.9	**100.0**	69.2
ITAG	60.3	**100.0**	75.2	54.5	**100.0**	70.6
FIRSTDEF+SYN	64.9	84.1	73.3	63.8	83.3	72.3
ALLDEFS+SYN	64.5	90.9	75.5	60.4	88.9	71.9
ITAG+SYN	64.5	90.9	75.5	61.8	94.4	**74.7**
FIRSTDEF+SYN $_{\text{COMB(LCS+DS+DSL2)}}$	**82.9**	85.3	84.1	81.9	80.0	69.8
ALLDEFS+SYN $_{\text{COMB(LCS+DS+DSL2)}}$	81.2	88.1	**84.5**	87.3	80.6	73.3
ITAG+SYN $_{\text{COMB(LCS+DS+DSL2)}}$	81.0	88.1	84.1	**88.0**	81.1	73.9

Table 2: Compositionality prediction results over the ENC dataset, relative to the first component (the modifier noun) and the second component (the head noun)

reasons: first, the guidelines given to the annotators of our datasets might be different from what Wiktionary contributors assume to be an idiom. Second, the baseline method assumes that for any non-compositional MWE, all components must be equally non-compositional, despite the wealth of MWEs where one or more components are compositional (e.g. from the Wiktionary guidelines for idiom inclusion,[3] *computer chess*, *basketball player*, *telephone box*).

We also compare our method with: (1) "LCS", the string similarity-based method of Salehi and Cook (2013), in which 54 languages are used; (2) "DS", the monolingual distributional similarity method of Salehi et al. (2014); (3) "DS+DSL2", the multilingual distributional similarity method of Salehi et al. (2014), including supervised language selection for a given dataset, based on cross-validation; and (4) "LCS+DS+DSL2", whereby the first three methods are combined using a supervised support vector regression model. In each case, the continuous output of the model is equal-width discretised to generate a binary classification. We additionally present results for the combination of each of the six methods proposed in this paper with LCS, DS and DSL2, using a linear-kernel support vector machine (represented with the suffix "$_{\text{COMB(LCS+DS+DSL2)}}$" for a given method). The results are based on cross-validation, and for direct comparability, the partitions are exactly the same as Salehi et al. (2014).

Tables 2 and 3 provide the results when our proposed method for detecting non-compositionality is applied to the ENC and EVPC datasets, respectively. The inclusion of translation data was found to improve all of precision, recall and F-score across the board for all of the proposed methods. For reasons of space, results without translation data are therefore omitted from the paper.

Overall, the simple unsupervised methods proposed in this paper are comparable with the unsupervised and supervised state-of-the-art methods of Salehi and Cook (2013) and Salehi et al. (2014), with ITAG achieving the highest F-score for the ENC dataset and for the verb components of the EVPC dataset. The inclusion of synonyms boosts results in most cases.

When we combine each of our proposed methods with the string and distributional similarity methods of Salehi and Cook (2013) and Salehi et al. (2014), we see substantial improvements over the comparable combined method of "LCS+DS+DSL2" in most cases, demonstrating both the robustness of the proposed methods and their complementarity with the earlier methods. It is important to reinforce that the proposed methods make no language-specific assumptions and are therefore applicable to any type of MWE and any language, with the only requirement being that the MWE of interest be listed in the Wiktionary for

[3] http://en.wiktionary.org/wiki/Wiktionary:Idioms_that_survived_RFD

Method	First Component			Second Component		
	Precision	Recall	F-score	Precision	Recall	F-score
Baseline	24.6	36.8	29.5	59.6	40.5	48.2
LCS	36.5	49.2	39.3	61.5	63.7	60.3
DS	32.8	34.1	33.5	80.9	19.6	29.7
DS+DSL2	31.8	72.4	44.2	74.8	27.5	36.6
LCS+DS+DSL2	36.1	62.6	45.8	77.9	42.8	49.2
FIRSTDEF	24.8	84.2	38.3	54.5	94.0	69.0
ALLDEFS	25.0	**97.4**	39.8	53.6	**97.6**	**69.2**
ITAG	26.2	89.5	40.5	54.6	91.7	68.4
FIRSTDEF+SYN	32.9	65.8	43.9	60.4	65.5	62.9
ALLDEFS+SYN	28.4	81.6	42.1	62.5	77.4	69.1
ITAG+SYN	30.5	65.8	41.7	57.8	61.9	59.8
FIRSTDEF+SYN $_{\text{COMB(LCS+DS+DSL2)}}$	34.0	65.3	44.7	**83.6**	67.3	65.4
ALLDEFS+SYN $_{\text{COMB(LCS+DS+DSL2)}}$	**37.4**	70.9	**48.9**	80.4	65.9	63.0
ITAG+SYN $_{\text{COMB(LCS+DS+DSL2)}}$	35.6	70.9	47.4	83.5	64.9	64.2

Table 3: Compositionality prediction results over the EVPC dataset, relative to the first component (the head verb) and the second component (the particle)

that language.

6 Error Analysis

We analysed all items in each dataset where the system score differed from that of the human annotators. For both datasets, the majority of incorrectly-labelled items were compositional but predicted to be non-compositional by our system, as can be seen in the relatively low precision scores in Tables 2 and 3. In many of these cases, the prediction based on definitions and synonyms was compositional but the prediction based on translations was non-compositional. In such cases, we arbitrarily break the tie by labelling the instance as non-compositional, and in doing so favour recall over precision.

Some of the incorrectly-labelled ENCs have a gold-standard annotation of around 2.5, or in other words are semi-compositional. For example, the compositionality score for *game* in *game plan* is 2.82/5, but our system labels it as non-compositional; a similar thing happens with *figure* and the EVPC *figure out*. Such cases demonstrate the limitation of approaches to MWE compositionality that treat the problem as a binary classification task.

On average, the EVPCs have three senses, which is roughly twice the number for ENCs. This makes the prediction of compositionality harder, as there is more information to combine across (an effect that is compounded with the addition of synonyms and translations). In future work, we hope to address this problem by first finding the sense which matches best with the sentences given to the annotators.

7 Conclusion

We have proposed an unsupervised approach for predicting the compositionality of an MWE relative to each of its components, based on lexical overlap using Wiktionary, optionally incorporating synonym and translation data. Our experiments showed that the various instantiations of our approach are superior to previous state-of-the-art supervised methods. All code to replicate the results in this paper has been made publicly available at `https://github.com/bsalehi/wiktionary_MWE_compositionality`.

Acknowledgements

We thank the anonymous reviewers for their insightful comments and valuable suggestions. NICTA is funded by the Australian government as represented by Department of Broadband, Communication and Digital Economy, and the Australian Research Council through the ICT Centre of Excellence programme.

References

Timothy Baldwin and Su Nam Kim. 2009. Multiword expressions. In Nitin Indurkhya and Fred J. Dam-

erau, editors, *Handbook of Natural Language Processing*. CRC Press, Boca Raton, USA, 2nd edition.

Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression decomposability. In *Proceedings of the ACL-2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, pages 89–96, Sapporo, Japan.

Timothy Baldwin, Jonathan Pool, and Susan M. Colowick. 2010. PanLex and LEXTRACT: Translating all words of all languages of the world. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 37–40, Beijing, China.

Colin James Bannard. 2006. *Acquiring Phrasal Lexicons from Corpora*. Ph.D. thesis, University of Edinburgh.

Afsaneh Fazly, Paul Cook, and Suzanne Stevenson. 2009. Unsupervised type and token identification of idiomatic expressions. *Computational Linguistics*, 35(1):61–103.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA.

Richard Forthergill and Timothy Baldwin. 2011. Fleshing it out: A supervised approach to MWE-token and MWE-type classification. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011)*, pages 911–919, Chiang Mai, Thailand.

David Kamholz, Jonathan Pool, and Susan Colowick. 2014. PanLex: Building a resource for panlingual lexical translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3145–3150, Reykjavik, Iceland.

Su Nam Kim and Timothy Baldwin. 2007. Detecting compositionality of English verb-particle constructions using semantic similarity. In *Proceedings of the 7th Meeting of the Pacific Association for Computational Linguistics (PACLING 2007)*, pages 40–48, Melbourne, Australia.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, pages 24–26, Ontario, Canada.

Grace Muzny and Luke Zettlemoyer. 2013. Automatic idiom identification in Wiktionary. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1417–1421, Seattle, USA.

Siva Reddy, Diana McCarthy, and Suresh Manandhar. 2011. An empirical study on compositionality in compound nouns. In *Proceedings of IJCNLP*, pages 210–218, Chiang Mai, Thailand.

Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the 3rd International Conference on Intelligent Text Processing Computational Linguistics (CICLing-2002)*, pages 189–206, Mexico City, Mexico.

Bahar Salehi and Paul Cook. 2013. Predicting the compositionality of multiword expressions using translations in multiple languages. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, volume 1, pages 266–275, Atlanta, USA.

Bahar Salehi, Paul Cook, and Timothy Baldwin. 2014. Using distributional similarity of multi-way translations to predict multiword expression compositionality. In *Proceedings of the 14th Conference of the EACL (EACL 2014)*, pages 472–481, Gothenburg, Sweden.

Joint Emotion Analysis via Multi-task Gaussian Processes

Daniel Beck[†] **Trevor Cohn**[‡] **Lucia Specia**[†]
[†]Department of Computer Science, University of Sheffield, United Kingdom
{debeck1,l.specia}@sheffield.ac.uk
[‡]Computing and Information Systems, University of Melbourne, Australia
t.cohn@unimelb.edu.au

Abstract

We propose a model for jointly predicting multiple emotions in natural language sentences. Our model is based on a low-rank coregionalisation approach, which combines a vector-valued Gaussian Process with a rich parameterisation scheme. We show that our approach is able to learn correlations and anti-correlations between emotions on a news headlines dataset. The proposed model outperforms both single-task baselines and other multi-task approaches.

1 Introduction

Multi-task learning (Caruana, 1997) has been widely used in Natural Language Processing. Most of these learning methods are aimed for Domain Adaptation (Daumé III, 2007; Finkel and Manning, 2009), where we hypothesize that we can learn from multiple domains by assuming similarities between them. A more recent use of multi-task learning is to model annotator bias and noise for datasets labelled by multiple annotators (Cohn and Specia, 2013).

The settings mentioned above have one aspect in common: they assume some degree of positive correlation between tasks. In Domain Adaptation, we assume that some "general", domain-independent knowledge exists in the data. For annotator noise modelling, we assume that a "ground truth" exists and that annotations are some noisy deviations from this truth. However, for some settings these assumptions do not necessarily hold and often tasks can be *anti-correlated*. For these cases, we need to employ multi-task methods that are able to learn these relations from data and correctly employ them when making predictions, avoiding negative knowledge transfer.

An example of a problem that shows this behaviour is Emotion Analysis, where the goal is to automatically detect emotions in a text (Strapparava and Mihalcea, 2008; Mihalcea and Strapparava, 2012). This problem is closely related to Opinion Mining (Pang and Lee, 2008), with similar applications, but it is usually done at a more fine-grained level and involves the prediction of a set of labels (one for each emotion) instead of a single label. While we expect some emotions to have some degree of correlation, this is usually not the case for all possible emotions. For instance, we expect *sadness* and *joy* to be anti-correlated.

We propose a multi-task setting for Emotion Analysis based on a vector-valued Gaussian Process (GP) approach known as *coregionalisation* (Álvarez et al., 2012). The idea is to combine a GP with a low-rank matrix which encodes task correlations. Our motivation to employ this model is three-fold:

- Datasets for this task are scarce and small so we hypothesize that a multi-task approach will results in better models by allowing a task to borrow statistical strength from other tasks;
- The annotation scheme is subjective and very fine-grained, and is therefore heavily prone to bias and noise, both which can be modelled easily using GPs;
- Finally, we also have the goal to learn a model that shows sound and interpretable correlations between emotions.

2 Multi-task Gaussian Process Regression

Gaussian Processes (GPs) (Rasmussen and Williams, 2006) are a Bayesian kernelised framework considered the state-of-the-art for regression. They have been recently used successfully for translation quality prediction (Cohn and Specia, 2013; Beck et al., 2013; Shah et al., 2013)

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1798–1803,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

and modelling text periodicities (Preotiuc-Pietro and Cohn, 2013). In the following we give a brief description on how GPs are applied in a regression setting.

Given an input $\mathbf{x}$, the GP regression assumes that its output y is a noise corrupted version of a latent function evaluation, $y = f(\mathbf{x}) + \eta$, where $\eta \sim \mathcal{N}(0, \sigma_n^2)$ is the added white noise and the function f is drawn from a GP prior:

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{1}$$

where $\mu(\mathbf{x})$ is the *mean* function, which is usually the 0 constant, and $k(\mathbf{x}, \mathbf{x}')$ is the kernel or *covariance* function, which describes the covariance between values of f at locations $\mathbf{x}$ and $\mathbf{x}'$.

To predict the value for an unseen input $\mathbf{x}_*$, we compute the Bayesian posterior, which can be calculated analytically, resulting in a Gaussian distribution over the output y_*:[1]

$$\begin{aligned} y_* \sim \mathcal{N}(&\mathbf{k}_*(\mathbf{K} + \sigma_n \mathbf{I})^{-1}\mathbf{y}^T, \\ &k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T(\mathbf{K} + \sigma_n \mathbf{I})^{-1}\mathbf{k}_*), \end{aligned} \tag{2}$$

where $\mathbf{K}$ is the Gram matrix corresponding to the covariance kernel evaluated at every pair of training inputs and $\mathbf{k}_* = [\langle \mathbf{x}_1, \mathbf{x}_* \rangle, \langle \mathbf{x}_2, \mathbf{x}_* \rangle, \ldots, \langle \mathbf{x}_n, \mathbf{x}_* \rangle]$ is the vector of kernel evaluations between the test input and each training input.

2.1 The Intrinsic Coregionalisation Model

By extending the GP regression framework to vector-valued outputs we obtain the so-called coregionalisation models. Specifically, we employ a separable vector-valued kernel known as *Intrinsic Coregionalisation Model* (ICM) (Álvarez et al., 2012). Considering a set of D tasks, we define the corresponding vector-valued kernel as:

$$k((\mathbf{x}, d), (\mathbf{x}', d')) = k_{\text{data}}(\mathbf{x}, \mathbf{x}') \times \mathbf{B}_{d,d'}, \tag{3}$$

where k_{data} is a kernel on the input points (here a Radial Basis Function, RBF), d and d' are task or metadata information for each input and $\mathbf{B} \in \mathbb{R}^{D \times D}$ is the coregionalisation matrix, which encodes task covariances and is symmetric and positive semi-definite.

A key advantage of GP-based modelling is its ability to learn hyperparameters directly from data by maximising the marginal likelihood:

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \int_f p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, f)p(f). \tag{4}$$

This process is usually performed to learn the noise variance and kernel hyperparameters, including the coregionalisation matrix. In order to do this, we need to consider how $\mathbf{B}$ is parameterised.

Cohn and Specia (2013) treat the diagonal values of $\mathbf{B}$ as hyperparameters, and as a consequence are able to leverage the inter-task transfer between each independent task and the global "pooled" task. They however fix non-diagonal values to 1, which in practice is equivalent to assuming equal correlation across tasks. This can be limiting, in that this formulation cannot model anti-correlations between tasks.

In this work we lift this restriction by adopting a different parameterisation of $\mathbf{B}$ that allows the learning of all task correlations. A straightforward way to do that would be to consider every correlation as an hyperparameter, but this can result in a matrix which is not positive semi-definite (and therefore, not a valid covariance matrix). To ensure this property, we follow the method proposed by Bonilla et al. (2008), which decomposes $\mathbf{B}$ using Probabilistic Principal Component Analysis:

$$\mathbf{B} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T + \text{diag}(\boldsymbol{\alpha}), \tag{5}$$

where $\mathbf{U}$ is an $D \times R$ matrix containing the R principal eigenvectors and Λ is a $R \times R$ diagonal matrix containing the corresponding eigenvalues. The choice of R defines the *rank* of $\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$, which can be understood as the capacity of the manifold with which we model the D tasks. The vector $\boldsymbol{\alpha}$ allows for each task to behave more or less independently with respect to the global task. The final rank of $\mathbf{B}$ depends on both terms in Equation 5.

For numerical stability, we use the incomplete-Cholesky decomposition over the matrix $\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$, resulting in the following parameterisation for $\mathbf{B}$:

$$\mathbf{B} = \tilde{\mathbf{L}}\tilde{\mathbf{L}}^T + \text{diag}(\boldsymbol{\alpha}), \tag{6}$$

where $\tilde{\mathbf{L}}$ is a $D \times R$ matrix. In this setting, we treat all elements of $\tilde{\mathbf{L}}$ as hyperparameters. Setting a larger rank allows more flexibility in modelling task correlations. However, a higher number of hyperparameters may lead to overfitting problems or otherwise cause issues in optimisation due

[1] We refer the reader to Rasmussen and Williams (2006, Chap. 2) for an in-depth explanation of GP regression.

to additional non-convexities in the log likelihood objective. In our experiments we evaluate this behaviour empirically by testing a range of ranks for each setting.

The low-rank model can subsume the ones proposed by Cohn and Specia (2013) by fixing and tying some of the hyperparameters:

Independent: fixing $\tilde{\mathbf{L}} = \mathbf{0}$ and $\boldsymbol{\alpha} = \mathbf{1}$;

Pooled: fixing $\tilde{\mathbf{L}} = \mathbf{1}$ and $\boldsymbol{\alpha} = \mathbf{0}$;

Combined: fixing $\tilde{\mathbf{L}} = \mathbf{1}$ and tying all components of $\boldsymbol{\alpha}$;

Combined+: fixing $\tilde{\mathbf{L}} = \mathbf{1}$.

These formulations allow us to easily replicate their modelling approach, which we evaluate as competitive baselines in our experiments.

3 Experimental Setup

To address the feasibility of our approach, we propose a set of experiments with three goals in mind:

- To find our whether the ICM is able to learn sensible emotion correlations;
- To check if these correlations are able to improve predictions for unseen texts;
- To investigate the behaviour of the ICM model as we increase the training set size.

Dataset We use the dataset provided by the "Affective Text" shared task in SemEval-2007 (Strapparava and Mihalcea, 2007), which is composed of 1000 news headlines annotated in terms of six emotions: *Anger, Disgust, Fear, Joy, Sadness* and *Surprise*. For each emotion, a score between 0 and 100 is given, 0 meaning total lack of emotion and 100 maximum emotional load. We use 100 sentences for training and the remaining 900 for testing.

Model For all experiments, we use a Radial Basis Function (RBF) data kernel over a bag-of-words feature representation. Words were downcased and lemmatized using the WordNet lemmatizer in the NLTK[2] toolkit (Bird et al., 2009). We then use the GPy toolkit[3] to combine this kernel with a coregionalisation model over the six emotions, comparing a number of low-rank approximations.

Baselines and Evaluation We compare prediction results with a set of single-task baselines: a Support Vector Machine (SVM) using an RBF kernel with hyperparameters optimised via cross-validation and a single-task GP, optimised via likelihood maximisation. The SVM models were trained using the Scikit-learn toolkit[4] (Pedregosa et al., 2011). We also compare our results against the ones obtained by employing the "Combined" and "Combined+" models proposed by Cohn and Specia (2013). Following previous work in this area, we use Pearson's correlation coefficient as evaluation metric.

4 Results and Discussion

4.1 Learned Task Correlations

Figure 1 shows the learned coregionalisation matrix setting the initial rank as 1, reordering the emotions to emphasize the learned structure. We can see that the matrix follows a block structure, clustering some of the emotions. This picture shows two interesting behaviours:

- *Sadness* and *fear* are highly correlated. *Anger* and *disgust* also correlate with them, although to a lesser extent, and could be considered as belonging to the same cluster. We can also see correlation between *surprise* and *joy*. These are intuitively sound clusters based on the polarity of these emotions.
- In addition to correlations, the model learns anti-correlations, especially between *joy/surprise* and the other emotions. We also note that *joy* has the highest diagonal value, meaning that it gives preference to independent modelling (instead of pooling over the remaining tasks).

Inspecting the eigenvalues of the learned matrix allows us to empirically determine its resulting rank. In this case we find that the model has learned a matrix of rank 3, which indicates that our initial assumption of a rank 1 coregionalisation matrix may be too small in terms of modelling capacity[5]. This suggests that a higher rank is justified, although care must be taken due to the local optima and overfitting issues cited in §2.1.

[2] http://www.nltk.org

[3] http://github.com/SheffieldML/GPy

[4] http://scikit-learn.org

[5] The eigenvalues were $592, 62, 86, 4, 3 \times 10^{-3}$ and 9×10^{-5}.

	Anger	Disgust	Fear	Joy	Sadness	Surprise	All
SVM	**0.3084**	**0.2135**	0.3525	0.0905	**0.3330**	0.1148	0.2603
Single GP	0.1683	0.0035	0.3462	0.2035	0.3011	0.1599	0.3659
ICM GP (Combined)	0.2301	0.1230	0.2913	0.2202	0.2303	0.1744	0.3295
ICM GP (Combined+)	0.1539	0.1240	0.3438	0.2466	0.2850	0.2027	0.3723
ICM GP (Rank 1)	0.2133	0.1075	0.3623	**0.2810**	0.3137	0.2415	**0.3988**
ICM GP (Rank 5)	0.2542	0.1799	**0.3727**	0.2711	0.3157	**0.2446**	0.3957

Table 1: Prediction results in terms of Pearson's correlation coefficient (higher is better). Boldface values show the best performing model for each emotion. The scores for the "All" column were calculated over the predictions for all emotions concatenated (instead of just averaging over the scores for each emotion).

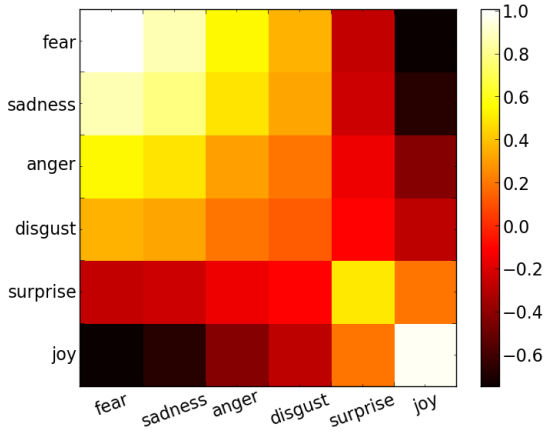


Figure 1: Heatmap showing a learned coregionalisation matrix over the emotions.

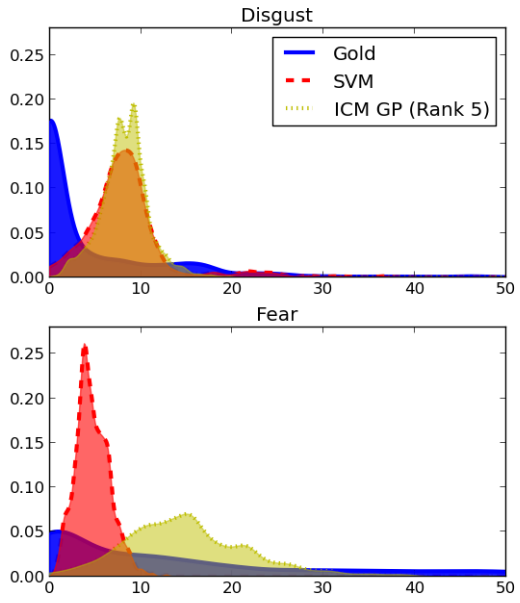


Figure 2: Test score distributions for *disgust* and *fear*. For clarity, only scores between 0 and 50 are shown. SVM performs better on *disgust*, while GP performs better on *fear*.

4.2 Prediction Results

Table 1 shows the Pearson's scores obtained in our experiments. The low-rank models outperformed the baselines for the full task (predicting all emotions) and for *fear*, *joy* and *surprise* subtasks. The rank 5 models were also able to outperform all GP baselines for the remaining emotions, but could not beat the SVM baseline. As expected, the "Combined" and "Combined+" performed worse than the low-rank models, probably due to their inability to model anti-correlations.

4.3 Error analysis

To check why SVM performs better than GPs for some emotions, we analysed their gold-standard score distributions. Figure 2 shows the smoothed distributions for *disgust* and *fear*, comparing the gold-standard scores to predictions from the SVM and GP models. The distributions for the training set follow similar shapes.

We can see that GP obtains better matching score distributions in the case when the gold-standard scores are more spread over the full support of response values, i.e., $[0, 100]$. Since our GP model employs a Gaussian likelihood, it is effectively minimising a squared-error loss. The SVM model, on the other hand, uses hinge loss, which is linear beyond the margin envelope constraints. This affects the treatment of outlier points, which attract quadratic cf. linear penalties for the GP and SVM respectively. Therefore, when training scores are more uniformly distributed (which is the case for *fear*), the GP model has to take the high scores into account, resulting in broader coverage of the full support. For *disgust*, the scores are much more peaked near zero, favouring the

more narrow coverage of the SVM.

More importantly, Figure 2 also shows that both SVM and GP predictions tend to exhibit a Gaussian shape, while the true scores show an exponential behaviour. This suggests that both models are making wrong prior assumptions about the underlying score distribution. For SVMs, this is a non-trivial issue to address, although it is much easier for GPs, where we can use a different likelihood distribution, e.g., a Beta distribution to reflect that the outputs are only valid over a bounded range. Note that non-Gaussian likelihoods mean that exact inference is no longer tractable, due to the lack of conjugacy between the prior and likelihood. However a number of approximate inference methods are appropriate which are already widely used in the GP literature for use with non-Gaussian likelihoods, including expectation propagation (Jylänki et al., 2011), the Laplace approximation (Williams and Barber, 1998) and Markov Chain Monte Carlo sampling (Adams et al., 2009).

4.4 Training Set Influence

We expect multi-task models to perform better for smaller datasets, when compared to single-task models. This stems from the fact that with small datasets often there is more uncertainty associated with each task, a problem which can be alleviated using statistics from the other tasks. To measure this behaviour, we performed an additional experiment varying the size of the training sets, while using 100 sentences for testing.

Figure 3 shows the scores obtained. As expected, for smaller datasets the single-task models are outperformed by ICM, but their performance become equivalent as the training set size increases. SVM performance tends to be slightly worse for most sizes. To study why we obtained an outlier for the single-task model with 200 sentences, we inspected the prediction values. We found that, in this case, predictions for *joy*, *surprise* and *disgust* were all around the same value.[6] For larger datasets, this effect disappears and the single-task models yield good predictions.

5 Conclusions and Future Work

This paper proposed an multi-task approach for Emotion Analysis that is able to learn correlations and anti-correlations between emotions. Our formulation is based on a combination of a Gaussian Process and a low-rank coregionalisation model, using a richer parameterisation that allows the learning of fine-grained task similarities. The proposed model outperformed strong baselines when applied to a news headline dataset.

As it was discussed in Section 4.3, we plan to further explore the possibility of using non-Gaussian likelihoods with the GP models. Another research avenue we intend to explore is to employ multiple layers of metadata, similar to the model proposed by Cohn and Specia (2013). An example is to incorporate the dataset provided by Snow et al. (2008), which provides multiple non-expert emotion annotations for each sentence, obtained via crowdsourcing. Finally, another possible extension comes from more advanced vector-valued GP models, such as the linear model of coregionalisation (Álvarez et al., 2012) or hierarchical kernels (Hensman et al., 2013). These models can be specially useful when we want to employ multiple kernels to explain the relation between the input data and the labels.

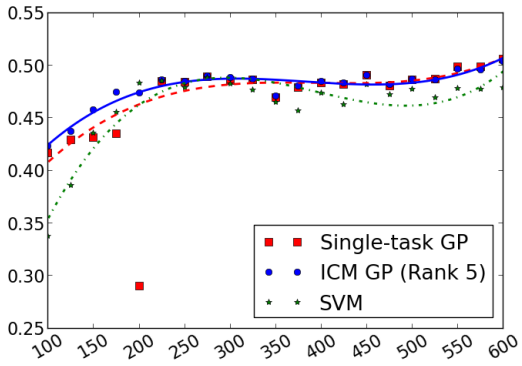


Figure 3: Pearson's correlation score according to training set size (in number of sentences).

Acknowledgements

Daniel Beck was supported by funding from CNPq/Brazil (No. 237999/2012-9). Dr. Cohn is the recipient of an Australian Research Council Future Fellowship (project number FT130101105).

References

Ryan Prescott Adams, Iain Murray, and David J. C. MacKay. 2009. Tractable Nonparametric Bayesian

[6] Looking at the predictions for smaller datasets, we found the same behaviour, but because the values found were near the mean they did not hurt the Pearson's score as much.

Inference in Poisson Processes with Gaussian Process Intensities. In *Proceedings of ICML*, pages 1–8, New York, New York, USA. ACM Press.

Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. 2012. Kernels for Vector-Valued Functions: a Review. *Foundations and Trends in Machine Learning*, pages 1–37.

Daniel Beck, Kashif Shah, Trevor Cohn, and Lucia Specia. 2013. SHEF-Lite : When Less is More for Translation Quality Estimation. In *Proceedings of WMT13*, pages 337–342.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Edwin V. Bonilla, Kian Ming A. Chai, and Christopher K. I. Williams. 2008. Multi-task Gaussian Process Prediction. *Advances in Neural Information Processing Systems*.

Rich Caruana. 1997. Multitask Learning. *Machine Learning*, 28:41–75.

Trevor Cohn and Lucia Specia. 2013. Modelling Annotator Bias with Multi-task Gaussian Processes: An Application to Machine Translation Quality Estimation. In *Proceedings of ACL*.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of ACL*.

Jenny Rose Finkel and Christopher D. Manning. 2009. Hierarchical Bayesian Domain Adaptation. In *Proceedings of NAACL*.

James Hensman, Neil D Lawrence, and Magnus Rattray. 2013. Hierarchical Bayesian modelling of gene expression time series across irregularly sampled replicates and clusters. *BMC Bioinformatics*, 14:252.

Pasi Jylänki, Jarno Vanhatalo, and Aki Vehtari. 2011. Robust Gaussian Process Regression with a Student-t Likelihood. *Journal of Machine Learning Research*, 12:3227–3257.

Rada Mihalcea and Carlo Strapparava. 2012. Lyrics, Music, and Emotions. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 590–599.

Bo Pang and Lillian Lee. 2008. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Duborg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Daniel Preotiuc-Pietro and Trevor Cohn. 2013. A temporal model of text periodicities using Gaussian Processes. In *Proceedings of EMNLP*.

Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian processes for machine learning*, volume 1. MIT Press Cambridge.

Kashif Shah, Trevor Cohn, and Lucia Specia. 2013. An Investigation on the Effectiveness of Features for Translation Quality Estimation. In *Proceedings of MT Summit XIV*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and Fast - But is it Good?: Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of EMNLP*.

Carlo Strapparava and Rada Mihalcea. 2007. SemEval-2007 Task 14 : Affective Text. In *Proceedings of SEMEVAL*.

Carlo Strapparava and Rada Mihalcea. 2008. Learning to identify emotions in text. In *Proceedings of the 2008 ACM Symposium on Applied Computing*.

Christopher K. I. Williams and David Barber. 1998. Bayesian Classification with Gaussian Processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351.

Detecting Latent Ideology in Expert Text: Evidence From Academic Papers in Economics

Zubin Jelveh[1], **Bruce Kogut**[2], **and Suresh Naidu**[3]

[1]Dept. of Computer Science & Engineering, New York University
[2]Columbia Business School and Dept. of Sociology, Columbia University
[3]Dept. of Economics and SIPA, Columbia University
zj292@nyu.edu, bruce.kogut@columbia.edu, sn2430@columbia.edu

Abstract

Previous work on extracting ideology from text has focused on domains where expression of political views is expected, but it's unclear if current technology can work in domains where displays of ideology are considered inappropriate. We present a supervised ensemble n-gram model for ideology extraction with topic adjustments and apply it to one such domain: research papers written by academic economists. We show economists' political leanings can be correctly predicted, that our predictions generalize to new domains, and that they correlate with public policy-relevant research findings. We also present evidence that unsupervised models can under-perform in domains where ideological expression is discouraged.

1 Introduction

Recent advances in text mining demonstrate that political ideology can be predicted from text – often with great accuracy. Standard experimental settings in this literature are ones where ideology is explicit, such as speeches by American politicians or editorials by Israeli and Palestinian authors. An open question is whether ideology can be detected in arenas where it is strongly discouraged. A further consideration for applied researchers is whether these tools can offer insight into questions of import for policymakers. To address both of these issues, we examine one such domain that is both policy-relevant and where ideology is not overtly expressed: research papers written by academic economists.

Why economics? Economic ideas are important for shaping policy by influencing the public debate and setting the range of expert opinion on various policy options (Rodrik, 2014). Economics also views itself as a science (Chetty, 2013) carefully applying rigorous methodologies and using institutionalized safe-guards such as peer review. The field's most prominent research organization explicitly prohibits researchers from making policy recommendations in papers that it releases (National Bureau of Economic Research, 2010). Despite these measures, economics' close proximity to public policy decisions have led many to see it as being driven by ideology (A.S., 2010). Does this view of partisan economics have any empirical basis?

To answer the question of whether economics is politicized or neutral, we present a supervised ensemble n-gram model of ideology extraction with topic adjustments.[1] Our methodology is most closely related to Taddy (2013) and Gentzkow and Shapiro (2010), the latter of which used χ^2 tests to find phrases most associated with ideology as proxied by the language of U.S. Congresspersons. We improve on this methodology by accounting for ideological word choice within topics and incorporating an ensemble approach that increases predictive accuracy. We also motivate the need to adjust for topics even if doing so does not improve accuracy (although it does in this case). We further provide evidence that fully unsupervised methods (Mei et al., 2007; Lin et al., 2008; Ahmed and Xing, 2010; Paul and Girju, 2010; Eisenstein et al., 2011; Wang et al., 2012) may encounter difficulties learning latent ideological aspects when those aspects are not first order in the data.

Our algorithm is able to correctly predict the ideology of 69.2% of economists in our data purely from their academic output. We also show that our predictions generalize and are predictors of responses by a panel of top economists on issues of economic importance. In a companion paper (Jelveh et al., 2014), we further show that

[1]Grimmer and Stewart (2013) provide an overview of models used for ideology detection.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1804–1809,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

predicted ideologies are significantly correlated to economists' research findings. The latter result shows the relevance and applicability of these tools beyond the task of ideology extraction.

2 Data

Linking Economists to Their Political Activity: We obtain the member directory of the American Economics Association (AEA) and link it to two datasets: economists' political campaign contributions and petition signing activities. We obtain campaign contribution data from the Federal Election Commission's website and petition signing data from Hedengren et al. (2010). From this data, we construct a binary variable to indicate the ground-truth ideologies of economists. See our companion paper (Jelveh et al., 2014) for further details on the construction of this dataset. Revealed ideology through contributions and petitions is largely consistent. Of 441 economists appearing in both datasets, 83.4% showed agreement between contributions and petitions. For the final dataset of ground-truth authors we include all economists with campaign contributions and/or petition signatures, however, we drop those economists whose ideologies where different across the contribution and petition datasets. Overall, 60% of 2,204 economists with imputed ideologies in this final dataset are left-leaning while 40% lean rightwards.

Economic Papers Corpus: To create our corpus of academic writings by economists, we collect 17,503 working papers from NBER's website covering June 1973 to October 2011. We also obtained from JSTOR the fulltext of 62,888 research articles published in 93 journals in economics for the years 1991 to 2008. Combining the set of economists and papers leaves us with 2,171 authors with ground truth ideology and 17,870 papers they wrote. From the text of these papers we create n-grams of length two through eight. While n-grams greater than three words in length are uncommon, Margolin et al. (2013) demonstrate that ideological word choice can be detected by longer phrases. To capture other expressions of ideology not revealed in adjacent terms, we also include skipgrams of length two by combining non-adjacent terms that are three to five words apart. We remove phrases used by fewer than five authors.

Topic Adjustments: Table 1 presents the top 20 most conservative and liberal bigrams ranked by χ^2 scores from a Pearson's test of independence between phrase usage by left- and right-leaning economists. It appears that top ideological phrases are related to specific research subfields. For example, right-leaning terms 'free_bank', 'stock_return', and 'feder_reserv' are related to finance and left-leaning terms 'mental_health', 'child_care', and 'birth_weight' are related to health care. This observation leads us to ask: Are apparently ideological phrases merely a by-product of an economist's research interest rather than reflective of true ideology?

To see why this is a critical question, consider that ideology has both direct and indirect effects on word choice, the former of which is what we wish to capture. The indirect pathway is through topic: ideology may influence the research area an economist enters into, but not the word choice within that area. In that case, if more conservative economists choose macroeconomics, the observed correlation between macro-related phrases and right-leaning ideology would be spurious. The implication is that accounting for topics may not necessarily improve performance but provide evidence to support an underlying model of how ideology affects word choice. Therefore, to better capture the direct effect of ideology on phrase usage we adjust our predictions by topic by creating mappings from papers to topics. For a **topic mapping**, we predict economists' ideologies from their word choice *within* each topic and combine these results to form and overall prediction. We compare different supervised and unsupervised topic mappings and assess their predictive ability.

To create supervised topic mappings, we take advantage of the fact that economics papers are manually categorized by the Journal of Economic Literature (JEL). These codes are hierarchical indicators of an article's subject area. For example, the code C51 can be read, in increasing order of specificity, as Mathematical and Quantitative Methods (C), Econometric Modeling (C5), Model Construction and Estimation (C51). We construct two sets of topic mappings: **JEL1** derived from the 1st-level codes (e.g. C) and **JEL2** derived from the 2nd-level codes (e.g. C5). The former covers broad areas (e.g. macroeconomics, microeconomics, etc.) while the latter contains more refined ones (e.g. monetary policy, firm behavior, etc.).

For unsupervised mappings, we run Latent

Left-Leaning Bigrams	Right-Leaning Bigrams
mental_health	public_choic
post_keynesian	stock_return
child_care	feder_reserv
labor_market	yes_yes
health_care	market_valu
work_time	journal_financi
keynesian_econom	bank_note
high_school	money_suppli
polici_analys	free_bank
analys_politiqu	liquid_effect
politiqu_vol	journal_financ
birth_weight	median_voter
labor_forc	law_econom
journal_post	vote_share
latin_america	war_spend
mental_ill	journal_law
medic_care	money_demand
labour_market	gold_reserv
social_capit	anna_j
singl_mother	switch_cost

Table 1: Top 20 bigrams and trigrams.

Dirichilet Allocation (Blei et al., 2003) on our corpus. We use 30, 50, and 100 topics to create **LDA30**, **LDA50**, and **LDA100** topic mappings. We use the topic distributions estimated by LDA to assign articles to topics. A paper p is assigned to a topic t if the probability that t appears in p is greater than 5%. While 5% might seem to be a lower threshold, the topic distributions estimated by LDA tend to be sparse. For example, even with 50 topics to 'choose' from in **LDA50** and a threshold of 5%, 99.5% of the papers would be assigned to five or fewer topics. This compares favorably with JEL2 codings where 98.8% of papers have five or fewer topics.

3 Algorithm

There are two components to our topic-adjusted algorithm for ideology prediction. First, we focus on n-grams and skipgrams that are most correlated with ideology in the training data. For each topic within a topic mapping, we count the total number of times each phrase is used by all left- and all right-leaning economists. Then, we compute Pearson's χ^2 statistic and associated p-values and keep phrases with p ≤ 0.05. As an additional filter, we split the data into ten folds and perform the χ^2 test within each fold. For each topic, we keep phrases that are consistently ideological across all folds. This greatly reduces the number of ideological phrases. For **LDA50**, the mean number of ideological phrases per topic before the cross validation filter is 12,932 but falls to 963 afterwards.

With the list of ideological phrases in hand, the second step is to iterate over each topic and predict the ideologies of economists in our test set. To compute the predictions we perform partial least squares (PLS): With our training data, we construct the standardized frequency matrix $\mathbf{F}_{t,train}$ where the (e,p)-th entry is the number of times economist e used partisan phrase p across all of e's papers in t. This number is divided by the total number of phrases used by e in topic t. For papers with multiple authors, each author gets same count of phrases. About 5% of the papers in our dataset are written by authors with differing ideologies. We do not treat these differently. Columns of $\mathbf{F}_{t,train}$ are standardized to have unit variance. Let $\mathbf{y}$ be the vector of ground-truth ideologies, test set ideologies are predicted as follows:

1) Compute $\mathbf{w} = Corr(\mathbf{F}_{t,train}, \mathbf{y})$, the correlations between each phrase and ideology
2) Project to one dimension: $\mathbf{z} = \mathbf{F}_{t,train}\mathbf{w}$
3) Regress ideology, $\mathbf{y}$, on the constructed variable $\mathbf{z}$: $\mathbf{y} = b_1\,\mathbf{z}$
4) Predict ideology $\hat{y}_e$ of new economist by $\hat{y}_e = b_1 \tilde{\mathbf{f_e}}'\mathbf{w}$, ($\tilde{\mathbf{f_e}}$ is scaled frequency vector)

To avoid over-fitting we introduce an ensemble element: For each t, we sample from the list of significant n-grams in t and sample with replacement from the authors who have written in t.[2] PLS is performed on this sample data 125 times. Each PLS iteration can be viewed as a vote on whether an author is left- or right-leaning. We calculate the vote as follows. For each iteration, we predict the ideologies of economists in the *training data*. We find the threshold f that minimizes the distance between the true and false positive rates for the current iteration and the same rates for the perfect classifier: 1.0 and 0.0, respectively. Then, an author in the test set is voted left-leaning if $y_{t,test} \leq f$ and right-leaning otherwise.

For a given topic mapping, our algorithm returns a three-dimensional array with the (e,t,c)-th entry representing the number of votes economist e received in topic t for ideology c (left- or right-

[2]The number of phrases sampled each iteration is the square root of the number of ideological phrases in the topic.

leaning). To produce a final prediction, we sum across the second dimension and compute ideology as the percentage of right-leaning votes received across all topics within a topic-mapping. Therefore, ideology values closer to zero are associated with a left-leaning ideology and values closer to one are associated with a rightward lean.

To recap, we start with a topic mapping and then for each topic run an ensemble algorithm with PLS at its core.[3] The output for each topic is a set of votes. We sum across topics to compute a final prediction for ideology.

4 Validation and Results

We split our ground-truth set of 2,171 authors into training (80%) and test sets (20%) and compute predictions as above. As our data exhibits skew with 1.5 left-leaning for every right-leaning economist, we report the area under the curve (AUC) which is robust to class skew (Fawcett, 2006). It's worth noting that a classifier that randomly predicts a liberal economist 60% of the time would have an AUC of 0.5. To compare our model with fully unsupervised methods, we also include results from running the Topic-Aspect Model (TAM) (Paul and Girju, 2010) on our data. TAM decomposes documents into two components: one affecting topics and one affecting a latent aspect that influences all topics in a similar manner. We run TAM with 30 topics and 2 aspects (**TAM2/30**). We follow Paul and Girju and use the learned topic and aspect distributions as training data for a SVM.[4]

Columns 2 to 4 from Table 2 show that our models' predictions have a clear association with ground-truth ideology. The LDA topic mappings outperform the supervised mappings as well as a model that does not adjust for topics (**NoTopic**). Perhaps not surprisingly, TAM does not perform well in our domain. A drawback of unsupervised methods is that the learned aspects may not be related to ideology but some other hidden factor.

For further insight into how well our model generalizes, we use data from Gordon and Dahl (2013) to compare our predictions to potentially ideological responses of economists on a survey conducted by the University of Chicago.[5] Each survey question asks for an economists opinion on an issue of political relevance such as minimum wages or tax rates. For further details on the data see Gordon and Dahl. Of importance here is that Gordon and Dahl categorize 22 questions where agreement (disagreement) with the statement implies belief a conservative (liberal) viewpoint.

To see if our predicted ideologies are correlated with survey responses, we run an ordered-logistic regression (McCullagh, 1980). Survey responses are coded with the following order: Strongly Disagree, Disagree, Uncertain, Agree, Strongly Agree. We regress survey responses onto predicted ideologies. We also include question-level dummies and explanatory variables for a re-

(1) Topic Map	(2) Accuracy(%)	(3) Corr. w/ Truth	(4) AUC
LDA50	**69.2**	**0.381**	**0.719**
LDA100	66.3	0.364	0.707
LDA30	65.0	0.313	0.674
NoTopic	63.9	0.290	0.672
JEL1	61.0	0.263	0.647
JEL2	61.8	0.240	0.646
TAM2/30	61.5	0.228	0.580

Table 2: Model comparisons

	(1)	(2)	(3)
LDA50	1.814***	2.457***	2.243***
Log-Lik.	-1075.0	-758.7	-740.6
JEL1	1.450***	2.128***	1.799***
Log-Lik.	-1075.3	-757.4	-740.5
No Topic	0.524***	0.659***	0.824***
Log-Lik.	-1075.3	-760.5	-741.0
Question	No	Yes	Yes
Demog./Prof.	No	No	Yes
Observations	715	715	715
Individuals	39	39	39

$^{*}\ p < 0.10$, $^{**}\ p < 0.05$, $^{***}\ p < 0.01$

Table 3: IGM correlations. Column (1) shows results of regression of response on predicted ideology. Column (2) adds question dummies. Column (3) adds demographic and professional variables.

[3] Other predictions algorithms could be dropped in for PLS. Logistic regression and SVM produced similar results.

[4] Authors are treated as documents. TAM is run for 1,000 iterations with the following priors: $\alpha = 1.0$, $\beta = 1.0$, $\gamma_0 = 1$, $\gamma_1 = 1$, $\delta_0 = 20$, $\delta_1 = 80$.

[5] http://igmchicago.org

spondent's gender, year of Ph.D., Ph.D. university, NBER membership, and experience in federal government. Table 3 shows the results of these regressions for three topic mappings. The correlation between our predictions and survey respondents are all strongly significant.

One way to interpret these results is to compare the change in predicted probability of providing an Agree or Strongly Agree answer (agreeing with the conservative view point) if we change predicted ideology from most liberal to most conservative. For **NoTopic**, this predicted probability is 35% when ideology is set to most liberal and jumps to 73.7% when set to most conservative. This difference increases for topic-adjusted models. For **LDA50**, the probability of a conservative answer when ideology is set to most liberal is 14.5% and 93.8% for most conservative.

Figure 1 compares the predicted probabilities of choosing different answers when ideology is set to most liberal and most conservative. Our topic-adjusted models suggest that the most conservative economists are much more likely to strongly agree with a conservative response than for the most liberal economists to strongly agree with a liberal response. It is worthwhile to note from the small increase in log-likelihood in Table 3 when controls are added, suggesting that our ideology scores are much better predictors of IGM responses than demographic and professional controls.

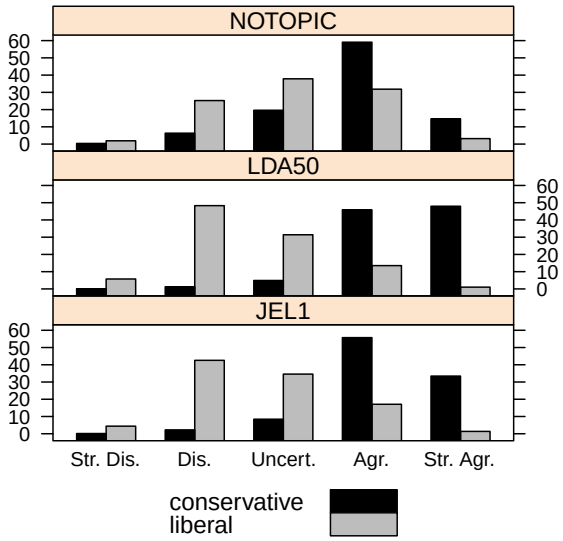


Figure 1: The predicted probability of agreeing with a conservative response when ideology is set to most liberal (gray) and most conservative (black).

5 Conclusions and Future Work

We've presented a supervised methodology for extracting political sentiment in a domain where it's discouraged and shown how it even predicts the partisanship calculated from completely unrelated IGM survey data. In a companion paper (Jelveh et al., 2014) we further demonstrate how this tool can be used to aid policymakers in de-biasing research findings. When compared to domains where ideological language is expected, our predictive ability is reduced. Future work should disentangle how much this difference is due to modeling decisions and limitations versus actual absence of ideology. Future works should also investigate how fully unsupervised methods can be extended to match our performance.

Acknowledgement

This work was supported in part by the NSF (under grant 0966187). The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of any of the sponsors.

References

Amr Ahmed and Eric P. Xing. 2010. Staying informed: supervised and semi-supervised multi-view topical analysis of ideological perspective. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1140–1150. Association for Computational Linguistics.

A.S. 2010. Is economics a right-wing conspiracy? *The Economist*, August.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Raj Chetty. 2013. Yes, economics is a science. *The New York Times*, October.

Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. 2011. Sparse additive generative models of text. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1041–1048.

T. Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874.

Matthew Gentzkow and Jesse M. Shapiro. 2010. What drives media slant? evidence from U.S. daily newspapers. *Econometrica*, 78(1):35–71.

Roger Gordon and Gordon B Dahl. 2013. Views among economists: Professional consensus or point-counterpoint? *American Economic Review*, 103(3):629–635, May.

J. Grimmer and B. M. Stewart. 2013. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, 21(3):267–297, January.

David Hedengren, Daniel B. Klein, and Carrie Milton. 2010. Economist petitions: Ideology revealed. *Econ Journal Watch*, 7(3):288–319.

Zubin Jelveh, Bruce Kogut, and Suresh Naidu. 2014. Political language in economics.

Wei-Hao Lin, Eric Xing, and Alexander Hauptmann. 2008. A joint topic and perspective model for ideological discourse. In *Machine Learning and Knowledge Discovery in Databases*, pages 17–32. Springer.

Drew Margolin, Yu-Ru Lin, and David Lazer. 2013. Why so similar?: Identifying semantic organizing processes in large textual corpora.

Peter McCullagh. 1980. Regression models for ordinal data. *Journal of the royal statistical society. Series B (Methodological)*, pages 109–142.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th international conference on World Wide Web*, pages 171–180. ACM.

National Bureau of Economic Research. 2010. Amended and restated by-laws of national bureau of economic research, inc.

Michael Paul and Roxana Girju. 2010. A two-dimensional topic-aspect model for discovering multi-faceted topics. *Urbana*, 51.

Dani Rodrik. 2014. When ideas trump interests: Preferences, worldviews, and policy innovations . *Journal of Economic Perspectives*, 28(1):189–208, February.

Matt Taddy. 2013. Multinomial inverse regression for text analysis. *Journal of the American Statistical Association*, 108.

William Yang Wang, Elijah Mayfield, Suresh Naidu, and Jeremiah Dittmar. 2012. Historical analysis of legal opinions with a sparse mixed-effects latent variable model. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 740–749. Association for Computational Linguistics.

A Model of Individual Differences in Gaze Control During Reading

Niels Landwehr[1] and **Sebastian Arzt**[1] and **Tobias Scheffer**[1] and **Reinhold Kliegl**[2]
[1] Department of Computer Science, Universität Potsdam
August-Bebel-Straße 89, 14482 Potsdam, Germany
{landwehr, sarzt, scheffer}@cs.uni-potsdam.de
[2] Department of Psychology, Universität Potsdam
Karl-Liebknecht-Straße 24/25, 14476 Potsdam OT/Golm
kliegl@uni-potsdam.de

Abstract

We develop a statistical model of saccadic eye movements during reading of isolated sentences. The model is focused on representing individual differences between readers and supports the inference of the most likely reader for a novel set of eye movement patterns. We empirically study the model for biometric reader identification using eye-tracking data collected from 20 individuals and observe that the model distinguishes between 20 readers with an accuracy of up to 98%.

1 Introduction

During skilled reading, the eyes of a reader do not move smoothly over a text. Instead, reading proceeds by alternating between brief fixations on individual words and short ballistic movements called *saccades* that move the point of fixation to a new location. Evidence in psychological research indicates that patterns of fixations and saccades are driven partly by low-level visual cues (*e.g.*, word length), and partly by linguistic and cognitive processing of the text (Kliegl et al., 2006; Rayner, 1998).

Eye-movement patterns are frequently studied in cognitive psychology as they provide a rich and detailed record of the visual, oculomotor, and linguistic processes involved in reading. Computational models of eye-movement control developed in psychology, such as SWIFT (Engbert et al., 2005; Schad and Engbert, 2012) or E-Z Reader (Reichle et al., 1998; Reichle et al., 2012), simulate the generation of eye movements based on physiological and psychological constraints related to attention, visual perception, and the oculomotor system. Recently, the problem of modeling eye movements has also been approached from a machine-learning perspective. Matties and Søgaard (2013) and Hara et al. (2012) study conditional random field models to predict which words in a text are fixated by a reader. Nilsson and Nivre (2009) use a transition-based log-linear model to predict a sequence of fixations for a text.

A central observation made by these studies, as well as by earlier psychological work (Erdmann and Dodge, 1898; Huey, 1908), is that eye movement patterns vary significantly between individuals. As one example of the strength of individual differences in reading eye movements, we cite Dixon (1951) who compared the reading processes of university professors and graduate students of physics, education, and history on reading material in their own and the two other fields. He did not find strong effects of his experimental variables (i.e., field of research, expertise in research) but "if there is one thing that this study has shown, it is that individual differences in reading skill existed among the subjects of all departments. Fast and slow readers were found in every department, and the overlapping of distributions from passage to passage was enormous" (p. 173). Even though it is possible to predict across a large base of readers with some accuracy whether specific words will be fixated (Matties and Søgaard, 2013), a strong variation between readers in attributes such as the fraction of skipped words and total number of saccades has been observed (Hara et al., 2012).

Some recent work has studied eye movement patterns as a biometric feature. Most studies are based on an artificial visual stimulus, such as a moving (Kasprowski and Ober, 2004; Komogortsev et al., 2010; Rigas et al., 2012b; Zhang and Juhola, 2012) or fixed (Bednarik et al., 2005) dot on a computer screen, or a specific image stimulus (Rigas et al., 2012a). In the most common use

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1810–1815,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

case of biometric user identification, a decision on whether access should be granted has to be made after performing some test that requires the user's attention and therefore cannot take a long time. By contrast, our work is motivated by a less intrusive scenario in which the user is monitored continuously during access to, for instance, a device or document. When the accumulated evidence supports the conclusion that the user is not authorized, access can be terminated or additional credentials requested. In this use case, identification has to be based on saccadic eye movements that occur while a user is reading an arbitrary text—as opposed to movements that occur in response to a fixed, controlled visual stimulus. Holland and Komogortsev (2012) study reader recognition based on a set of aggregate features derived from eye movements, irrespective of the text being read; their work will serve as reference in our empirical study.

The paper is organized as follows. Section 2 details the problem setting. Section 3 introduces the statistical model and discusses parameter estimation and inference. Section 5 presents empirical results, Section 6 concludes.

2 Problem Setting and Notation

Let $\mathcal{R}$ denote a set of readers, and $\mathcal{X} = \{\mathbf{X}_1, ..., \mathbf{X}_n\}$ a set of texts. Each $r \in \mathcal{R}$ generates a set of eye movement patterns $\mathcal{S}^{(r)} = \{\mathbf{S}_1^{(r)}, \ldots, \mathbf{S}_n^{(r)}\}$ on the set of texts $\mathcal{X}$, by

$$\mathbf{S}_i^{(r)} \sim p(\mathbf{S}|\mathbf{X}_i, r)$$

where $p(\mathbf{S}|\mathbf{X}, r)$ is a reader-specific distribution over eye movement patterns given a text $\mathbf{X}$. A pattern is a sequence $\mathbf{S} = ((s_1, d_1), \ldots, (s_T, d_T))$ of fixations, consisting of a fixation position s_t (position in text that was fixated) and duration $d_t \in \mathbb{R}$ (length of fixation in milliseconds). In our experiments, individual sentences are presented in a single line on a screen, thus we only model a horizontal gaze position $s_t \in \mathbb{R}$.

At test time, we observe novel eye movement patterns $\bar{\mathcal{S}} = \{\bar{\mathbf{S}}_1, \ldots, \bar{\mathbf{S}}_m\}$ on a novel set of texts $\bar{\mathcal{X}} = \{\bar{\mathbf{X}}_1, ..., \bar{\mathbf{X}}_m\}$ generated by an unknown reader $r \in \mathcal{R}$. The goal is to infer

$$r_* = \arg\max_{r \in \mathcal{R}} p(r|\bar{\mathcal{S}}, \bar{\mathcal{X}}). \quad (1)$$

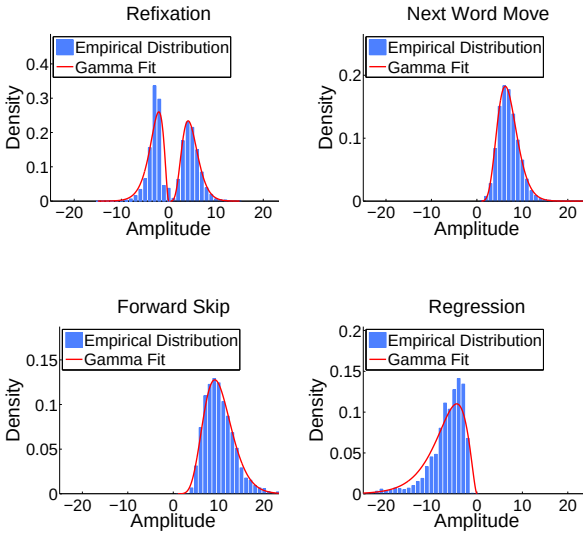


Figure 1: Empirical distributions over amplitudes and Gamma fits for different saccade types.

3 Statistical Model of Eye Movements

We solve Problem 1 by estimating reader-specific models $p(\mathbf{S}|\mathbf{X}; \boldsymbol{\theta}_r)$ for $r \in \mathcal{R}$, and solving for

$$p(r|\bar{\mathcal{S}}, \bar{\mathcal{X}}; \boldsymbol{\Theta}) \propto \left(\prod_{i=1}^{m} p(\bar{\mathbf{S}}_i|\bar{\mathbf{X}}_i; \boldsymbol{\theta}_r) \right) p(r) \quad (2)$$

where all $\boldsymbol{\theta}_r$ are aggregated into a global model $\boldsymbol{\Theta}$. Assuming a uniform prior $p(r)$ over readers, this reduces to predicting the reader r that maximizes the likelihood $p(\bar{\mathcal{S}}|\bar{\mathcal{X}}; \boldsymbol{\theta}_r) = \prod_{i=1}^{m} p(\bar{\mathbf{S}}_i|\bar{\mathbf{X}}_i; \boldsymbol{\theta}_r)$.

We formulate a model $p(\mathbf{S}|\mathbf{X}; \boldsymbol{\theta})$ of a sequence $\mathbf{S}$ of fixations given a text $\mathbf{X}$. The model defines a dynamic probabilistic process that successively generates the fixation positions s_t and durations d_t in $\mathbf{S}$, reflecting how a reader generates a sequence of saccades in response to a text stimulus $\mathbf{X}$. The joint distribution over all fixation positions and durations is assumed to factorize as

$$p(s_1, \ldots, s_T, d_1, \ldots, d_T|\mathbf{X}; \boldsymbol{\theta}) = p(s_1, d_1|\mathbf{X}; \boldsymbol{\theta}) \prod_{t=1}^{T-1} p(s_{t+1}, d_{t+1}|s_t, \mathbf{X}; \boldsymbol{\theta}).$$

The conditional $p(s_{t+1}, d_{t+1}|s_t, \mathbf{X}; \boldsymbol{\theta})$ models the generation of the next fixation position and duration given the current fixation position s_t. In the psychological literature, four different *saccadic types* are distinguished: a reader can refixate the current word (refixation), fixate the next word in the text (next word movement), move the

fixation to a word after the next word (forward skip), or regress to fixate a word occurring earlier than the currently fixated word in the text (regression) (Heister et al., 2012). We observe empirically, that modeling the amplitude as a mixture of four Gamma distributions matches the empirical distribution of amplitudes in our data well—see Figure 1. Modeling the amplitudes as a single distribution, instead of a mixture of four distributions, results in a substantially lower out-of-sampling likelihood of the model. Therefore, at each time t, the model first draws a saccadic type $u_{t+1} \in \{1, 2, 3, 4\}$ from a multinomial distribution and then generates a saccade amplitude a_{t+1} and fixation duration d_{t+1} from type-specific Gamma distributions. Formally, the generative process is given by

$$u_{t+1} \sim p(u|\boldsymbol{\pi}) = \text{Mult}(u|\boldsymbol{\pi}) \quad (3)$$
$$a_{t+1} \sim p(a|u_{t+1}, s_t, \mathbf{X}; \boldsymbol{\eta}) \quad (4)$$
$$d_{t+1} \sim p(d|u_{t+1}; \boldsymbol{\lambda}). \quad (5)$$

Afterwards the model updates the fixation position according to $s_{t+1} = s_t + a_{t+1}$. The joint parameter vector $\boldsymbol{\theta}$ concatenates parameters of the individual distributions in Equations 3 to 5. Figure 2 shows a slice in the dynamical model.

Given the current fixation position s_t, the text $\mathbf{X}$, and the chosen saccadic type u_{t+1}, the amplitude is constrained to fall within a specific interval—for instance, within the characters of the currently fixated word for refixations. Therefore, we model the distribution over the saccade amplitude given the saccadic type (Equation 4) as truncated Gamma distributions, given by[1]

$$\mathcal{G}(x|[l,r];\alpha,\beta) = \begin{cases} \frac{\mathcal{G}(x|\alpha,\beta)}{\int_l^r \mathcal{G}(\bar{x}|\alpha,\beta)\mathrm{d}\bar{x}} & \text{if } x \in [l,r] \\ 0 & \text{otherwise} \end{cases}$$

$$\text{where} \quad \mathcal{G}(x|\alpha,\beta) = \frac{1}{\beta^\alpha \Gamma(\alpha)} x^{\alpha-1} e^{\frac{-x}{\beta}}$$

is the Gamma distribution with shape parameter α and scale parameter β, and Γ is the Gamma function. For $x \sim \mathcal{G}(x|\alpha,\beta)$ it holds that $\mathcal{G}(x|[l,r];\alpha,\beta)$ is the conditional distribution of x given that $x \in [l,r]$. The distribution over a saccade amplitude given the saccade type is given by

$$p(a|u_{t+1}=1, s_t, \mathbf{X}; \boldsymbol{\eta}) = \begin{cases} \mu\mathcal{G}(a|[0,r];\alpha_1,\beta_1) & \text{if } a > 0 \\ (1-\mu)\mathcal{G}(-a|[0,l];\bar{\alpha}_1,\bar{\beta}_1) & \text{otherwise} \end{cases} \quad (6)$$

where the parameter μ reflects the probability for a forward saccade within a refixation, and

$$\begin{aligned} p(a|u_{t+1}=2, s_t, \mathbf{X}; \boldsymbol{\eta}) &= \mathcal{G}(a|[l^+, r^+];\alpha_2,\beta_2) \\ p(a|u_{t+1}=3, s_t, \mathbf{X}; \boldsymbol{\eta}) &= \mathcal{G}(a|(r^+, \infty);\alpha_3,\beta_3) \\ p(a|u_{t+1}=4, s_t, \mathbf{X}; \boldsymbol{\eta}) &= \mathcal{G}(-a|(-l, \infty);\alpha_4,\beta_4). \end{aligned} \quad (7)$$

Here, the truncation intervals reflect the constraints on the amplitude a_{t+1} given u_{t+1}, s_t and $\mathbf{X}$. Let w_l (w_r) denote the position of the left-most (right-most) character of the currently fixated word, and let w_l^+, w_r^+ denote these positions for the word following the currently fixated word. Then $l = w_l - s_t$, $r = w_r - s_t$, $l^+ = w_l^+ - s_t$, and $r^+ = w_r^+ - s_t$. The parameter vector $\boldsymbol{\eta}$ contains the parameters μ, $\bar{\alpha_1}$, $\bar{\beta_1}$ and α_i, β_i for $i \in \{2, 3, 4\}$.

The distribution over fixation durations given saccade type is modeled by a Gamma distribution

$$p(d|u_{t+1}; \boldsymbol{\lambda}) = \mathcal{G}(d|\gamma_{u_{t+1}}, \delta_{u_{t+1}})$$

with type-specific parameters γ_u, δ_u for $u \in \{1, 2, 3, 4\}$ that are concatenated into a parameter vector $\boldsymbol{\lambda}$.

It remains to specify the distribution over initial fixation positions and durations $p(s_1, d_1|\mathbf{X}; \boldsymbol{\theta})$, which is given by additional Gamma distributions

$$s_1 \sim \mathcal{G}(d|\alpha_0, \beta_0) \qquad d_1 \sim \mathcal{G}(d|\gamma_0, \delta_0)$$

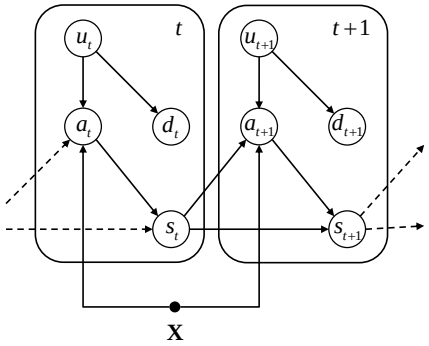


Figure 2: Graphical model notation of a slice in the dynamic model. Parameters are omitted to avoid notational clutter.

[1]The definition is straightforwardly generalized to open truncation intervals.

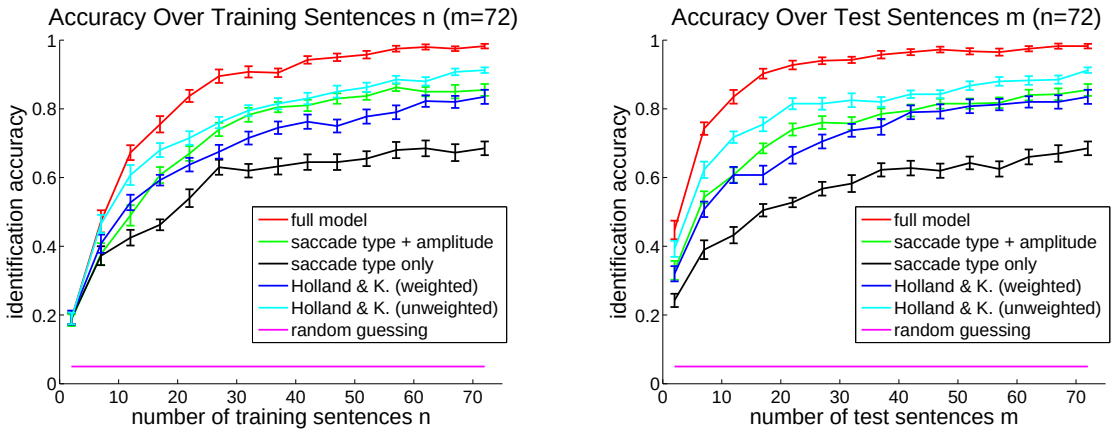


Figure 3: Reader identification accuracy as a function of the number of training sentences (left) and test sentences (right) read for different model variants. Error bars indicate the standard error.

where the parameters $\alpha_0, \beta_0, \gamma_0, \delta_0$ are aggregated into the joint parameter vector $\boldsymbol{\theta}$.

4 Parameter Estimation and Inference

Given a set $\mathcal{S}^{(r)}$ of eye movement observations for reader $r \in \mathcal{R}$ on texts $\mathcal{X}$, the MAP estimate of the parameters is

$$\begin{aligned}\boldsymbol{\theta}_r &= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{S}^{(r)}, \mathcal{X}) \\ &= \arg\max_{\boldsymbol{\theta}} \left(\prod_{i=1}^{n} p(\mathbf{S}_i^{(r)}|\mathbf{X}_i; \boldsymbol{\theta}) \right) p(\boldsymbol{\theta}). \quad (8)\end{aligned}$$

A Dirichlet distribution (add-one smoothing) is a natural, conjugate prior for the multinomial distribution; we use uninformative priors for all other distributions. The structure of the model implies that the posterior can be maximized by fitting the parameters $\boldsymbol{\pi}$ to the observed saccadic types under the Dirichlet prior, and independently fitting the distributions $p(a_t|u_t, \mathbf{X}, s_t; \boldsymbol{\eta})$ and $p(d_t|u_t; \boldsymbol{\lambda})$ by maximum likelihood to the saccade amplitudes and durations observed for each saccade type. The resulting maximum likelihood problems are slightly non-standard in that we have to fit Gamma distributions that are truncated differently for each data point, depending on the textual content at the position where the saccade occurs (see Equations 6 and 7). We solve the resulting optimization problems using a Quasi-Newton method. To avoid overfitting, we use a backoff-smoothing technique for $p(a_t|u_t, \mathbf{X}, s_t; \boldsymbol{\eta})$ and $p(d_t|u_t; \boldsymbol{\lambda})$: we replace reader-specific parameter estimates by estimates obtained from the corresponding data of all readers if the number of data points from which the distributions are estimated falls below a cutoff value. The cutoff value is tuned by cross-validation on the training data.

At test time, we have to infer likelihoods $p(S_i|\mathbf{X}; \boldsymbol{\theta}_r)$ (Equation 2). This is done by evaluating the multinomial and (truncated) Gamma distributions in the model for the corresponding observations and model parameters.

5 Empirical Study

We empirically study the proposed model and several baselines using eye-movement records of 20 individuals (Heister et al., 2012). For each individual, eye movements are recorded while reading each of the 144 sentences in the *Potsdam Sentence Corpus* (Kliegl et al., 2006). The data set contains fixation positions and durations that have been obtained from raw eye movement data by appropriate preprocessing. Eye movements were recorded with an EyeLink II system with a 500-Hz sampling rate (SR Research, Osgoode, Ontario, Canada). All recordings and calibration were binocular. We randomly sample disjoint sets of n training sentences and m test sentences from the set of 144 sentences. Models are estimated on the eye movement records of individuals on the training sentences (Equation 8). The eye-movement records of one individual on all test sentences constitute a test example; the model infers the most likely individual to have generated these test observations (Equation 2). *Identification accuracy* is the fraction of times an individual is correctly inferred; random guessing yields an accuracy of 0.05. Results are averaged over 20 training and test sets.

We study the model introduced in Section 3

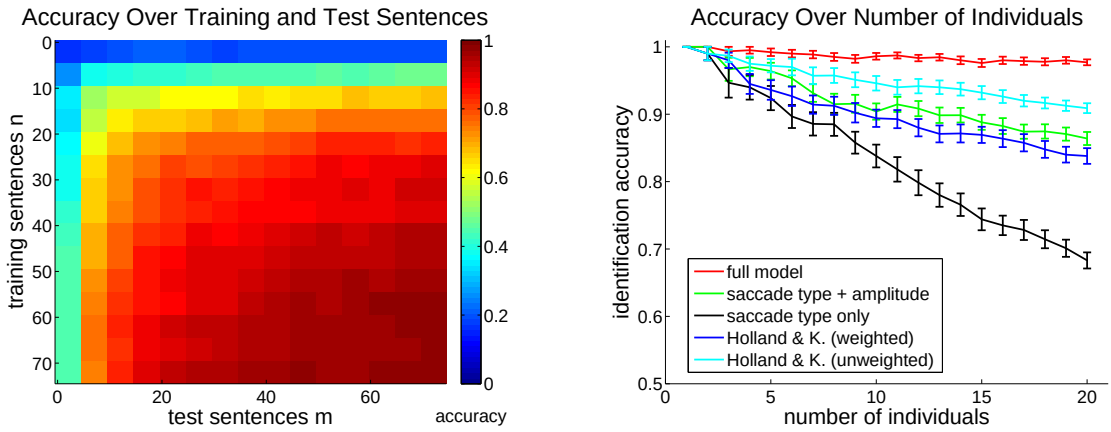


Figure 4: Identification accuracy as a function of the number of training and test sentences read for full model (left). Identification accuracy as a function of the number of individuals that have to be distinguished for different model variants (right). Error bars indicate the standard error.

(*full model*), a model variant in which the variable d_{t+1} and corresponding distribution is removed (*saccade type + amplitude*), and a simple model that only fits a multinomial distribution to saccade types (*saccade type only*). Additionally, we compare against the feature-based reader identification approach by Holland & Komogortsev (2012). Six of the 14 features used by Holland & Komogortsev depend on saccade velocities and vertical fixation positions. As this information was not available in the preprocessed data set that we used, we implemented the remaining features. There is extensive empirical evidence that saccade velocity scales with saccade amplitude. Specifically, the relationship between logarithmic peak saccade velocity and logarithmic saccade amplitude is linear over a wide range of amplitudes and velocities; this is known as the main sequence relationship (Bahill et al., 1975). Therefore, we do not expect that adding saccade velocities would dramatically affect performance of this baseline. Holland & Komogortsev employ a weighted combination of features; we report results for the method with and without feature weighting.

Figure 3 shows identification accuracy as a function of the number n of training sentences used to estimate model parameters (left) and as a function of the number m of test sentences on which inference of the most likely reader is based (right, cf. Equation 2). The full model achieves up to 98.25% accuracy, significantly outperforming the Holland & Komogortsev (2012) baseline (91.25%, without feature weighting) and simpler model variants. All methods perform much better than random guessing. Figure 4 (left) shows identification accuracy as a function of both training size n and test size m for the full model.

We finally study how identification accuracy changes with the number of individuals that have to be distinguished. To this end, we perform the same study as above, but with randomly sampled subsets of the overall set of 20 individuals. In these experiments, we average over 50 random train-test splits. Figure 4 (right) shows identification accuracy as a function of the number of individuals. We observe that identification accuracy drops with the number of individuals for all methods; our model consistently outperforms the baselines.

6 Conclusions

We have developed a model of individual differences in eye movements during reading, and studied its application in a biometric task. At test time, individuals are identified based on eye movements on novel text. Our approach thus provides potentially unobtrusive biometric identification without requiring users to react to a specific stimulus. Empirical results show clear advantages over an existing approach for reader identification.

Acknowledgments

We would like to thank Christoph Sawade for insightful discussions and help with the eye movement data. We gratefully acknowledge support from the German Research Foundation (DFG), grant LA 3270/1-1.

References

A. Terry Bahill, Michael R. Clark, and Lawrence Stark. 1975. The main sequence: a tool for studying human eye movements. *Mathematical Biosciences*, 24:191–204.

Roman Bednarik, Tomi Kinnunen, Andrei Mihaila, and Pasi Fränti. 2005. Eye-movements as a biometric. In *Proceedings of the 14th Scandinavian Conference on Image Analysis*.

W. Robert Dixon. 1951. Studies in the psychology of reading. In W. S. Morse, P. A. Ballantine, and W. R. Dixon, editors, *Univ. of Michigan Monographs in Education No. 4*. Univ. of Michigan Press.

Ralf Engbert, Antje Nuthmann, Eike M. Richter, and Reinhold Kliegl. 2005. SWIFT: A dynamical model of saccade generation during reading. *Psychological Review*, 112(4):777–813.

Bruno Erdmann and Raymond Dodge. 1898. *Psychologische Untersuchungen über das Lesen*. Halle: Max Niemeyer.

Tadayoshi Hara, Daichi Mochihashi, Yoshino Kano, and Akiko Aizawa. 2012. Predicting word fixations in text with a CRF model for capturing general reading strategies among readers. In *Proceedings of the First Workshop on Eye-Tracking and Natural Language Processing*.

Julian Heister, Kay-Michael Würzner, and Reinhold Kliegl. 2012. Analysing large datasets of eye movements during reading. In James S. Adelman, editor, *Visual word recognition. Vol. 2: Meaning and context, individuals and development*, pages 102–130.

Corey Holland and Oleg V. Komogortsev. 2012. Biometric identification via eye movement scanpaths in reading. In *Proceedings of the 2011 International Joint Conference on Biometrics*.

Edmund B. Huey. 1908. *The psychology and pedagogy of reading*. Cambridge, Mass.: MIT Press.

Pawel Kasprowski and Jozef Ober. 2004. Eye movements in biometrics. In *Proceedings of the 2004 International Biometric Authentication Workshop*.

Reinhold Kliegl, Antje Nuthmann, and Ralf Engbert. 2006. Tracking the mind during reading: The influence of past, present, and future words on fixation durations. *Journal of Experimental Psychology: General*, 135(1):12–35.

Oleg V. Komogortsev, Sampath Jayarathna, Cecilia R. Aragon, and Mechehoul Mahmoud. 2010. Biometric identification via an oculomotor plant mathematical model. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*.

Franz Matties and Anders Søgaard. 2013. With blinkers on: robust prediction of eye movements across readers. In *Proceedings of the 2013 Conference on Empirical Natural Language Processing*.

Mattias Nilsson and Joakim Nivre. 2009. Learning where to look: Modeling eye movements in reading. In *Proceedings of the 13th Conference on Computational Natural Language Learning*.

Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124(3):372–422.

Erik D. Reichle, Tessa Warren, and Kerry McConnell. 2012. Using e-z reader to model the effects of higher-level language processing on eye movements during reading. *Psychonomic Bulletin & Review*, 16(1):1–21.

Ioannis Rigas, George Economou, and Spiros Fotopoulos. 2012a. Biometric identification based on the eye movements and graph matching techniques. *Pattern Recognition Letters*, 33(6).

Ioannis Rigas, George Economou, and Spiros Fotopoulos. 2012b. Human eye movements as a trait for biometrical identification. In *Proceedings of the IEEE 5th International Conference on Biometrics: Theory, Applications and Systems*.

Daniel Schad and Ralf Engbert. 2012. The zoom lens of attention: Simulating shuffled versus normal text reading using the swift model. *Visual Cognition*, 20(4-5):391–421.

Youming Zhang and Martti Juhola. 2012. On biometric verification of a user by means of eye movement data mining. In *Proceedings of the 2nd International Conference on Advances in Information Mining and Management*.

Muli-label Text Categorization with Hidden Components

Li Li Longkai Zhang Houfeng Wang
Key Laboratory of Computational Linguistics (Peking University) Ministry of Education, China
li.l@pku.edu.cn, zhlongk@qq.com, wanghf@pku.edu.cn

Abstract

Multi-label text categorization (MTC) is supervised learning, where a document may be assigned with multiple categories (labels) simultaneously. The labels in the MTC are correlated and the correlation results in some hidden components, which represent the "share" variance of correlated labels. In this paper, we propose a method with hidden components for MTC. The proposed method employs PCA to capture the hidden components, and incorporates them into a joint learning framework to improve the performance. Experiments with real-world data sets and evaluation metrics validate the effectiveness of the proposed method.

1 Introduction

Many real-world text categorization applications are multi-label text categorization (Srivastava and Zane-Ulman, 2005; Katakis et al., 2008; Rubin et al., 2012; Nam et al., 2013), where a documents is usually assigned with *multiple* labels simultaneously. For example, as figure 1 shows, a newspaper article concerning global warming can be classified into two categories, *Environment*, and *Science* simultaneously. Let $\mathcal{X} = R^d$ be the documents corpus, and $\mathcal{Y} = \{0, 1\}^m$ be the label space with m labels. We denote by $\{(\boldsymbol{x_1}, \boldsymbol{y_1}), (\boldsymbol{x_2}, \boldsymbol{y_2}), ..., (\boldsymbol{x_n}, \boldsymbol{y_n})\}$ the training set of n documents. Each document is denoted by a vector $\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, ..., x_{i,d}]$ of d dimensions. The labeling of the i-th document is denoted by vector $\boldsymbol{y}_i = [y_{i,1}, y_{i,2}, ..., y_{i,m}]$, where y_{il} is 1 when the i-th document has the l-th label and 0 otherwise. The goal is to learn a function $\boldsymbol{f} : \mathcal{X} \rightarrow \mathcal{Y}$. Generally, we can assume $\mathbf{f}$ consists of m functions, one for a label.

$$\boldsymbol{f} = [f_1, f_2, ..., f_m]$$

Figure 1: A newspaper article concerning global warming can be classified into two categories, *Environment*, and *Science*.

The labels in the MLC are correlated. For example, a "politics" document is likely to be an "economic" document simultaneously, but likely not to be a "literature" document. According to the latent variable model (Tabachnick et al., 2001), the labels with correlation result in some hidden components, which represent the "share" variance of correlated labels. Intuitively, if we can capture and utilize these hidden components in MTC, the performance will be improved. To implement this idea, we propose a multi- label text categorization method with hidden components, which employ PCA to capture the hidden components, and then incorporates these hidden components into a joint learning framework. Experiments with various data sets and evaluation metrics validate the values of our method. The research close to our work is ML-LOC (Multi-Label learning using LOcal Correlation) in (Huang and Zhou, 2012). The differ-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1816–1821,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

ences between ours and ML-LOC is that ML-LOC employs the cluster method to gain the local correlation, but we employ the PCA to obtain the hidden code. Meanwhile, ML-LOC uses the linear programming in learning the local code, but we employ the gradient descent method since we add non-linear function to the hidden code.

The rest of this paper is organized as follows. Section 2 presents the proposed method. We conduct experiments to demonstrate the effectiveness of the proposed method in section 3. Section 4 concludes this paper.

2 Methodology

2.1 Capturing Hidden Component via Principle Component Analysis

The first step of the proposed method is to capture hidden components of training instances. Here we employ Principal component analysis (PCA). This is because PCA is a well-known statistical tool that converts a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principle components. These principle components represent the inner structure of the correlated variables.

In this paper, we directly employ PCA to convert labels of training instances into their principle components, and take these principle components as hidden components of training instances. We denote by $\boldsymbol{h}_i$ the hidden components of the i-th instance captured by PCA.

2.2 Joint Learning Framework

We expand the original feature representation of the instance $\boldsymbol{x}_i$ by its hidden component code vector $\boldsymbol{c}_i$. For simplicity, we use logistic regression as the motivating example. Let $\boldsymbol{w}_l$ denote weights in the l-th function f_l, consisting of two parts: 1) $\boldsymbol{w}_l^x$ is the part involving the instance features. 2) $\boldsymbol{w}_l^c$ is the part involving the hidden component codes. Hence f_l is:

$$f_l(\boldsymbol{x}, \boldsymbol{c}) = \frac{1}{1 + \exp(-\boldsymbol{x}^T \boldsymbol{w}_l^x - \boldsymbol{c}^T \boldsymbol{w}_l^c)} \tag{1}$$

where $\boldsymbol{C}$ is the code vectors set of all training instances.

The natural choice of the code vector $\boldsymbol{c}$ is $\boldsymbol{h}$. However, when testing an instance, the labeling is unknown (exactly what we try to predict), consequently we can not capture $\boldsymbol{h}$ with PCA to replace the code vector $\boldsymbol{c}$ in the prediction function Eq.(1). Therefore, we assume a linear transformation $\boldsymbol{M}$ from the training instances to their independent components, and use $\boldsymbol{Mx}$ as the approximate independent component. For numerical stability, we add a non-linear function (e.g., the tanh function) to $\boldsymbol{Mx}$. This is formulated as follows.

$$\boldsymbol{c} = tanh(\boldsymbol{Mx}) \tag{2}$$

Aiming to the discrimination fitting and the independent components encoding, we optimize the following optimization problem.

$$\min_{\boldsymbol{W}, C} \sum_{i=1}^{n} \sum_{l=1}^{m} \ell(\boldsymbol{x}_i, \boldsymbol{c}_i, y_{il}, f_l) + \lambda_1 \Omega(\boldsymbol{f}) + \lambda_2 Z(\boldsymbol{C}) \tag{3}$$

The first term of Eq.(3) is the loss function. ℓ is the loss function defined on the training data, and $\boldsymbol{W}$ denotes all weights in the our model, i.e., $\boldsymbol{w}_1, ..., \boldsymbol{w}_l, ..., \boldsymbol{w}_m$. Since we utilize the logistic regression in our model, the loss function is defined as follows.

$$\begin{aligned} & \ell(\boldsymbol{x}, \boldsymbol{c}, y, f) \\ = \; & -y ln f(\boldsymbol{x}, \boldsymbol{c}) - (1 - y) ln(1 - f(\boldsymbol{x}, \boldsymbol{c})) \end{aligned} \tag{4}$$

The second term of Eq.(3) Ω is to punish the model complexity, which we use the ℓ_2 regularization term.

$$\Omega(\boldsymbol{f}) = \sum_{l=1}^{m} ||\boldsymbol{w}_l||^2. \tag{5}$$

The third term of Eq.(3) Z is to enforce the code vector close to the independent component vector. To obtain the goal, we use the least square error between the code vector and the independent component vector as the third regularized term.

$$Z(C) = \sum_{i=1}^{n} ||\boldsymbol{c}_i - \boldsymbol{h}_i||^2. \tag{6}$$

By substituting the Eq.(5) and Eq.(6) into Eq.(3) and changing $\boldsymbol{c}$ to $tanh(\boldsymbol{Mx})$ (Eq.(2)), we obtain the following optimization problem.

$$\begin{aligned} & \min_{\boldsymbol{W}, \boldsymbol{M}} \sum_{i=1}^{n} \sum_{l=1}^{m} \ell(\boldsymbol{x}_i, tanh(\boldsymbol{Mx}_i), y_{il}, \boldsymbol{f}) \\ & + \lambda_1 \sum_{l=1}^{m} ||\boldsymbol{w}_l||^2 + \lambda_2 \sum_{i=1}^{n} ||\boldsymbol{Mx}_i - \boldsymbol{h_i}||^2 \end{aligned} \tag{7}$$

2.3 Alternative Optimization method

We solve the optimization problem in Eq.(7) by the alternative optimization method, which optimize one group of the two parameters with the other fixed. When the $\boldsymbol{M}$ fixed, the third term of Eq.(7) is a constant and thus can be ignored, then Eq.(7) can be rewritten as follows.

$$\min_{\boldsymbol{W}} \sum_{i=1}^{n}\sum_{l=1}^{m} \ell(\boldsymbol{x}_i, tanh(\boldsymbol{M}\boldsymbol{x}_i), y_{il}, f_l) + \lambda_1 \sum_{l=1}^{m} ||\boldsymbol{w}_l||^2 \quad (8)$$

By decomposing Eq.(8) based on the label, the equation Eq.(8) can be simplified to:

$$\min_{\boldsymbol{w}_l} \sum_{i=1}^{n} \ell(\boldsymbol{x}_i, tanh(\boldsymbol{M}\boldsymbol{x}_i), y_{il}, f_l) + \lambda_1 ||\boldsymbol{w}_l||^2 \quad (9)$$

Eq.(9) is the standard logistic regression, which has many efficient optimization algorithms.

When $\boldsymbol{W}$ fixed, the second term is constant and can be omitted, then Ep.(7) can rewritten to Eq.(10). We can apply the gradient descent method to optimize this problem.

$$\min_{\boldsymbol{M}} \sum_{i=1}^{n}\sum_{l=1}^{m} \ell(\boldsymbol{x}_i, tanh(\boldsymbol{M}\boldsymbol{x}_i), y_{il}, f_l) + \lambda_2 \sum_{i=1}^{n} ||\boldsymbol{M}\boldsymbol{x}_i - \boldsymbol{h_i}||^2 \quad (10)$$

3 Experiments

3.1 Evaluation Metrics

Compared with the single-label classification, the multi-label setting introduces the additional degrees of freedom, so that various multi-label evaluation metrics are requisite. We use three different multi-label evaluation metrics, include the hamming loss evaluation metric.

The hamming loss is defined as the percentage of the wrong labels to the total number of labels.

$$Hammingloss = \frac{1}{m}|\mathbf{h}(\boldsymbol{x})\Delta\boldsymbol{y}| \quad (11)$$

where Δ denotes the symmetric difference of two sets, equivalent to XOR operator in Boolean logic. m denotes the label number.

The multi-label 0/1 loss, also known as subset accuracy, is the exact match measure as it requires any predicted set of labels $\mathbf{h}(\boldsymbol{x})$ to match the true set of labels S *exactly*. The 0/1 loss is defined as follows:

$$0/1loss = I(\mathbf{h}(\boldsymbol{x}) \neq \boldsymbol{y}) \quad (12)$$

Let a_j and r_j denote the precision and recall for the j-th label. The macro-averaged F is a harmonic mean between precision and recall, defined as follows:

$$F = \frac{1}{m}\sum_{i=j}^{m} \frac{2 * a_j * r_j}{a_j + r_j} \quad (13)$$

3.2 Datasets

We perform experiments on three MTC data sets: 1) the first data set is slashdot (Read et al., 2011). The slashdot data set is concerned about science and technology news categorization, which predicts multiply labels given article titles and partial blurbs mined from Slashdot.org. 2) the second data set is medical (Pestian et al., 2007). This data set involves the assignment of ICD-9-CM codes to radiology reports. 3) the third data set is tmc2007 (Srivastava and Zane-Ulman, 2005). It is concerned about safety report categorization, which is to label aviation safety reports with respect to what types of problems they describe. The characteristics of them are shown in Table 1, where n denotes the size of the data set, d denotes the dimension of the document instance, and m denotes the number of labels.

dataset	n	d	m	Lcard
slashdot	3782	1079	22	1.18
medical	978	1449	45	1.245
tmc2007	28596	500	22	2.16

Table 1: Multi-label data sets and associated statistics

The measure label cardinality $Lcard$, which is one of the standard measures of "multi-labelness", defined as follows, introduced in (Tsoumakas and Katakis, 2007).

$$Lcard(D) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} y_j^i}{n}$$

where D denotes the data set, l_j^i denotes the j-th label of the i-th instance in the data set.

3.3 Compared to Baselines

To examine the values of the joint learning framework, we compare our method to two baselines. The baseline 1 eliminates the PCA, which just adds an extra set of non-linear features. To implement this baseline, we only need to set $\lambda_2 = 0$. The baseline 2 eliminates the joint learning framework. This baseline captures the hidden component codes with PCA, trains a linear regression model to fit the hidden component codes, and utilizes the outputs of the linear regression model as features.

For the proposed method, we set $\lambda_1 = 0.001$ and $\lambda_2 = 0.1$. For the baseline 2, we employ logistic regression with 0.001 $\ell 2$ regularization as the base classifier. Evaluations are done in ten-fold cross validation. Note that all of them produce real-valued predictions. A threshold t needs to be used to determine the final multi-label set $\boldsymbol{y}$ such that $l_j \in \boldsymbol{y}$ where $p_j \geq t$. We select threshold t, which makes the $Lcard$ measure of predictions for the training set is closest to the $Lcard$ measure of the training set (Read et al., 2011). The threshold t is determined as follows, where D_t is the training set and a multi-label model H_t predicts for the training set under threshold t.

$$t = \underset{t \in [0,1]}{\operatorname{argmin}} |Lcard(D_t) - Lcard(H_t(D_t))| \quad (14)$$

Table 2 reports our method wins over the baselines in terms of different evaluation metrics, which shows the values of PCA and our joint learning framework. The hidden component code only fits the hidden component in the baseline method. The hidden component code obtains balance of fitting hidden component and fitting the training data in the joint learning framework.

3.4 Compared to Other Methods

We compare the proposed method to BR, CC (Read et al., 2011), RAKEL (Tsoumakas and Vlahavas, 2007) and ML-KNN (Zhang and Zhou, 2007). entropy. ML-kNN is an adaption of kNN algorithm for multilabel classification. methods. Binary Revelance (BR) is a simple but effective method that trains binary classifiers for each label independently. BR has a low time complexity but makes an arbitrary assumption that the labels are independent from each other. CC organizes the classifiers along a chain and take predictions produced by the former classifiers as features of the latter classifiers. ML-kNN uses kNN algorithms independently for each label with considering prior probabilities. The Label Powerset (LP) method models independencies among labels by treating each label combination as a new class. LP consumes too much time, since there are 2^m label combinations with m labels. RAndom K labEL (RAKEL) is an ensemble method of LP. RAKEL learns several LP models with random subsets of size k from all labels, and then uses a vote process to determine the final predictions.

For our proposed method, we employ the setup in subsection 3.3. We utilize logistic regression with 0.001 $\ell 2$ regularization as the base classifier for BR, CC and RAKEL. For RAKEL, the number of ensemble is set to the number of label and the size of the label subset is set to 3. For MLKNN, the number of neighbors used in the k-nearest neighbor algorithm is set to 10 and the smooth parameter is set to 1. Evaluations are done in ten-fold cross validation. We employ the threshold-selection strategy introduced in subsection 3.3

Table 2 also reports the detailed results in terms of different evaluation metrics. The mean metric value and the standard deviation of each method are listed for each data set. We see our proposed method shows majorities of wining over the other state-of-the-art methods nearly at all data sets under hamming loss, 0/1 loss and macro f score. Especially, under the macro f score, the advantages of our proposed method over the other methods are very clear.

4 CONCLUSION

Many real-world text categorization applications are multi-label text categorization (MTC), where a documents is usually assigned with *multiple* labels simultaneously. The key challenge of MTC is the label correlations among labels. In this paper, we propose a MTC method via hidden components to capture the label correlations. The proposed method obtains hidden components via PCA and incorporates them into a joint learning framework. Experiments with various data sets and evaluation metrics validate the effectiveness of the proposed method.

Acknowledge

We thank anonymous reviewers for their helpful comments and suggestions. This research was partly supported by National High Tech-

	hamming↓. Lower is better.		
Dataset	slashdot	medical	tmc2007
Proposed	0.044 ± 0.004	0.010 ± 0.002	0.056 ± 0.002
Baseline1	$0.046 \pm 0.003\bullet$	0.010 ± 0.002	0.056 ± 0.001
Baseline2	$0.047 \pm 0.003\bullet$	0.011 ± 0.001	$0.059 \pm 0.001\bullet$
BR	$0.058 \pm 0.003\bullet$	0.010 ± 0.001	$0.060 \pm 0.001\bullet$
CC	$0.049 \pm 0.003\bullet$	0.010 ± 0.001	$0.058 \pm 0.001\bullet$
RAKEL	$0.039 \pm 0.002\circ$	0.011 ± 0.002	0.057 ± 0.001
MLKNN	$0.067 \pm 0.003\bullet$	$0.016 \pm 0.003\bullet$	$0.070 \pm 0.002\bullet$
	0/1 loss↓. Lower is better.		
Dataset	slashdot	medical	tmc2007
Proposed	0.600 ± 0.042	0.316 ± 0.071	0.672 ± 0.010
Baseline1	$0.615 \pm 0.034\bullet$	$0.324 \pm 0.058\bullet$	0.672 ± 0.008
Baseline2	$0.669 \pm 0.039\bullet$	$0.354 \pm 0.062\bullet$	$0.698 \pm 0.007\bullet$
BR	$0.803 \pm 0.018\bullet$	$0.337 \pm 0.063\bullet$	$0.701 \pm 0.008\bullet$
CC	$0.657 \pm 0.025\bullet$	$0.337 \pm 0.064\bullet$	$0.687 \pm 0.010\bullet$
RAKEL	$0.686 \pm 0.024\bullet$	$0.363 \pm 0.064\bullet$	$0.682 \pm 0.009\bullet$
MLKNN	$0.776 \pm 0.020\bullet$	$0.491 \pm 0.083\bullet$	$0.746 \pm 0.003\bullet$
	F score↑. Larger is better.		
Dataset	slashdot	medical	tmc2007
Proposed	0.429 ± 0.026	0.575 ± 0.067	0.587 ± 0.010
Baseline1	$0.413 \pm 0.032\bullet$	$0.547 \pm 0.056\bullet$	0.577 ± 0.011
Baseline2	$0.398 \pm 0.032\bullet$	$0.561 \pm 0.052\bullet$	$0.506 \pm 0.011\bullet$
BR	$0.204 \pm 0.011\bullet$	$0.501 \pm 0.058\bullet$	$0.453 \pm 0.011\bullet$
CC	$0.303 \pm 0.022\bullet$	$0.510 \pm 0.052\bullet$	$0.505 \pm 0.011\bullet$
RAKEL	$0.349 \pm 0.023\bullet$	$0.589 \pm 0.063\circ$	$0.555 \pm 0.011\bullet$
MLKNN	$0.297 \pm 0.031\bullet$	$0.410 \pm 0.064\bullet$	$0.431 \pm 0.014\bullet$

Table 2: Performance (mean±std.) of our method and baseline in terms of different evaluation metrics. $\bullet/\circ$ indicates whether the proposed method is statistically superior/inferior to baseline (pairwise t-test at 5% significance level).

nology Research and Development Program of China (863 Program) (No.2012AA011101), National Natural Science Foundation of China (No.91024009), Major National Social Science Fund of China (No. 12&ZD227). The contact author of this paper, according to the meaning given to this role by Key Laboratory of Computational Linguistics, Ministry of Education, School of Electronics Engineering and Computer Science, Peking University, is Houfeng Wang

References

Sheng-Jun Huang and Zhi-Hua Zhou. 2012. Multi-label learning by exploiting label correlations locally. In *AAAI*.

Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2008. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD*.

Jinseok Nam, Jungi Kim, Iryna Gurevych, and Johannes Fürnkranz. 2013. Large-scale multi-label text classification-revisiting neural networks. *arXiv preprint arXiv:1312.5419*.

John P Pestian, Christopher Brew, Paweł Matykiewicz, DJ Hovermale, Neil Johnson, K Bretonnel Cohen, and Włodzisław Duch. 2007. A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 97–104. Association for Computational Linguistics.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359.

Timothy N Rubin, America Chambers, Padhraic Smyth, and Mark Steyvers. 2012. Statistical topic models for multi-label document classification. *Machine Learning*, 88(1-2):157–208.

Ashok N Srivastava and Brett Zane-Ulman. 2005. Discovering recurring anomalies in text reports regard-

ing complex space systems. In *Aerospace Conference, 2005 IEEE*, pages 3853–3862. IEEE.

Barbara G Tabachnick, Linda S Fidell, et al. 2001. Using multivariate statistics.

Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13.

Grigorios Tsoumakas and Ioannis Vlahavas. 2007. Random k-labelsets: An ensemble method for multilabel classification. *Machine Learning: ECML 2007*, pages 406–417.

Min-Ling Zhang and Zhi-Hua Zhou. 2007. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048.

#TAGSPACE: Semantic Embeddings from Hashtags

Jason Weston
Facebook AI Research
jase@fb.com

Sumit Chopra
Facebook AI Research
spchopra@fb.com

Keith Adams
Facebook AI Research
kma@fb.com

Abstract

We describe a convolutional neural network that learns feature representations for short textual posts using hashtags as a supervised signal. The proposed approach is trained on up to 5.5 billion words predicting 100,000 possible hashtags. As well as strong performance on the hashtag prediction task itself, we show that its learned representation of text (ignoring the hashtag labels) is useful for other tasks as well. To that end, we present results on a document recommendation task, where it also outperforms a number of baselines.

1 Introduction

Hashtags (single tokens often composed of natural language n-grams or abbreviations, prefixed with the character '#') are ubiquitous on social networking services, particularly in short textual documents (a.k.a. posts). Authors use hashtags to diverse ends, many of which can be seen as labels for classical NLP tasks: disambiguation (`chips #futurism` vs. `chips #junkfood`); identification of named entities (`#sf49ers`); sentiment (`#dislike`); and topic annotation (`#yoga`). *Hashtag prediction* is the task of mapping text to its accompanying hashtags. In this work we propose a novel model for hashtag prediction, and show that this task is also a useful surrogate for learning good representations of text.

Latent representations, or *embeddings*, are vectorial representations of words or documents, traditionally learned in an unsupervised manner over large corpora. For example LSA (Deerwester et al., 1990) and its variants, and more recent neural-network inspired methods like those of Bengio et al. (2006), Collobert et al. (2011) and word2vec (Mikolov et al., 2013) learn word embeddings. In the word embedding paradigm, each word is represented as a vector in $\mathbb{R}^n$, where n is a hyperparameter that controls capacity. The embeddings of words comprising a text are combined using a model-dependent, possibly learned function, producing a point in the same embedding space. A similarity measure (for example, inner product) gauges the pairwise relevance of points in the embedding space.

Unsupervised word embedding methods train with a reconstruction objective in which the embeddings are used to predict the original text. For example, word2vec tries to predict all the words in the document, given the embeddings of surrounding words. We argue that hashtag prediction provides a more direct form of supervision: the tags are a labeling by the author of the salient aspects of the text. Hence, predicting them may provide stronger semantic guidance than unsupervised learning alone. The abundance of hashtags in real posts provides a huge labeled dataset for learning potentially sophisticated models.

In this work we develop a convolutional network for large scale ranking tasks, and apply it to hashtag prediction. Our model represents both words and the entire textual post as embeddings as intermediate steps. We show that our method outperforms existing unsupervised (word2vec) and supervised (WSABIE (Weston et al., 2011)) embedding methods, and other baselines, at the hashtag prediction task.

We then probe our model's generality, by transfering its learned representations to the task of *personalized document recommendation*: for each of M users, given N previous positive interactions with documents (likes, clicks, etc.), predict the $N+1$'th document the user will positively interact with. To perform well on this task, the representation should capture the user's interest in textual content. We find representations trained on hashtag prediction outperform representations from unsupervised learning, and that our convolu-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1822–1827,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

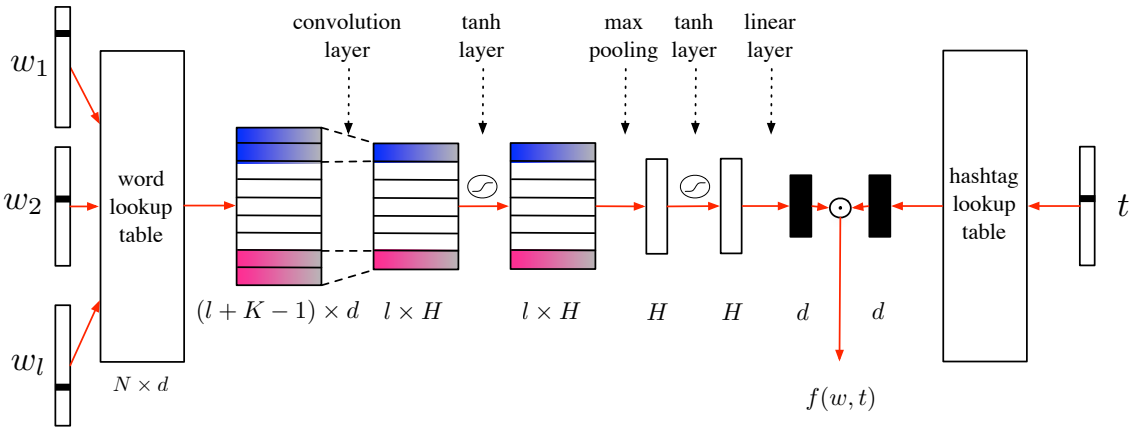


Figure 1: #TAGSPACE convolutional network $f(w,t)$ for scoring a (document, hashtag) pair.

tional architecture performs better than WSABIE trained on the same hashtag task.

2 Prior Work

Some previous work (Davidov et al., 2010; Godin et al., 2013; She and Chen, 2014) has addressed hashtag prediction. Most such work applies to much smaller sets of hashtags than the 100,000 we consider, with the notable exception of Ding et al. (2012), which uses an unsupervised method.

As mentioned in Section 1, many approaches learn unsupervised word embeddings. In our experiments we use word2vec (Mikolov et al., 2013) as a representative scalable model for unsupervised embeddings. WSABIE (Weston et al., 2011) is a supervised embedding approach that has shown promise in NLP tasks (Weston et al., 2013; Hermann et al., 2014). WSABIE is shallow, linear, and ignores word order information, and so may have less modeling power than our approach.

Convolutional neural networks (CNNs), in which shared weights are applied across the input, are popular in the vision domain and have recently been applied to semantic role labeling (Collobert et al., 2011) and parsing (Collobert, 2011). Neural networks in general have also been applied to part-of-speech tagging, chunking, named entity recognition (Collobert et al., 2011; Turian et al., 2010), and sentiment detection (Socher et al., 2013). All these tasks involve predicting a limited (2-30) number of labels. In this work, we make use of CNNs, but apply them to the task of ranking a very large set of tags. We thus propose a model and training scheme that can scale to this class of problem.

3 Convolutional Embedding Model

Our model #TAGSPACE (see Figure 1), like other word embedding models, starts by assigning a d-dimensional vector to each of the l words of an input document $w_1, \ldots, w_l$, resulting in a matrix of size $l \times d$. This is achieved using a matrix of $N \times d$ parameters, termed the lookup-table layer (Collobert et al., 2011), where N is the vocabulary size. In this work N is 10^6, and each row of the matrix represents one of the million most frequent words in the training corpus.

A convolution layer is then applied to the $l \times d$ input matrix, which considers all successive windows of text of size K, sliding over the document from position 1 to l. This requires a further $Kd \times H$ weights and H biases to be learned. To account for words at the two boundaries of the document we also apply a special padding vector at both ends. In our experiments K was set to 5 and H was set to 1000. After the convolutional step, a $\tanh$ nonlinearity followed by a $\max$ operation over the $l \times H$ features extracts a fixed-size (H-dimensional) global feature vector, which is independent of document size. Finally, another $\tanh$ non-linearity followed by a fully connected linear layer of size $H \times d$ is applied to represent the entire document in the original embedding space of d-dimensions.

Hashtags are also represented using d-dimensional embeddings using a lookup-table. We represent the top 100,000 most frequent tags. For a given document w we then rank any given hashtag t using the scoring function:

$$f(w,t) = e_{conv}(w) \cdot e_{lt}(t)$$

where $e_{conv}(w)$ is the embedding of the document by the CNN just described and $e_{lt}(t)$ is the embedding of a candidate tag t. We can thus rank all candidate hashtags via their scores $f(w,t)$, largest first.

To train the above scoring function, and hence the parameters of the model we minimize a ranking loss similar to the one used in WSABIE as a training objective: for each training example, we sample a positive tag, compute $f(w,t^+)$, then sample random tags $\bar{t}$ up to 1000 times until $f(w,\bar{t}) > m + f(w,t^+)$, where m is the margin. A gradient step is then made to optimize the pairwise hinge loss:

$$\mathcal{L} = \max\{0, m - f(w,t^+) + f(w,\bar{t})\}.$$

We use $m = 0.1$ in our experiments. This loss function is referred to as the WARP loss in (Weston et al., 2011) and is used to approximately optimizing the top of the ranked list, useful for metrics like precision and recall@k. In particular, the search for a negative candidate tag means that more energy is spent on improving the ranking performance of positive labels already near the top of the ranked list, compared to only randomly sampling of negatives, which would optimize the average rank instead.

Minimizing our loss is achieved with parallel stochastic gradient descent using the hogwild algorithm (Niu et al., 2011). The lookup-table layers are initialized with the embeddings learned by WSABIE to expedite convergence. This kind of 'pre-training' is a standard trick in the neural network literature, see e.g. (Socher et al., 2011).

The ranking loss makes our model scalable to 100,000 (or more) hashtags. At each training example only a subset of tags have to be computed, so it is far more efficient than a standard classification loss that considers them all.

4 Experiments

4.1 Data

Our experiments use two large corpora of posts containing hashtags from a popular social network.[1] The first corpus, which we call *people*, consists of 201 million posts from individual user accounts, comprising 5.5 billion words.

The second corpus, which we call *pages*, consists of 35.3 million page posts, comprising 1.6 billion words. These posts' authorial voice is a public entity, such as a business, celebrity, brand, or product. The posts in the pages dataset are presumably intended for a wider, more general audience than the posts in the people dataset. Both are summarized in Table 1.

Dataset	Posts (millions)	Words (billions)	Top 4 tags
Pages	35.3	1.6	`#fitness`, `#beauty`, `#luxury`, `#cars`
People	201	5.5	`#FacebookIs10`, `#love`, `#tbt`, `#happy`

Table 1: Datasets used in hashtag prediction.

Both corpora comprise posts between February 1st and February 17th, 2014. Since we are not attempting a multi-language model, we use a simple trigram-based language prediction model to consider only posts whose most likely language is English.

The two datasets use hashtags very differently. The pages dataset has a fatter head, with popular tags covering more examples. The people dataset uses obscure tags more heavily. For example, the top 100 tags account for 33.9% of page tags, but only 13.1% of people tags.

4.2 Hashtag prediction

The hashtag prediction task attempts to rank a post's ground-truth hashtags higher than hashtags it does not contain. We trained models on both the people and page datasets, and collected precision at 1, recall at 10, and mean rank for 50,000 randomly selected posts withheld from training. A further 50,000 withheld posts are used for selecting hyperparameters. We compare #TAGSPACE with the following models:

Frequency This simple baseline ignores input text, always ranking hashtags by their frequency in the training data.

#words This baseline assigns each tag a static score based on its frequency plus a large bonus if it corresponds to a word in the input text. For example, on input "crazy commute this am", #words ranks `#crazy`, `#commute`, `#this` and `#am` highest, in frequency order.

Word2vec We trained the unsupervised model of Mikolov et al. (2013) on both datasets, treating hashtags the same as all other words. To ap-

[1] Both corpora were de-identified during collection.

Crazy commute this am, was lucky to even get in to work.	#nyc, #snow, #puremichigan, #snowday, #snowstorm, #tubestrike, #blizzard, #commute, #snowpocalypse, #chiberia
This can't go on anymore, we need marriage equality now!	#samelove, #equalrights, #equality, #equalityforall, #loveislove, #lgbt, #marriageequality, #noh8, #gayrights, #gaymarriage
Kevin spacey what a super hottie :)	#houseofcards, #hoc, #houseofcardsseason2, #season2, #kevinspacey, #frankunderwood, #netflix, #suits, #swoon, #hubbahubba
Went shopping today and found a really good place to get fresh mango.	#mango, #shopping, #heaven, #100happydays, #yummy, #lunch, #retailtherapy, #yum, #cravings, #wholefoods
Went running today -- my feet hurt so much!	#running, #ouch, #pain, #nopainnogain, #nike #marathontraining, #sore, #outofshape, #nikeplus, #runnerproblems
Wow, what a goal that was, just too fast, Mesut Ozil is the best!	#arsenal, #coyg, #ozil, #afc, #arsenalfc #lfc, #ynwa, #mesut, #gunners, #ucl
Working really hard on the paper all last night.	#thestruggle, #smh, #lol, #collegelife, #homework #sad, #wtf, #confused, #stressed, #work
The restaurant was too expensive and the service was slow.	#ripoff, #firstworldproblems, #smh, #fail, #justsaying #restaurant, #badservice, #food, #middleclassproblems, #neveragain
The restaurant had great food and was reasonably priced.	#dinner, #restaurant, #yum, #food, #delicious #stuffed, #goodtimes, #foodporn, #yummy, #winning
He has the longest whiskers, omg so sweet!	#cat, #kitty, #meow, #cats, #catsofinstagram #crazycatlady, #cute, #kitten, #catlady, #adorable

Table 2: #TAGSPACE (256 dim) predictions for some example posts.

ply these word embeddings to ranking, we first sum the embeddings of each word in the text (as word2vec does), and then rank hashtags by similarity of their embedding to that of the text.[2]

WSABIE (Weston et al., 2011) is a supervised bilinear embedding model. Each word and tag has an embedding. The words in a text are averaged to produce an embedding of the text, and hashtags are ranked by similarity to the text embedding. That is, the model is of the form:

$$f(w,t) = w^\top U^\top V t$$

where the post w is represented as a bag of words (a sparse vector in $\mathbb{R}^N$), the tag is a one-hot-vector in $\mathbb{R}^N$, and U and V are $k \times N$ embedding matrices. The WARP loss, as described in section 3, is used for training.

Performance of all these models at hashtag prediction is summarized in Tables 3 and 4. We find similar results for both datasets. The frequency and #words baselines perform poorly across the board, establishing the need to learn from text. Among the learning models, the unsupervised word2vec performs the worst. We believe this is due to it being unsupervised – adding supervision better optimizes the metric we evaluate. #TAGSPACE outperforms WSABIE at all dimensionalities. Due to the relatively large test sets, the results are statistically significant; for example, comparing #TAGSPACE (64 dim) beats Wsabie (64 dim) for the page dataset 56% of the time, and draws 23% of the time in terms of the rank metric, and is statistically significant with a Wilcoxon signed-rank test.

Some example predictions for #TAGSPACE are given for some constructed examples in Table 2. We also show nearest word embeddings to the posts. Training data was collected at the time of the pax winter storm, explaining predictions for the first post, and Kevin Spacey appears in the show "House of Cards,". In all cases the hashtags reveal labels that capture the semantics of the posts, not just syntactic similarity of individual words.

Comparison to Production System We also compare to a proprietary system in production in Facebook for hashtag prediction. It trains a logistic regression model for every hashtag, using a bag of unigrams, bigrams, and trigrams as the

[2]Note that the unsupervised Word2vec embeddings could be used as input to a supervised classifier, which we did not do. For a supervised embedding baseline we instead use WSABIE. WSABIE trains word embeddings U and hashtag embeddings V in a supervised fashion, whereas Word2vec trains them both unsupervised. Adding supervision to Word2vec would effectively do something in-between: U would still be unsupervised, but V would then be supervised.

Method	dim	P@1	R@10	Rank
Freq. baseline	-	1.06%	2.48%	11277
#words baseline	-	0.90%	3.01%	11034
Word2Vec	256	1.21%	2.85%	9973
Word2Vec	512	1.14%	2.93%	8727
WSABIE	64	4.55%	8.80%	6921
WSABIE	128	5.25%	9.33%	6208
WSABIE	256	5.66%	10.34%	5519
WSABIE	512	5.92%	10.74%	5452
#TAGSPACE	64	6.69%	12.42%	3569
#TAGSPACE	128	6.91%	12.57%	3858
#TAGSPACE	256	**7.37%**	**12.58%**	**3820**

Table 3: Hashtag test results for people dataset.

Method	dim	P@1	R@10	Rank
Freq. baseline	-	4.20%	1.59%	11103
#words baseline	-	2.63%	5.05%	10581
Word2Vec	256	4.66%	8.15%	10149
Word2Vec	512	5.26%	9.33%	9800
WSABIE	64	24.45%	29.64%	2619
WSABIE	128	27.47%	32.94%	2325
WSABIE	256	29.76%	35.28%	1992
WSABIE	512	30.90%	36.96%	1184
#TAGSPACE	64	34.08%	38.96%	1184
#TAGSPACE	128	36.27%	41.42%	1165
#TAGSPACE	256	**37.42%**	**43.01%**	**1155**

Table 4: Hashtag test results for pages dataset.

input features. Unlike the other models we consider here, this baseline has been trained using a set of approximately 10 million posts. Engineering constraints prevent measuring mean rank performance. We present it here as a serious effort at solving the same problem from outside the embedding paradigm. On the people dataset this system achieves 3.47% P@1 and 5.33% R@10. On the pages dataset it obtains 5.97% P@1 and 6.30% R@10. It is thus outperformed by our method. However, we note the differences in experimental setting mean this comparison is perhaps not completely fair (different training sets). We expect performance of linear models such as this to be similar to WSABIE as that has been in the case in other datasets (Gupta et al., 2014), but at the cost of more memory usage. Note that models like logistic regression and SVM do not scale well if you have millions of hashtags, which we could handle in our models.

4.3 Personalized document recommendation

To investigate the generality of these learned representations, we apply them to the task of recommending documents to users based on the user's interaction history. The data for this task comprise anonymized day-long interaction histories for a tiny subset of people on a popular social networking service. For each of the 34 thousand people considered, we collected the text of between 5 and 167 posts that she has expressed previous positive interactions with (likes, clicks, etc.). Given the person's trailing $n-1$ posts, we use our models to predict the n'th post by ranking it against 10,000 other unrelated posts, and measuring precison and recall. The score of the n^{th} post is obtained by taking the max similarity over the $n-1$ posts. We use cosine similarity between post embeddings instead of the inner product that was used for hashtag training so that the scores are not unduly influenced by document length. All learned hashtag models were trained on the people dataset. We also consider a TF-IDF weighted bag-of-words baseline (BoW). The results are given in Table 5.

Method	dim	P@1	R@10	R@50
Word2Vec	256	0.75%	1.96%	3.82%
BoW	-	1.36%	4.29%	8.03%
WSABIE	64	0.98%	3.14%	6.65%
WSABIE	128	1.02%	3.30%	6.71%
WSABIE	256	1.01%	2.98%	5.99%
WSABIE	512	1.01%	2.76%	5.19%
#TAGSPACE	64	1.27%	4.56%	9.64%
#TAGSPACE	128	1.48%	4.74%	9.96%
#TAGSPACE	256	**1.66%**	**5.29%**	**10.69%**
WSABIE+ BoW	64	1.61%	4.83%	9.00%
#TAGSPACE+ BoW	64	1.80%	5.90%	11.22%
#TAGSPACE+ BoW	256	**1.92%**	**6.15%**	**11.53%**

Table 5: Document recommendation task results.

Hashtag-based embeddings outperform BoW and unsupervised embeddings across the board, and #TAGSPACE outperforms WSABIE. The best results come from summing the bag-of-words scores with those of #TAGSPACE.

5 Conclusion

#TAGSPACE is a convolutional neural network that learns to rank hashtags with a minimum of task-specific assumptions and engineering. It performs well, beating several baselines and an industrial system engineered for hashtag prediction. The semantics of hashtags cause #TAGSPACE to learn a representation that captures many salient aspects of text. This representation is general enough to port to the task of personalized document recommendation, where it outperforms other well-known representations.

Acknowledgements

We thank Ledell Wu and Jeff Pasternak for their help with datasets and baselines.

References

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

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

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics*, number EPFL-CONF-192374.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 241–249, Stroudsburg, PA, USA. Association for Computational Linguistics.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.

Zhuoye Ding, Qi Zhang, and Xuanjing Huang. 2012. Automatic hashtag recommendation for microblogs using topic-specific translation model. In *COLING (Posters)'12*, pages 265–274.

Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. 2013. Using topic models for twitter hashtag recommendation. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 593–596. International World Wide Web Conferences Steering Committee.

Maya R Gupta, Samy Bengio, and Jason Weston. 2014. Training highly multiclass classifiers. *Journal of Machine Learning Research*, 15:1–48.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *Proceedings of ACL*.

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

Feng Niu, Benjamin Recht, Christopher Ré, and Stephen J Wright. 2011. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in Neural Information Processing Systems*, 24:693–701.

Jieying She and Lei Chen. 2014. Tomoha: Topic model-based hashtag recommendation on twitter. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 371–372. International World Wide Web Conferences Steering Committee.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

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

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2764–2770. AAAI Press.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Joint Decoding of Tree Transduction Models for Sentence Compression

Jin-ge Yao **Xiaojun Wan** **Jianguo Xiao**
Institute of Computer Science and Technology, Peking University, Beijing 100871, China
Key Laboratory of Computational Linguistic (Peking University), MOE, China
{yaojinge, wanxiaojun, xiaojianguo}@pku.edu.cn

Abstract

In this paper, we provide a new method for decoding tree transduction based sentence compression models augmented with language model scores, by jointly decoding two components. In our proposed solution, rich local discriminative features can be easily integrated without increasing computational complexity. Utilizing an unobvious fact that the resulted two components can be independently decoded, we conduct efficient joint decoding based on dual decomposition. Experimental results show that our method outperforms traditional beam search decoding and achieves the state-of-the-art performance.

1 Introduction

Sentence compression is the task of generating a grammatical and shorter summary for a long sentence while preserving its most important information. One specific instantiation is deletion-based compression, namely generating a compression by dropping words. Various approaches have been proposed to challenge the task of deletion-based compression. Earlier pioneering works (Knight and Marcu, 2000) considered several insightful approaches, including noisy-channel based generative models and discriminative decision tree models. Structured discriminative compression models (McDonald, 2006) are capable of integrating rich features and have been proved effective for this task. Another powerful paradigm for sentence compression should be mentioned here is constraints-based compression,including integer linear programming solutions (Clarke and Lapata, 2008) and first-order Markov logic networks (Huang et al., 2012; Yoshikawa et al., 2012).

A notable class of methods that explicitly deal with syntactic structures are tree transduction models (Cohn and Lapata, 2007; Cohn and Lapata, 2009). In such models a synchronous grammar is extracted from a corpus of parallel syntax trees with leaves aligned. Compressions are generated from the grammar with learned weights. Previous works have noticed that local coherence is usually needed by introducing ngram language model scores, which will make accurate decoding intractable. Traditional approaches conduct beam search to find approximate solutions (Cohn and Lapata, 2009).

In this paper we propose a joint decoding strategy to challenge this decoding task. We address the problem as jointly decoding a simple tree transduction model that only considers rule weights and an ngram compression model. Although either part can be independently solved by dynamic programming, the naive way to integrate two groups of partial scores into a huge dynamic programming chart table is computationally impractical. We provide an effective dual decomposition solution that utilizes the efficient decoding of both parts. By integrating rich structured features that cannot be efficiently involved in normal formulation, results get significantly improved.

2 Motivation

Under the tree transduction models, the sentence compression task is formulated as learning a mapping from an input source syntax tree to a target tree with reduced number of leaves. This mapping is known as a synchronous grammar. The synchronous grammar discussed through out this paper will be synchronous tree substitution grammar (STSG), as in previous studies.

In such formulations, sentence compression is finding the best derivation from a syntax tree that produces a simpler target tree, under the current definition of grammar and learned parameters. Each derivation is attached with a score. For the sake of efficient decoding, the score often decom-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1828–1833,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

poses with rules involved in the derivation. A typical score definition for a derivation $\mathbf{y}$ of source tree $\mathbf{x}$ is in such form (Cohn and Lapata, 2008; Cohn and Lapata, 2009):

$$S(\mathbf{x},\mathbf{y}) = \sum_{r \in \mathbf{y}} \mathbf{w}^T \phi_r(\mathbf{x}) + \log P(ngram(\mathbf{y})) \quad (1)$$

The first term is a weighted sum of features $\phi_r(\mathbf{x})$ defined on each rule r. It is plausible to introduce local scores from ngram models. The second term in the above score definition is added with such purpose.

Cohn and Lapata (2009) explained that exact decoding of Equation 1 is intractable. They proposed a beam search decoding strategy coupled with cube-pruning heuristic (Chiang, 2007), which can further improve decoding efficiency at the cost of largely losing exactness in log probability calculations. For efficiency reasons, rich local ngram features have not been introduced as well.

3 Components of Joint Decoding

The score in Equation 1 consists of two parts: sum of weighted rule features and local ngram scores retrieved from a language model. There is an implicit fact that either part can be used alone with slight modifications to generate a coarse candidate compression. Therefore, we can build a joint decoding system that consists of these two independently decodable components.

In this section we will refer to these two independent models as the pure tree transduction model and the pure ngram compression model, described in Section 3.1 and Section 3.2 respectively. There is a direct generalization of the ngram model by introducing rich local features, which results in the structured discriminative models (Section 3.3).

3.1 Pure Tree Transduction model

By merely considering scores from tree transduction rules, i.e. the first part of Equation 1, we can have our scores factorized with rules. Then finding the best derivation from a STSG grammar can be easily solved by a dynamic programming process described by Cohn and Lapata (2007).

This simplified pure tree transduction model can still produce decent compressions if the rule weights are properly learned during training.

3.2 Pure Ngram based Compression

The pure ngram based model will try to find the most locally smooth compression, reflected by having the maximum log probability score of ngrams.

To avoid the trivial solution of deleting all words, we find the target compression with specified length by dynamic programming.

Furthermore, we can integrate features other than log probabilities. This is equivalent to using a structured discriminative model with rich features on ngrams of candidate compressions.

3.3 Structured Discriminative Model

The structured discriminative model proposed by McDonald (2006) defines rich features on bigrams of possible compressions. The score is defined as weighted linear combination of those features:

$$f(\mathbf{x},\mathbf{z}) = \sum_{j=2}^{|\mathbf{z}|} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, L(\mathbf{z}_{j-1}), L(\mathbf{z}_j)) \quad (2)$$

where the function $L(\mathbf{z}_k)$ maps a token $\mathbf{z}_k$ in compression $\mathbf{z}$ back to the index of the original sentence $\mathbf{x}$. Decoding can still be efficiently done by dynamic programming.

With rich local structural information, the structured discriminative model can play a complementary role to the tree transduction model that focus more on global syntactic structures.

4 Joint Decoding

From now on the remaining issue is jointly decoding the components. Either part factorizes over local structures: rules for the tree transduction model and ngrams for the language model or structured discriminative model. We may build a large dynamic programming table to utilize this kind of locality. Unfortunately this is computationally impractical. It is mathematically equivalent to perform exact dynamic programming decoding of Equation 1, which would consume asymptotically $O(SRL^{2(n-1)V})$ [1] time for building the chart (Cohn and Lapata, 2009). Cohn and Lapata (2009) proposed a beam search approximation along with cube-pruning heuristics to reduce the time complexity down to $O(SRBV)$ [2].

[1] S, R, L and V denote respectively for the number of source tree nodes, the number of rules, size of target lexicon and number of variables involved in each rule.

[2] B denotes the beam width.

In this work we utilize the efficiency of independent decoding from the two components respectively and then combine their solutions according to certain standards. This naturally results in a dual decomposition (Rush et al., 2010) solution.

Dual decomposition has been applied in several natural language processing tasks, including dependency parsing (Koo et al., 2010), machine translation (Chang and Collins, 2011; Rush and Collins, 2011) and information extraction (Reichart and Barzilay, 2012). However, the strength of this inference strategy has seldom been noticed in researches on language generation tasks.

We briefly describe the formulation here.

4.1 Description

We denote the pure tree transduction part and the pure ngram part as $g(\mathbf{y})$ and $f(\mathbf{z})$ respectively. Then joint decoding is equivalent to solving:

$$\max_{\mathbf{y}\in\mathcal{Y},\mathbf{z}\in\mathcal{Z}} \quad g(\mathbf{y}) + f(\mathbf{z}) \tag{3}$$

$$\text{s.t.} \quad \mathbf{z}_{kt} = \mathbf{y}_{kt},\ \forall k \in \{1,...,n\},\ \forall t \in \{0,1\},$$

where $\mathbf{y}$ denotes a derivation which yields a final compression $\{\mathbf{y}_1,...,\mathbf{y}_m\}$. This derivation comes from a pure tree transduction model. $\mathbf{z}$ denotes the compression composed of $\{\mathbf{z}_1,...,\mathbf{z}_m\}$ from an ngram compression model. Without loss of generality, we consider $\mathbf{y}_k$ and $\mathbf{z}_k$ as indicators that take value 1 if the k's token of original sentence has been preserved in the compression and 0 if it has been deleted. In the constraints of problem 3, $\mathbf{y}_{kt}$ or $\mathbf{z}_{kt}$ denote indicator variables that take value 1 if $\mathbf{y}_k$ or $\mathbf{z}_k = t$ and 0 otherwise.

Let $L(u,\mathbf{y},\mathbf{z})$ be the Lagrangian of (3). Then the dual objective naturally factorizes into two parts that can be evaluated independently:

$$\begin{aligned} L(u) &= \max_{\mathbf{y}\in\mathcal{Y},\mathbf{z}\in\mathcal{Z}} L(u,\mathbf{y},\mathbf{z}) \\ &= \max_{\mathbf{y}\in\mathcal{Y},\mathbf{z}\in\mathcal{Z}} g(\mathbf{y}) + f(\mathbf{z}) + \sum_{k,t} u_{kt}(\mathbf{z}_{kt} - \mathbf{y}_{kt}) \\ &= \max_{\mathbf{y}\in\mathcal{Y}}(g(\mathbf{y}) - \sum_{k,t} u_{kt}\mathbf{y}_{kt}) + \\ &\quad \max_{\mathbf{z}\in\mathcal{Z}}(f(\mathbf{z}) + \sum_{k,t} u_{kt}\mathbf{z}_{kt}) \end{aligned}$$

With this factorization, Algorithm 1 tries to solve the dual problem $\min_u L(u)$ by alternatively decoding each component.

This framework is feasible and plausible in that the two subproblems (line 3 and line 4 in Algorithm 1) can be easily solved with slight modifications on the values of the original dynamic programming chart. Joint decoding of a pure tree transduction model and a structured discriminative model is almost the same.

Algorithm 1 Dual Decomposition Joint Decoding

1: **Initialization**: $u_k^{(0)} = 0,\ \forall k \in \{1,...,n\}$
2: **for** $i = 1$ to MAX_ITER **do**
3: $\quad \mathbf{y}^{(i)} \leftarrow \text{argmax}_{\mathbf{y}\in\mathcal{Y}}(g(\mathbf{y}) - \sum_{k,t} u_{kt}^{(i-1)}\mathbf{y}_{kt})$
4: $\quad \mathbf{z}^{(i)} \leftarrow \text{argmax}_{\mathbf{z}\in\mathcal{Z}}(f(\mathbf{z}) + \sum_{k,t} u_{kt}^{(i-1)}\mathbf{z}_{kt})$
5: $\quad$ **if** $\mathbf{y}_{kt}^{(i)} = \mathbf{z}_{kt}^{(i)}\ \forall k\ \forall t$ **then**
6: $\quad\quad$ **return** $(\mathbf{y}^{(i)}, \mathbf{z}^{(i)})$
7: $\quad$ **else**
8: $\quad\quad u_{kt}^{(i)} \leftarrow u_{kt}^{(i-1)} - \delta_i(\mathbf{z}_{kt}^{(i)} - \mathbf{y}_{kt}^{(i)})$
9: $\quad$ **end if**
10: **end for**

The asymptotic time complexity of Algorithm 1 is $O(k(SRV + L^{2(n-1)}))$, where k denotes the number of iterations. This is a significant reduction of $O(SRL^{2(n-1)V})$ by directly solving the original problem and is also comparable to $O(SRBV)$ of conducting beam search decoding.

We apply a similar heuristic with Rush and Collins (2012) to set the step size $\delta_i = \frac{1}{t+1}$, where $t < i$ is the number of past iterations that increase the dual value. This setting decreases the step size only when the dual value moves towards the wrong direction. We limit the maximum iteration number to 50 and return the best primal solution $\mathbf{y}^{(i)}$ among all previous iterations for cases that do not converge in reasonable time.

5 Experiments

5.1 Baselines

The pure tree transduction model and the discriminative model naturally become part of our baselines for comparison [3]. Besides comparing our methods against the tree-transduction model with ngram scores by beam search decoding, we also compare them against the available previous work from Galanis and Androutsopoulos (2010). This state-of-the-art work adopts a two-stage method to rerank results generated by a discriminative maximum entropy model.

5.2 Data Preparation

We evaluated our methods on two standard corpora [4], refer to as Written and Spoken respectively.

[3] The pure ngram language model should not be considered here as it requires additional length constraints and in general does not produce competitive results at all merely by itself.

[4] Available at http://jamesclarke.net/research/resources

We split the datasets according to Table 1.

Table 1: Dataset partition (number of sentences)

Corpus	Training	Development	Testing
Written	1,014	324	294
Spoken	931	83	254

All tree transduction models require parallel parse trees with aligned leaves. We parsed all sentences with the Stanford Parser [5] and aligned sentence pairs with minimum edit distance heuristic [6]. Syntactic features of the discriminative model were also taken from these parse trees.

For systems involving ngram scores, we trained a trigram language model on the Reuters Corpus (Volume 1) [7] with modified Kneser-Ney smoothing, using the widely used tool SRILM [8].

5.3 Model Training

The training process of a tree transduction model followed similarly to Cohn and Lapata (2007) using structured SVMs (Tsochantaridis et al., 2005). The structured discriminative models were trained according to McDonald (2006).

5.4 Evaluation Metrics

We assessed the compression results by the F1-score of grammatical relations (provided by a dependency parser) of generated compressions against the gold-standard compression (Clarke and Lapata, 2006). All systems were controlled to produce similar compression ratios (CR) for fair comparison. We also reported manual evaluation on a sampled subset of 30 sentences from each dataset. Three unpaid volunteers with self-reported fluency in English were asked to rate every candidate. Ratings are in the form of 1-5 scores for each compression.

6 Results

We report test set performance of the structured discriminative model, the pure tree transduction (T3), Galanis and Androutsopoulos (2010)'s method (G&A2010), tree transduction with language model scores by beam search and the proposed joint decoding solutions.

[5] http://nlp.stanford.edu/software/lex-parser.shtml

[6] Ties were broken by always aligning a token in compression to its last appearance in the original sentence. This may better preserve the alignments of full constituents.

[7] http://trec.nist.gov/data/reuters/reuters.html

[8] http://www-speech.sri.com/projects/srilm/

Table 2 shows the compression ratios and F-measure of grammatical relations in average for each dataset. Table 3 presents averaged human rating results for each dataset. We carried out pairwise t-test to examine the statistical significance of the differences [9]. In both datasets joint decoding with dual decomposition solution outperforms other systems, especially when structured models involved. We can also find certain improvements of joint modeling with dual decomposition on the original beam search decoding of Equation 1, under very close compression ratios.

Joint decoding of pure tree transduction and discriminative model gives better performance than the joint model of tree transduction and language model. From Table 3 we can see that integrating discriminative model will mostly improve the preservation of important information rather than grammaticality. This is reasonable under the fact that the language model is trained on large scale data and will often preserve local grammatical coherence, while the discriminative model is trained on small but more compression specific corpora.

Table 2: Results of automatic evaluation. (†: sig. diff. from T3+LM(DD); *: sig. diff. from T3+Discr.(DD) for $p < 0.01$)

Written	CR(%)	GR-F1(%)
Discriminative	70.3	$52.4^{\dagger *}$
G&A2010	71.6	60.2^{*}
Pure Tree-Transduction	72.6	$52.3^{\dagger *}$
T3+LM (Beam Search)	70.4	58.8^{*}
T3+LM (Dual Decomp.)	70.7	60.5
T3+Discr. (Dual Decomp.)	71.0	**62.3**
Gold-Standard	71.4	100.0

Spoken	CR(%)	GR-F1(%)
Discriminative	69.5	$50.6^{\dagger *}$
G&A2010	71.7	59.2^{*}
Pure Tree-Transduction	73.6	$53.8^{\dagger *}$
T3+LM (Beam Search)	75.5	59.5^{*}
T3+LM (Dual Decomp.)	75.3	61.5
T3+Discr. (Dual Decomp.)	74.9	**63.3**
Gold-Standard	72.4	100.0

Table 4 shows some examples of compressed sentences produced by all the systems in comparison. The two groups of outputs are compressions of one sentence from the Written corpora and the Spoken corpora respectively. Ungrammatical compressions can be found very often by several baselines for different reasons, such as the outputs from pure tree transduction and the discriminative model in the first group. The reason behind the

[9] For all multiple comparisons in this paper, significance level was adjusted by the Holm-Bonferroni method.

Table 3: Results of human rating. (†: sig. diff. from T3+LM(DD); *: sig. diff. from T3+Discr.(DD), for $p < 0.01$)

Written	**GR.**	**Imp.**	**CR(%)**
Discriminative	$3.92^{\dagger *}$	$3.46^{\dagger *}$	70.6
G&A2010	$4.11^{\dagger *}$	$3.50^{\dagger *}$	72.4
Pure Tree-Transduction	$3.85^{\dagger *}$	$3.42^{\dagger *}$	70.1
T3+LM (Beam Search)	$4.22^{\dagger *}$	3.69^{*}	73.0
T3+LM (Dual Decomp.)	**4.63**	3.98	73.2
T3+Discr. (Dual Decomp.)	4.62	**4.25**	73.5
Gold-Standard	4.89	4.76	72.9

Spoken	**GR.**	**Imp.**	**CR(%)**
Discriminative	$3.95^{\dagger *}$	$3.62^{\dagger *}$	71.2
G&A2010	$4.09^{\dagger *}$	3.96^{*}	72.5
Pure Tree-Transduction	$3.92^{\dagger *}$	$3.55^{\dagger *}$	71.4
T3+LM (Beam Search)	4.20^{*}	3.78^{*}	75.0
T3+LM (Dual Decomp.)	4.35	4.18	74.5
T3+Discr. (Dual Decomp.)	**4.47**	**4.26**	74.7
Gold-Standard	4.83	4.80	73.1

Table 4: Example outputs

Original: *It was very high for people who took their full-time education beyond the age of 18 , and higher among women than men for all art forms except jazz and art galleries .*
Discr.: *It was high for people took education higher among women .*
(Galanis and Androutsopoulos, 2010): *It was high for people who took their education beyond the age of 18 , and higher among women .*
Pure T3: *It was very high for people who took .*
T3+LM-BeamSearch: *It was very high for people who took their education beyond the age of 18 , and higher among women than men .*
T3+LM-DualDecomp: *It was very high for people who took their education beyond the age of 18 , and higher among women than men .*
T3+Discr.: *It was high for people who took education beyond the age of 18 , and higher among women than men .*
Gold-Standard: *It was very high for people who took full-time education beyond 18 , and higher among women for all except jazz and galleries .*

Original: *But they are still continuing to search the area to try and see if there were , in fact , any further shooting incidents .*
Discr.: *they are continuing to search the area to try and see if there were , further shooting incidents .*
(Galanis and Androutsopoulos, 2010): *But they are still continuing to search the area to try and see if there were , in fact , any further shooting incidents .*
Pure T3: *they are continuing to search the area to try and see if there were any further shooting incidents .*
T3+LM-BeamSearch: *But they are continuing to search the area to try and see if there were , in fact , any further shooting incidents .*
T3+LM-DualDecomp: *But they are continuing to search the area to try and see if there were any further shooting incidents .*
T3+Discr.: *they are continuing to search the area to try and see if there were further shooting incidents .*
Gold-Standard: *they are continuing to search the area to see if there were any further incidents .*

under generation of pure tree transduction is that it mainly deals with global syntactic integrity merely in terms of the application of synchronous rules. Introducing language model scores will smooth the candidate compressions and avoid many aggressive decisions of tree transduction. Discriminative models are good at local decisions with poor consideration of grammaticality. We can see that the joint models have collected their predictive power together. Unfortunately we can still observe some redundancy from our outputs in the examples. The size of training corpus is not large enough to provide enough lexicalized information.

On the other hand, the time consumption of the joint model with dual decomposition decoding in our experiments matched the aforementioned asymptotic analysis. The training process based on new decoding method consumes similar time as beam search with cube-pruning heuristic.

7 Conclusion and Future Work

In this paper we propose a joint decoding scheme for tree transduction based sentence compression. Experimental results suggest that the proposed framework works well. The overall performance gets further improved under our framework by introducing the structured discriminative model.

As several recent efforts have focused on extracting large-scale parallel corpus for sentence compression (Filippova and Altun, 2013), we would like to study how larger corpora can affect tree transduction and our joint decoding solution. Meanwhile, We would like to explore on how other text-rewriting problems can be formulated as a joint model and be applicable to similar strategies described in this work.

Acknowledgements

This work was supported by National Hi-Tech Research and Development Program (863 Program) of China (2014AA015102, 2012AA011101) and National Natural Science Foundation of China (61170166, 61331011). We also thank the anonymous reviewers for very helpful comments.

The contact author of this paper, according to the meaning given to this role by Peking University, is Xiaojun Wan.

References

Yin-Wen Chang and Michael Collins. 2011. Exact decoding of phrase-based translation models through lagrangian relaxation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 26–37, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 377–384. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:273–381.

Trevor Cohn and Mirella Lapata. 2007. Large margin synchronous generation and its application to sentence compression. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 73–82, Prague, Czech Republic, June. Association for Computational Linguistics.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 137–144. Association for Computational Linguistics.

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

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1481–1491, Seattle, Washington, USA, October. Association for Computational Linguistics.

Dimitrios Galanis and Ion Androutsopoulos. 2010. An extractive supervised two-stage method for sentence compression. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 885–893, Los Angeles, California, June. Association for Computational Linguistics.

Minlie Huang, Xing Shi, Feng Jin, and Xiaoyan Zhu. 2012. Using first-order logic to compress sentences. In *AAAI*.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization-step one: Sentence compression. In *AAAI/IAAI*, pages 703–710.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA, October. Association for Computational Linguistics.

Ryan T McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *EACL*.

Roi Reichart and Regina Barzilay. 2012. Multi-event extraction guided by global constraints. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 70–79, Montréal, Canada, June. Association for Computational Linguistics.

Alexander M. Rush and Michael Collins. 2011. Exact decoding of syntactic translation models through lagrangian relaxation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 72–82, Portland, Oregon, USA, June. Association for Computational Linguistics.

Alexander M Rush and Michael Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*, 45:305–362.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Cambridge, MA, October. Association for Computational Linguistics.

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

Katsumasa Yoshikawa, Tsutomu Hirao, Ryu Iida, and Manabu Okumura. 2012. Sentence compression with semantic role constraints. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 349–353. Association for Computational Linguistics.

Dependency-based Discourse Parser for Single-Document Summarization

Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
{yoshida.y,suzuki.jun,hirao.tsutomu,nagata.masaaki}@lab.ntt.co.jp

Abstract

The current state-of-the-art single-document summarization method generates a summary by solving a Tree Knapsack Problem (TKP), which is the problem of finding the optimal rooted subtree of the dependency-based discourse tree (DEP-DT) of a document. We can obtain a gold DEP-DT by transforming a gold Rhetorical Structure Theory-based discourse tree (RST-DT). However, there is still a large difference between the ROUGE scores of a system with a gold DEP-DT and a system with a DEP-DT obtained from an automatically parsed RST-DT. To improve the ROUGE score, we propose a novel discourse parser that directly generates the DEP-DT. The evaluation results showed that the TKP with our parser outperformed that with the state-of-the-art RST-DT parser, and achieved almost equivalent ROUGE scores to the TKP with the gold DEP-DT.

1 Introduction

Discourse structures of documents are believed to be highly beneficial for generating informative and coherent summaries. Several discourse-based summarization methods have been developed, such as (Marcu, 1998; Daumé III and Marcu, 2002; Hirao et al., 2013; Kikuchi et al., 2014). Moreover, the current best ROUGE score for the summarization benchmark data of the RST-discourse Treebank (Carlson et al., 2002) has been provided by (Hirao et al., 2013), whose method also utilizes discourse trees. Thus, the discourse-based summarization approach is one promising way to obtain high-quality summaries.

One possible weakness of discourse-based summarization techniques is that they rely greatly on the accuracy of the discourse parser they use. For example, the above discourse-based summarization methods utilize discourse trees based on the Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) for their discourse information. Unfortunately, the current state-of-the-art RST parser, as described in (Hernault et al., 2010), is insufficient as an off-the-shelf discourse parser. In fact, there is empirical evidence that the quality (i.e., ROUGE score) of summaries from auto-parsed discourse trees is significantly degraded compared with those generated from gold discourse trees (Marcu, 1998; Hirao et al., 2013).

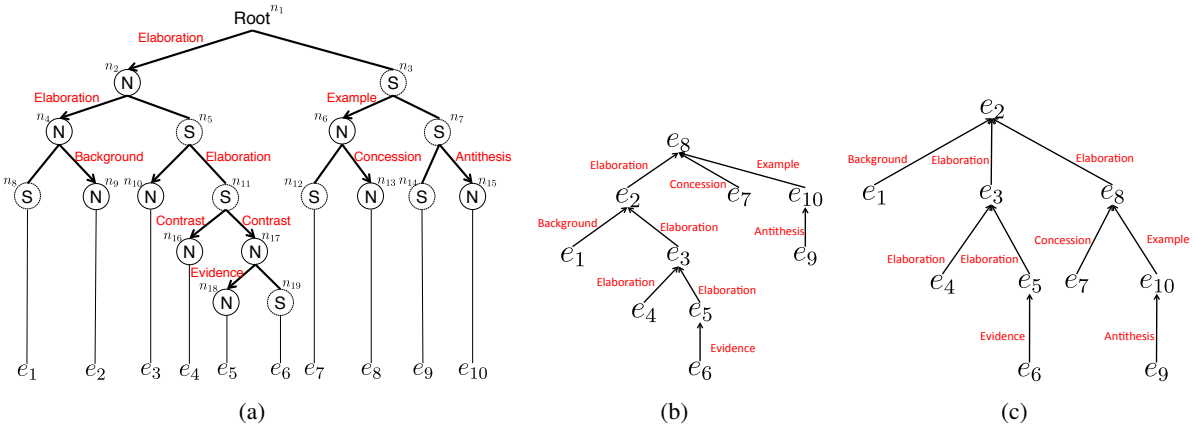
From this background, the goal of this paper is to develop an appropriate discourse parser for discourse-based summarization. We first focus on one of the best discourse-based single document summarization methods as proposed in (Hirao et al., 2013). Their method formulates a single document summarization problem as a Tree Knapsack Problem (TKP) over a dependency-based discourse tree (DEP-DT). In their method, DEP-DTs are automatically transformed from (auto-parsed) RST-discourse trees (RST-DTs) by heuristic rules. Instead, we develop a DEP-DT parser, that directly provides DEP-DTs for their state-of-the-art discourse-based summarization method. We show that summaries generated by our parser improve the ROUGE scores to almost the same level as those generated by gold DEP-DTs. We also investigate the way in which the parsing accuracy helps to improve the ROUGE scores.

2 Single-Document Summarization as a Tree Knapsack Problem

Hirao et al. (2013) formulated single-document summarization as a TKP that is run on the DEP-DT. They obtained a summary by trimming the DEP-DT, i.e. the summary is a rooted subtree of the DEP-DT.

Suppose that we have N EDUs in a document,

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1834–1839,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics



(a) (b) (c)

Figure 1: Examples of RST-DT and DEP-DT. $e_1, \cdots, e_{10}$ are EDUs. (a) Example of an RST-DT from (Marcu, 1998). $n_1, \cdots, n_{19}$ are the non-terminal nodes. (b) Example of the DEP-DT obtained from the incorrect RST-DT that is made by swapping the Nucleus-Satellite relationship of the node n_2 and the node n_3. (c) The correct DEP-DT obtained from the RST-DT in (a).

and the i-th EDU e_i has l_i words. L is the maximum number of words allowed in a summary. In the TKP, if we select e_i, we need to select its parent EDU in the DEP-DT. We denote parent(i) as the index of the parent of e_i in the DEP-DT. $\mathbf{x}$ is an N-dimensional binary vector that represents a summary, i.e. $x_i = 1$ denotes that e_i is included in the summary. The TKP is defined as the following ILP problem:

$$\begin{aligned} \underset{\mathbf{x}}{\text{maximize}} \quad & \textstyle\sum_{i=1}^{N} F(e_i) x_i \\ s.t. \quad & \textstyle\sum_{i=1}^{N} l_i x_i \leq L \\ & \forall i : x_{\text{parent}(i)} \geq x_i \\ & \forall i : x_i \in \{0, 1\}, \end{aligned}$$

where $F(e_i)$ is the score of e_i. We define $F(e_i)$ as follows:

$$F(e_i) = \frac{\sum_{w \in W(e_i)} \text{tf}(w, D)}{\text{Depth}(e_i)},$$

where $W(e_i)$ is the set of words contained in e_i. tf(w, D) is the term frequency of word w in a document D. Depth(e_i) is the depth of e_i in the DEP-DT.

3 Tree Knapsack Problem with Dependency-based Discourse Parser

3.1 Motivation

In (Hirao et al., 2013), they automatically obtain the DEP-DT by transforming from the parsed RST-DT. We simply followed their method for obtaining the DEP-DTs [1]. The transformation algorithm can be found in detail in (Hirao et al., 2013). Figure 1(a) shows an example of the RST-DT. According to RST, a document is represented as a tree whose terminal nodes correspond to elementary discourse units (EDUs) and whose non-terminal nodes indicate the role of the contiguous EDUs, namely, 'nucleus (N)' or 'satellite (S)'. Since a nucleus is more important than a satellite in terms of the writer's purpose, a satellite is always a child of a nucleus in the RST-DT. Some discourse relations between a nucleus and a satellite or two nuclei are defined.

Since the TKP of (Hirao et al., 2013) employs a DEP-DT obtained from an automatically parsed RST-DT, their method strongly relies on the accuracy of the RST parser. For example, in Figure 1(a), if the RST-DT parser incorrectly sets the node n_2 as Satellite and the node n_3 as Nucleus, we obtain an incorrect DEP-DT in Figure 1(b) because the transformation algorithm uses the Nucleus-Satellite relationships in the RST-DT. The dependency relationships in Figure 1(b) are quite different from that of the correct DEP-DT in Figure 1(c). In this example, the parser failed to determine the most salient EDU e_2, that is the root EDU of the gold DEP-DT. Thus, the summary extracted from this DEP-DT will have a low ROUGE score.

The results motivated us to design a new discourse parser fully trained on the DEP-DTs and

[1] Li et al. also defined a similar transformation algorithm (Li et al., 2014). In this paper, we follow the transformation algorithm defined in (Hirao et al., 2013).

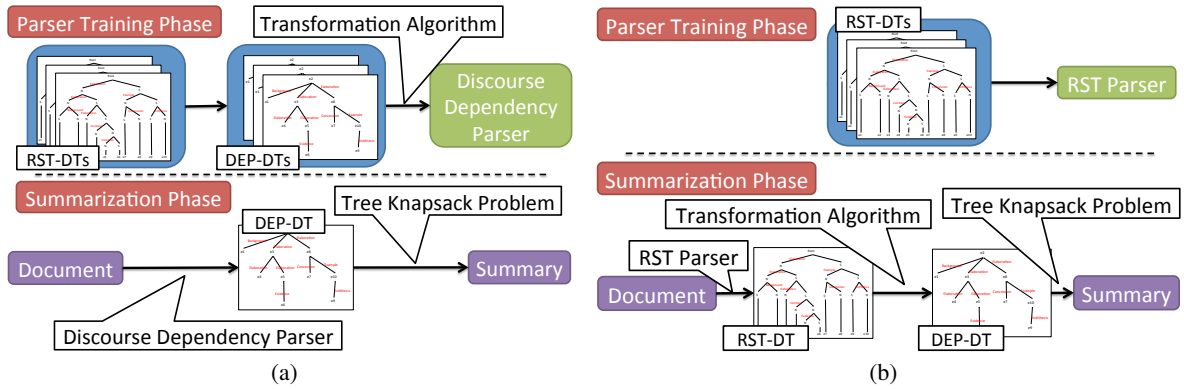


Figure 2: (a) Overview of our proposed method. In the parser training phase, the parser is trained on the DEP-DTs, and in the summarization phase, the document is directly parsed into the DEP-DT. (b) Overview of (Hirao et al., 2013). In the parser training phase, the parser is trained on RST-DTs, and in the summarization phase, the document is parsed into the RST-DT, and then transformed into the DEP-DT.

that could directly generate the DEP-DT. Figure 2(a) shows an overview of the TKP combined with our DEP-DT parser. In the parser training phase, we transform RST-DTs into DEP-DTs, and directly train our parser with the DEP-DTs. In the summarization phase, our method parses a raw document directly into a DEP-DT, and generates a summary with the TKP.

3.2 Description of Discourse Dependency Parser

Our parser is based on the first-order Maximum Spanning Tree (MST) algorithm (McDonald et al., 2005b). Our parser extracts the features from the EDU e_i and the EDU e_j. We use almost the features as those shown in (Hernault et al., 2010). **Lexical N-gram features** use the beginning (or end) lexical N-grams ($N \in \{1, 2, 3\}$) in e_i and e_j. We also include POS tags for the beginning (or end) lexical N-grams ($N \in \{1, 2, 3\}$) in e_i and e_j. **Organizational features** include the distance between e_i and e_j. They also include the number of tokens, and features for identifying whether or not e_i and e_j belong to the same sentence (or paragraph). Soricut et al. (2003) introduced **dominance set features**. They include syntactic labels and the lexical heads of head and attachment nodes along with their dominance relationship. We cannot use the **strong compositionality features** and **rhetorical structure features** described in (Hernault et al., 2010) because we have to know the subtree structures in advance when using these features.

To train the parser, we choose the Margin Infused Relaxed Algorithm (MIRA) (McDonald et al., 2005a; Crammer et al., 2006). We denote $s(\mathbf{w}, \mathbf{y}) = \mathbf{w}^T \mathbf{f}_\mathbf{y}$ as a score function given a weight vector $\mathbf{w}$ and a DEP-DT $\mathbf{y}$. $L(\mathbf{y}, \mathbf{y}^\star)$ is a loss function, and we define it as the number of EDUs that have an incorrect parent EDU in a predicted DEP-DT $\mathbf{y}^\star = \arg\max_{\mathbf{y}} s(\mathbf{w}, \mathbf{y})$. Then, we solve the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{w}} \ ||\mathbf{w} - \mathbf{w}^{(t)}|| \\ &\text{s.t.} \ \ s(\mathbf{w}, \mathbf{y}) - s(\mathbf{w}, \mathbf{y}^\star) \geq L(\mathbf{y}, \mathbf{y}^\star), \end{aligned} \tag{1}$$

where $\mathbf{w}^{(t)}$ is a weight vector in the t-th iteration.

3.3 Redesign of Loss Function for Tree Knapsack Problem

When we make a summary by solving a TKP, we do not necessarily need a DEP-DT where all of the parent-child relationships are correct. This is because we rarely select the EDUs around the leaves in the DEP-DT. On the other hand, the parent-child relationships around the root EDU in the DEP-DT are important because we often select the EDUs around the root EDU. Incorporating these intuitions enables us to develop a DEP-DT parser optimized for the TKP. To incorporate this information, we define the following loss function:

$$L_{\text{Depth}}(\mathbf{y}, \mathbf{y}^\star) = \sum_{(i,r,j) \in \mathbf{y}} \frac{[1 - I(\mathbf{y}^\star, i, j)]}{\text{Depth}(e_i)}, \tag{2}$$

where $I(\mathbf{y}^\star, i, j)$ is an indicator function that equals 1 if EDU e_j is the parent of EDU e_i in the

DEP-DT $\mathbf{y}^\star$ and 0 otherwise. In Section 4, we report results with the original loss function $L(\cdot,\cdot)$ and with the modified loss function $L_{\text{Depth}}(\cdot,\cdot)$.

4 Experimental Evaluation

4.1 Corpus

We used the RST-DT corpus (Carlson et al., 2002) for our experimental evaluations. The corpus consists of 385 Wall Street Journal articles with RST annotation, and 30 of these documents also have one human-made reference summary. We used these 30 documents as the test documents for the summarization evaluation, and used the remaining 355 RST annotated documents as the training data for the parser. Note that we did not use the 30 test documents for the summarization evaluation when we trained the parser.

4.2 Summarization Evaluation

We compared the following three systems that differ in the way they obtain the DEP-DT.

TKP-GOLD Used a DEP-DT converted from a gold RST-DT.

TKP-DIS-DEP Used a DEP-DT automatically parsed by our discourse dependency-based parser (DIS-DEP). Figure 2(a) shows an overview of this system.

TKP-DIS-DEP-LOSS Used a DEP-DT automatically parsed by our discourse dependency-based parser (DIS-DEP). Figure 2(a) shows an overview of this system. It is trained with the loss function defined in equation (2).

TKP-HILDA Used a DEP-DT obtained by transforming a RST-DT parsed by HILDA, a state-of-the-art RST-DT parser (Hernault et al., 2010). Figure 2(b) shows an overview of this system.

Hirao et al. (2013) proved that TKP-HILDA outperformed other methods including Marcu's method (Marcu, 1998), a simple knapsack model, a maximum coverage model and LEAD method that simply takes the first L tokens (L = summary length). Thus, we only employed TKP-HILDA as our baseline.

We follow the evaluation conditions described in (Hirao et al., 2013). The number of tokens in each summary is determined by the number in the human-annotated reference summary. The average length of the reference summaries corresponds to about 10% of the words in the source document. This is also the commonly used evaluation condition for single-document summarization evaluation on the RST-DT corpus. We employed the recall of ROUGE-1, 2 as the evaluation measures.

	ROUGE-1	ROUGE-2
TKP-GOLD	0.321	0.112
TKP-DIS-DEP	0.319	0.109
TKP-DIS-DEP-LOSS	0.323	0.121
TKP-HILDA	0.284	0.093

Table 1: ROUGE Recall scores

Table 1 shows ROUGE scores on the RST-DT corpus. We can see TKP-DIS-DEP and TKP-DIS-DEP-LOSS outperformed TKP-HILDA, and achieved almost the same ROUGE scores as TKP-GOLD. Wilcoxon's signed rank test in terms of ROUGE rejected the null hypothesis, "there is a difference between TKP-HILDA and TKP-DIS-DEP (or TKP-DIS-DEP-LOSS)" (Wilcoxon, 1945). This would be because test documents are relatively small.

We analyzed the differences between the proposed systems (TKP-DIS-DEP and TKP-DIS-DEP-LOSS) and TKP-HILDA. First, we evaluated the overlaps between the EDUs in summaries generated by the system and the EDUs in summaries generated by TKP-GOLD. To see the overlaps, we calculated the average F-value using Recall and Precision defined as follows: Recall $= |S_s \cap S_g|/|S_g|$, Precision $= |S_s \cap S_g|/|S_s|$, where S_s is a set of EDUs in a summary generated by a system, and S_g a set of EDUs in a summary generated by TKP-GOLD. The first line in Table 2 shows the results. TKP-DIS-DEP and TKP-DIS-DEP-LOSS outperformed TKP-HILDA as regards the average F-values. The result revealed that TKP-DIS-DEP and TKP-DIS-DEP-LOSS have more EDUs in common with TKP-GOLD than TKP-HILDA. This result is evidence that TKP-DIS-DEP and TKP-DIS-DEP-LOSS outperformed TKP-HILDA in terms of ROUGE score.

Second, we evaluated the root accuracy (RA), the rate at which a parser can find the root of DEP-DTs. Since the root of a gold DEP-DT is the most salient EDU in a document, it should be included in the summary. The second line in Table 2 shows that our methods succeeded in extracting the root

	TKP-DIS-DEP	TKP-DIS-DEP-LOSS	TKP-HILDA
Avg F-value	$0.532^{\star}$	$0.532^{\star}$	0.415
RA	$0.933^{\star}$	$0.933^{\star}$	0.733
Avg DAS	$0.847^{\star}$	$0.843^{\star}$	0.596

$\star$: significantly better than TKP-HILDA ($p < .05$)

Table 2: Average F-value, Root Accuracy (RA), and average Dependency Accuracy in Summary (DAS). Wilcoxon's signed rank test in terms of average F-value, RA and DAS accepted the null hypothesis.

TKP-GOLD:
Elcotel Inc. expects fiscal second-quarter earnings to trail 1988 results. **Elcotel, a telecommunications company, had net income of $272,000, or five cents a share, in its year-earlier second quarter.** The lower results, Mr. Pierce said. Elcotel will also benefit from moving into other areas. Elcotel has also developed an automatic call processor. Automatic call processors will provide that system for virtually any telephone, Mr. Pierce said, not just phones.

TKP-DIS-DEP, TKP-DIS-DEP-LOSS:
Elcotel Inc. expects fiscal second-quarter earnings to trail 1988 results. Elcotel, a telecommunications company, had net income of $272,000, or five cents a share, in its year-earlier second quarter. George Pierce, chairman and chief executive officer, said in an interview. Although Mr. Pierce expects that line of business to strengthen in the next year. Elcotel will also benefit from moving into other areas. Elcotel has also developed an automatic call processor.

TKP-HILDA:
Elcotel Inc. expects fiscal second-quarter earnings to trail 1988 results. **That several new products will lead to a "much stronger" performance in its second half.** George Pierce, chairman and chief executive officer, said in an interview. Mr. Pierce said Elcotel should realize a minimum of $10 of recurring net earnings for each machine each month. Elcotel has also developed an automatic call processor. Automatic call processors will provide that system for virtually any telephone.

Figure 3: Summaries of wsj_2317. The sentences shown in bold-face are the root EDUs in each DEP-DT of the summary.

of DEP-DT with high accuracy.

Third, to evaluate the coherency of the generated summaries, we compared the average Dependency Accuracy in Summary (DAS), which is defined as follows:

$$\mathrm{DAS}(S) = \frac{1}{|S|} \sum_{e \in S} \delta(e),$$

$$\delta(e) = \begin{cases} 1 & (\text{if parent}(e) \in S) \\ 0 & (\text{otherwise}), \end{cases}$$

where S is a set of EDUs contained in the summary and parent(e) returns the parent EDU of e in the gold DEP-DT. DAS(S) measures the rate of the correct parent-child relationships in S. When DAS equals 1, the summary is a rooted subtree of the gold DEP-DT. The third line in Table 2 shows the results. The results demonstrate that the summaries generated by TKP-DIS-DEP or TKP-DIS-DEP-LOSS tend to preserve the upper level dependency relationships between the EDUs within the gold DEP-DT.

Figure 3 shows summaries of wsj_2317 generated by the three systems. The EDUs corresponding to the root of the DEP-DT are used in each system shown in boldface. We can see that the root EDU in the gold DEP-DT is found in the summaries generated by TKP-DIS-DEP and TKP-DIS-DEP-LOSS, but not in the summary generated by TKP-HILDA.

5 Conclusion

In this paper, we proposed a novel dependency-based discourse parser for single-document summarization. The parser enables us to obtain the DEP-DT without transforming the RST-DT. The evaluation results showed that the TKP with our parser outperformed that with the state-of-the-art RST-DT parser, and achieved almost equivalent ROUGE scores to the TKP with the gold DEP-DT.

References

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2002. Rst discourse treebank, ldc2002t07.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585.

Hal Daumé III and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 449 – 456, Philadelphia, PA, July 6 – 12.

Hugo Hernault, Helmut Prendinger, Mitsuru Ishizuka, et al. 2010. Hilda: a discourse parser using support vector machine classification. *Dialogue and Discourse*, 1(3).

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-document summarization as a tree knapsack problem. In *Proceedings of the 2013 Conference on EMNLP*, pages 1515–1520.

Yuta Kikuchi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2014. Single document summarization based on nested tree structure. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 315–320, Baltimore, Maryland, June. Association for Computational Linguistics.

Sujian Li, Liang Wang, Ziqiang Cao, and Wenjie Li. 2014. Text-level discourse dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25–35, Baltimore, Maryland, June. Association for Computational Linguistics.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Daniel Marcu. 1998. Improving summarization through rhetorical parsing tuning. In *Proc. of The 6th Workshop on VLC*, pages 206–215.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 91–98, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, December.

Improving Word Alignment using Word Similarity

Theerawat Songyot
Dept of Computer Science
University of Southern California
songyot@usc.edu

David Chiang[†]
Dept of Computer Science and Engineering
University of Notre Dame
dchiang@nd.edu

Abstract

We show that semantic relationships can be used to improve word alignment, in addition to the lexical and syntactic features that are typically used. In this paper, we present a method based on a neural network to automatically derive word similarity from monolingual data. We present an extension to word alignment models that exploits word similarity. Our experiments, in both large-scale and resource-limited settings, show improvements in word alignment tasks as well as translation tasks.

1 Introduction

Word alignment is an essential step for learning translation rules in statistical machine translation. The task is to find word-level translation correspondences in parallel text. Formally, given a source sentence $\mathbf{e}$ consisting of words $e_1, e_2, \ldots, e_l$ and a target sentence $\mathbf{f}$ consisting of words $f_1, f_2, \ldots, f_m$, we want to infer an alignment $\mathbf{a}$, a sequence of indices $a_1, a_2, \ldots, a_m$ which indicates, for each target word f_i, the corresponding source word e_{a_i} or a null word. Machine translation systems, including state-of-the-art systems, then use the word-aligned corpus to extract translation rules.

The most widely used methods, the IBM models (Brown et al., 1993) and HMM (Vogel et al., 1996), define a probability distribution $p(\mathbf{f}, \mathbf{a} \mid \mathbf{e})$ that models how each target word f_i is generated from a source word e_{a_i} with respect to an alignment $\mathbf{a}$. The models, however, tend to misalign low-frequency words as they have insufficient training samples. The problem can get worse in low-resource languages. Two branches of research have tried to alleviate the problem. The first branch relies solely on the parallel data; however, additional assumptions about the data are required. This includes, but is not limited to, applying prior distributions (Mermer and Saraçlar, 2011; Vaswani et al., 2012) or smoothing techniques (Zhang and Chiang, 2014). The other branch uses information learned from monolingual data, which is generally easier to acquire than parallel data. Previous work in this branch mostly involves applying syntactic constraints (Yamada and Knight, 2001; Cherry and Lin, 2006; Wang and Zong, 2013) and syntactic features (Toutanova et al., 2002) into the models. The use of syntactic relationships can, however, be limited between historically unrelated language pairs.

Our motivation lies in the fact that a meaningful sentence is not merely a grammatically structured sentence; its semantics can provide insightful information for the task. For example, suppose that the models are uncertain about aligning e to f. If the models are informed that e is semantically related to e', f is semantically related to f', and f' is a translation of e', it should intuitively increase the probability that f is a translation of e. Our work focuses on using such a semantic relationship, in particular, word similarity, to improve word alignments.

In this paper, we propose a method to learn similar words from monolingual data (Section 2) and an extension to word alignment models in which word similarity can be incorporated (Section 3). We demonstrate its application in word alignment and translation (Section 4) and then briefly discuss the novelty of our work in comparison to other methods (Section 5).

2 Learning word similarity

Given a word w, we want to learn a word similarity model $p(w' \mid w)$ of what words w' might be used in place of w. Word similarity can be used to improve word alignment, as in this pa-

[†]Most of the work reported here was performed while the second author was at the University of Southern California.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1840–1845,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

per, but can potentially be useful for other natural language processing tasks as well. Such a model might be obtained from a monolingual thesaurus, in which humans manually provide subjective evaluation for word similarity probabilities, but an automatic method would be preferable. In this section, we present a direct formulation of the word similarity model, which can automatically be trained from monolingual data, and then consider a more practical variant, which we adopt in our experiments.

2.1 Model

Given an arbitrary word type w, we define a word similarity model $p(w' \mid w)$ for all word types w' in the vocabulary V as

$$p(w' \mid w) = \sum_c p(c \mid w)\, p(w' \mid c)$$

where c is a word context represented by a sequence $w_1, w_2, \ldots, w_{2n}$ consisting of n word tokens on the left and n word tokens on the right of w, excluding w. The submodel $p(c \mid w)$ can be a categorical distribution. However, modeling the word context model, $p(w' \mid c)$, as a categorical distribution would cause severe overfitting, because the number of all possible contexts is $|V|^{2n}$, which is exponential in the length of the context. We therefore parameterize it using a feedforward neural network as shown in Figure 1, since the structure has been shown to be effective for language modeling (Bengio et al., 2006; Vaswani et al., 2013). The input to the network is a one-hot representation of each word in c, where the special symbols <s>, </s>, <unk> are reserved for sentence beginning, sentence ending, and words not in the vocabulary. There is an output node for each $w' \in V$, whose activation is $p(w' \mid c)$. Following Bengio et al. (2006), the network uses a shared linear projection matrix to the input embedding layer, which allows information sharing among the context words and also substantially reduces the number of parameters. The input embedding layer has a dimensionality of 150 for each input word. The network uses two hidden layers with 1,000 and 150 rectified linear units, respectively, and a softmax output layer. We arbitrarily use $n = 5$ throughout this paper.

2.2 Training

We extract training data by either collecting or sampling the target words $w \in V$ and their word contexts from monolingual data. The submodel $p(c \mid w)$ can be independently trained easily by maximum likelihood estimation, while the word context model $p(w' \mid c)$ may be difficult to train at scale. We follow previous work (Mnih and Teh, 2012; Vaswani et al., 2013) in adopting noise-contrastive estimation (Gutmann and Hyvärinen, 2010), a fast and simple training algorithm that scales independently of the vocabulary size.

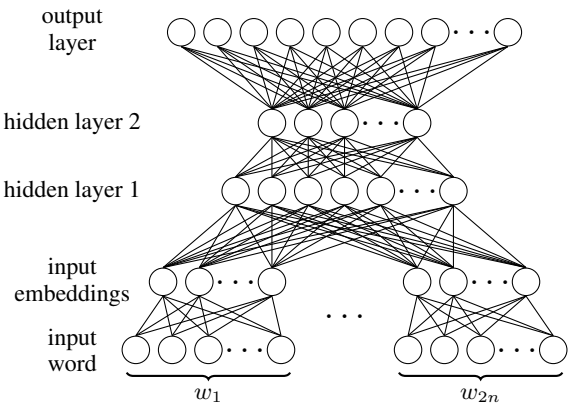


Figure 1: The structure of the word context model

2.3 Model variants

The above formulation of the word similarity model can be interpreted as a mixture model in which w' is similar to w if any of the context probabilities agrees. However, to guard against false positives, we can alternatively reformulate it as a product of experts (Hinton, 1999),

$$p(w' \mid w) = \frac{1}{Z(w)} \exp \sum_c p(c \mid w) \log p(w' \mid c)$$

where $Z(w)$ is a normalization constant. Under this model, w' is similar to w if *all* of the context probabilities agree. Both methods produce reasonably good word similarity; however, in practice, the latter performs better.

Since most of the $p(w' \mid w)$ will be close to zero, for computational efficiency, we can select the k most similar words and renormalize the probabilities. Table 1 shows some examples learned from the 402M-word Xinhua portion of the English Gigaword corpus (LDC2007T07), using a vocabulary V of the 30,000 most frequent words. We set $k = 5$ for illustration purposes.

3 Word alignment model

In this section, we present our word alignment models by extending the standard IBM models.

$p(w' \mid \text{country})$		$p(w' \mid \text{region})$		$p(w' \mid \text{area})$	
country	0.8363	region	0.8338	area	0.8551
region	0.0558	area	0.0760	region	0.0524
nation	0.0522	country	0.0524	zone	0.0338
world	0.0282	province	0.0195	city	0.0326
city	0.0273	city	0.0181	areas	0.0258

Table 1: Examples of word similarity

The method can easily be applied to other related models, for example, the log-linear reparameterization of Model 2 by Dyer et al. (2013). Basically, all the IBM models involve modeling lexical translation probabilities $p(f \mid e)$ which are parameterized as categorical distributions. IBM Model 1, for instance, is defined as

$$p(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) \propto \prod_{i=1}^{m} p(f_i \mid e_{a_i}) = \prod_{i=1}^{m} t(f_i \mid e_{a_i})$$

where each $t(f \mid e)$ denotes the model parameters directly corresponding to $p(f \mid e)$. Models 2–5 and the HMM-based model introduce additional components in order to capture word ordering and word fertility. However, they have $p(f \mid e)$ in common.

3.1 Model

To incorporate word similarity in word alignment models, we redefine the lexical translation probabilities as

$$p(f \mid e) = \sum_{e', f'} p(e' \mid e)\, t(f' \mid e')\, p(f \mid f')$$

for all f, e, including words not in the vocabulary. While the factor $p(e' \mid e)$ can be directly computed by the word similarity model, the factor $p(f \mid f')$ can be problematic because it vanishes for f out of vocabulary. One possible solution would be to use Bayes' rule

$$p(f \mid f') = \frac{p(f' \mid f)\, p(f)}{p(f')}$$

where $p(f' \mid f)$ is computed by the word similarity model. However, we find that this is prone to numerical instability and other complications. In our experiments, we tried the simpler assumption that $p(f \mid f') \approx p(f' \mid f)$, with the rationale that both probabilities are measures of word similarity, which is intuitively a symmetric relation. We also compared the performance of both methods. Table 2 shows that this simple solution works as well as the more exact method of using Bayes' rule. We describe the experiment details in Section 4.

Model	F1	BLEU	
		Test 1	Test 2
Chinese-English			
Bayes' rule	75.7	30.0	27.0
Symmetry assumption	75.3	29.9	27.0
Arabic-English			
Bayes' rule	70.4	37.9	36.7
Symmetry assumption	69.5	38.2	36.8

Table 2: Assuming that word similarity is symmetric, i.e. $p(f \mid f') \approx p(f' \mid f)$, works as well as computing $p(f \mid f')$ using Bayes' rule.

3.2 Re-estimating word similarity

Depending on the quality of word similarity and the distribution of words in the parallel data, applying word similarity directly to the model could lead to an undesirable effect where similar but not interchangeable words rank in the top of the translation probabilities. On the other hand, if we set

$$p(e' \mid e) = \mathbb{1}[e' = e]$$
$$p(f' \mid f) = \mathbb{1}[f' = f]$$

where $\mathbb{1}$ denotes the indicator function, the model reduces to the standard IBM models. To get the best of both worlds, we smooth the two models together so that we rely more on word similarity for rare words and less for frequent words

$$\tilde{p}(w' \mid w) = \frac{\text{count}(w)\mathbb{1}[w' = w] + \alpha\, p(w' \mid w)}{\text{count}(w) + \alpha}$$

This can be thought of as similar to Witten-Bell smoothing, or adding α pseudocounts distributed according to our $p(w' \mid w)$. The hyperparameter α controls how much influence our word similarity model has. We investigated the effect of α by varying this hyperparameter in our word alignment experiments whose details are described in Section 4. Figure 2 shows that performance of the model, as measured by F1 score, is rather insensitive to the choice of α. We used a value of 40 in our experiments.

3.3 Training

Our word alignment models can be trained in the same way as the IBM models using the Expectation Maximization (EM) algorithm to maximize the likelihood of the parallel data. Our extension only introduces an additional time complexity on the order of $\mathcal{O}(k^2)$ on top of the base models, where k is the number of word types used to estimate the full-vocabulary word similarity models.

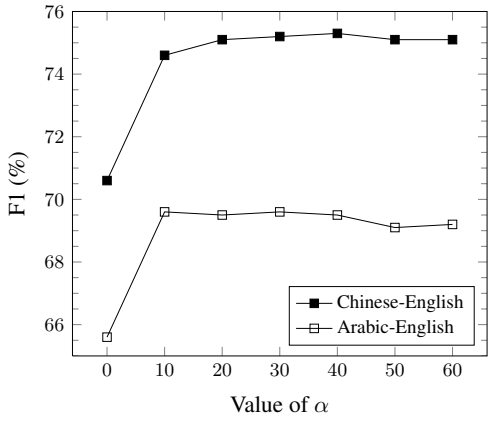


Figure 2: Alignment F1 is fairly insensitive to α over a large range of values

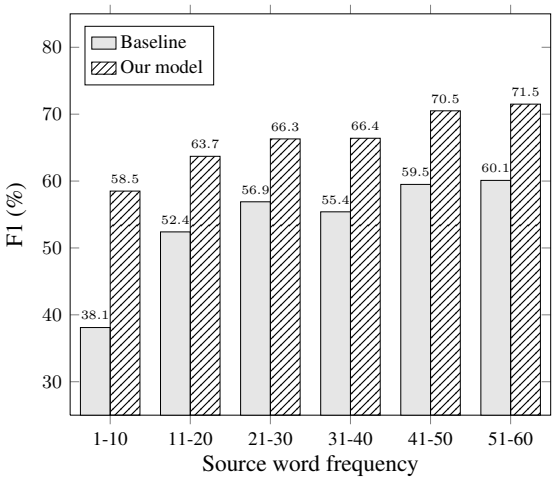


Figure 3: F1 scores for words binned by frequency. Our model gives the largest improvements for the lowest-frequency words.

The larger the value of k is, the closer to the full-vocabulary models our estimations are. In practice, a small value of k seems to be effective since $p(w' \mid w)$ is negligibly small for most w'.

4 Experiments

4.1 Alignment experiments

We conducted word alignment experiments on 2 language pairs: Chinese-English and Arabic-English. For Chinese-English, we used 9.5M+12.3M words of parallel text from the NIST 2009 constrained task[1] and evaluated on 39.6k+50.9k words of hand-aligned data (LDC2010E63, LDC2010E37). For Arabic-English, we used 4.2M+5.4M words of parallel text from the NIST 2009 constrained task[2] and evaluated on 10.7k+15.1k words of hand-aligned data (LDC2006E86). To demonstrate performance under resource-limited settings, we additionally experimented on only the first eighth of the full data, specifically, 1.2M+1.6M words for Chinese-English and 1.0M+1.4M words for Arabic-English. We trained word similarity models on the Xinhua portions of English Gigaword (LDC2007T07), Chinese Gigaword (LDC2007T38), and Arabic Gigaword (LDC2011T1), which are 402M, 323M, and 125M words, respectively. The vocabulary V was the 30,000 most frequent words from each corpus and the $k = 10$ most similar words were used.

We modified GIZA++ (Och and Ney, 2003) to incorporate word similarity. For all experiments, we used the default configuration of GIZA++: 5 iterations each of IBM Model 1, 2, HMM, 3 and 4. We aligned the parallel texts in both forward and backward directions and symmetrized them using grow-diag-final-and (Koehn et al., 2005). We evaluated alignment quality using precision, recall, and F1.

The results in Table 3 suggest that our modeling approach produces better word alignments. We found that our models not only learned smoother translation models for low frequency words but also ranked the conditional probabilities more accurately with respect to the correct translations. To illustrate this, we categorized the alignment links from the Chinese-English low-resource experiment into bins with respect to the English source word frequency and individually evaluated them. As shown in Figure 3, the gain for low frequency words is particularly large.

4.2 Translation experiments

We also ran end-to-end translation experiments. For both languages, we used subsets of the NIST 2004 and 2006 test sets as development data. We used two different data sets as test data: different subsets of the NIST 2004 and 2006 test sets (called Test 1) and the NIST 2008 test sets (called Test 2). We trained a 5-gram language model on the Xinhua portion of English Gigaword (LDC2007T07). We used the Moses toolkit (Koehn et al., 2007) to

[1]Catalog numbers: LDC2003E07, LDC2003E14, LDC2005E83, LDC2005T06, LDC2006E24, LDC2006E34, LDC2006E85, LDC2006E86, LDC2006E92, and LDC2006E93.

[2]Excluding: United Nations proceedings (LDC2004E13), ISI Automatically Extracted Parallel Text (LDC2007E08), and Ummah newswire text (LDC2004T18)

Model	Precision	Recall	F1	BLEU		METEOR	
				Test 1	Test 2	Test 1	Test 2
Chinese-English							
Baseline	65.2	76.9	70.6	29.4	26.7	29.7	28.5
Our model	71.4	79.7	75.3	29.9	27.0	30.0	28.8
Baseline (resource-limited)	56.1	68.1	61.5	23.6	20.3	26.0	24.4
Our model (resource-limited)	66.5	74.4	70.2	24.7	21.6	26.8	25.6
Arabic-English							
Baseline	56.1	79.0	65.6	37.7	36.2	31.1	30.9
Our model	60.0	82.4	69.5	38.2	36.8	31.6	31.4
Baseline (resource-limited)	56.7	76.1	65.0	34.1	33.0	27.9	27.7
Our model (resource-limited)	59.4	80.7	68.4	35.0	33.8	28.7	28.6

Table 3: Experimental results. Our model improves alignments and translations on both language pairs.

build a hierarchical phrase-based translation system (Chiang, 2007) trained using MIRA (Chiang, 2012). Then, we evaluated the translation quality using BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014), and performed significance testing using bootstrap resampling (Koehn, 2004) with 1,000 samples.

Under the resource-limited settings, our methods consistently show 1.1–1.3 BLEU (0.8–1.2 METEOR) improvements on Chinese-English and 0.8–0.9 BLEU (0.8–0.9 METEOR) improvements on Arabic-English, as shown in Table 3. These improvements are statistically significant ($p < 0.01$). On the full data, our method improves Chinese-English translation by 0.3–0.5 BLEU (0.3 METEOR), which is unfortunately not statistically significant, and Arabic-English translation by 0.5–0.6 BLEU (0.5 METEOR), which is statistically significant ($p < 0.01$).

5 Related work

Most previous work on word alignment problems uses morphosyntactic-semantic features, for example, word stems, content words, orthography (De Gispert et al., 2006; Hermjakob, 2009). A variety of log-linear models have been proposed to incorporate these features (Dyer et al., 2011; Berg-Kirkpatrick et al., 2010). These approaches usually require numerical optimization for discriminative training as well as language-specific engineering and may limit their applications to morphologically rich languages.

A more semantic approach resorts to training word alignments on semantic word classes (Ma et al., 2011). However, the resulting alignments are only used to supplement the word alignments learned on lexical words. To our knowledge, our work, which directly incorporates semantic relationships in word alignment models, is novel.

6 Conclusion

We have presented methods to extract word similarity from monolingual data and apply it to word alignment models. Our method can learn similar words and word similarity probabilities, which can be used inside any probability model and in many natural language processing tasks. We have demonstrated its effectiveness in statistical machine translation. The enhanced models can significantly improve alignment quality as well as translation quality.

Acknowledgments

We express our appreciation to Ashish Vaswani for his advice and assistance. We also thank Hui Zhang, Tomer Levinboim, Qing Dou, Aliya Deri for helpful discussions and the anonymous reviewers for their insightful critiques. This research was supported in part by DOI/IBC grant D12AP00225 and a Google Research Award to Chiang.

References

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of HLT NAACL*, pages 582–590.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation:

Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Colin Cherry and Dekang Lin. 2006. Soft syntactic constraints for word alignment through discriminative training. In *Proceedings of COLING/ACL*, pages 105–112.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, 13(1):1159–1187.

Adrià De Gispert, Deepa Gupta, Maja Popović, Patrik Lambert, Jose B. Mariño, Marcello Federico, Hermann Ney, and Rafael Banchs. 2006. Improving statistical word alignments with morpho-syntactic transformations. In *Advances in Natural Language Processing*, pages 368–379. Springer.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Chris Dyer, Jonathan Clark, Alon Lavie, and Noah A. Smith. 2011. Unsupervised word alignment with arbitrary features. In *Proceedings of ACL: HLT*, volume 1, pages 409–419.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM Model 2. In *Proceedings of NAACL-HLT*, pages 644–648.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics (AI-STATS)*, pages 297–304.

Ulf Hermjakob. 2009. Improved word alignment with statistics and linguistic heuristics. In *Proceedings of EMNLP*, volume 1, pages 229–237.

Geoffrey E. Hinton. 1999. Products of experts. In *International Conference on Artificial Neural Networks*, volume 1, pages 1–6.

Philipp Koehn, Amittai Axelrod, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL: Interactive Poster and Demonstration Sessions*, pages 177–180.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*.

Jeff Ma, Spyros Matsoukas, and Richard Schwartz. 2011. Improving low-resource statistical machine translation with a novel semantic word clustering algorithm. In *Proceedings of MT Summit*.

Coşkun Mermer and Murat Saraçlar. 2011. Bayesian word alignment for statistical machine translation. In *Proceedings of ACL: HLT*, volume 2, pages 182–187.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of ICML*.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Kristina Toutanova, H. Tolga Ilhan, and Christopher D. Manning. 2002. Extensions to HMM-based statistical word alignment models. In *Proceedings of EMNLP*, pages 87–94.

Ashish Vaswani, Liang Huang, and David Chiang. 2012. Smaller alignment models for better translations: unsupervised word alignment with the ℓ_0-norm. In *Proceedings of ACL*, volume 1, pages 311–319.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of EMNLP*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of COLING*, volume 2, pages 836–841.

Zhiguo Wang and Chengqing Zong. 2013. Large-scale word alignment using soft dependency cohesion constraints. *Transactions of the Association for Computational Linguistics*, 1(6):291–300.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of ACL*, pages 523–530.

Hui Zhang and David Chiang. 2014. Kneser-Ney smoothing on expected counts. In *Proceedings of ACL*.

Constructing Information Networks Using One Single Model

Qi Li[†] **Heng Ji**[†] **Yu Hong**[†‡] **Sujian Li**[¶]
[†]Computer Science Department, Rensselaer Polytechnic Institute, USA
[‡]School of Computer Science and Technology, Soochow University, China
[¶]Key Laboratory of Computational Linguistics, Peking University, MOE, China
[†]{liq7,hongy2,jih}@rpi.edu, [¶]lisujian@pku.edu.cn

Abstract

In this paper, we propose a new framework that unifies the output of three information extraction (IE) tasks - entity mentions, relations and events as an information network representation, and extracts all of them using one single joint model based on structured prediction. This novel formulation allows different parts of the information network fully interact with each other. For example, many relations can now be considered as the resultant states of events. Our approach achieves substantial improvements over traditional pipelined approaches, and significantly advances state-of-the-art end-to-end event argument extraction.

1 Introduction

Information extraction (IE) aims to discover entity mentions, relations and events from unstructured texts, and these three subtasks are closely interdependent: entity mentions are core components of relations and events, and the extraction of relations and events can help to accurately recognize entity mentions. In addition, the theory of eventualities (Dölling, 2011) suggested that relations can be viewed as states that events start from and result in. Therefore, it is intuitive but challenging to extract all of them simultaneously in a single model. Some recent research attempted to jointly model multiple IE subtasks (e.g., (Roth and Yih, 2007; Riedel and McCallum, 2011; Yang and Cardie, 2013; Riedel et al., 2009; Singh et al., 2013; Li et al., 2013; Li and Ji, 2014)). For example, Roth and Yih (2007) conducted joint inference over entity mentions and relations; Our previous work jointly extracted event triggers and arguments (Li et al., 2013), and entity mentions and relations (Li and Ji, 2014). However, a single model that can extract all of them has never been studied so far.

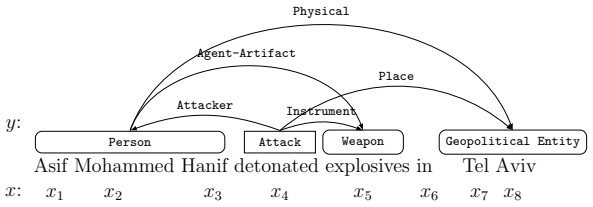


Figure 1: Information Network Representation. Information nodes are denoted by rectangles. Arrows represent information arcs.

For the first time, we uniformly represent the IE output from each sentence as an *information network*, where entity mentions and event triggers are nodes, relations and event-argument links are arcs. We apply a structured perceptron framework with a segment-based beam-search algorithm to construct the *information networks* (Collins, 2002; Li et al., 2013; Li and Ji, 2014). In addition to the perceptron update, we also apply k-best MIRA (McDonald et al., 2005), which refines the perceptron update in three aspects: it is flexible in using various loss functions, it is a large-margin approach, and it can use mulitple candidate structures to tune feature weights.

In an *information network*, we can capture the interactions among multiple nodes by learning joint features during training. In addition to the cross-component dependencies studied in (Li et al., 2013; Li and Ji, 2014), we are able to capture interactions between relations and events. For example, in Figure 1, if we know that the *Person* mention "*Asif Mohammed Hanif*" is an *Attacker* of the *Attack* event triggered by "*detonated*", and the *Weapon* mention "*explosives*" is an *Instrument*, we can infer that there exists an *Agent-Artifact* relation between them. Similarly we can infer the *Physical* relation between "*Asif Mohammed Hanif*" and "*Tel Aviv*".

However, in practice many useful interactions are missing during testing because of the data spar-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1846–1851,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

sity problem of event triggers. We observe that 21.5% of event triggers appear fewer than twice in the ACE'05[1] training data. By using only lexical and syntactic features we are not able to discover the corresponding nodes and their connections. To tackle this problem, we use FrameNet (Baker and Sato, 2003) to generalize event triggers so that semantically similar triggers are clustered in the same frame.

The following sections will elaborate the detailed implementation of our new framework.

2 Approach

We uniformly represent the IE output from each sentence as an *information network* $y = (V, E)$. Each node $v_i \in V$ is represented as a triple $\langle u_i, v_i, t_i \rangle$ of start index u_i, end index v_i, and node type t_i. A node can be an entity mention or an event trigger. A particular type of node is $\perp$ (neither entity mention nor event trigger), whose maximal length is always 1. Similarly, each information arc $e_j \in E$ is represented as $\langle u_j, v_j, r_j \rangle$, where u_j and v_j are the end offsets of the nodes, and r_j is the arc type. For instance, in Figure 1, the event trigger "*detonated*" is represented as $\langle 4, 4, \texttt{Attack} \rangle$, the entity mention "*Asif Mohammed Hanif*" is represented as $\langle 1, 3, \texttt{Person} \rangle$, and their argument arc is $\langle 4, 3, \texttt{Attacker} \rangle$. Our goal is to extract the whole information network y for a given sentence x.

2.1 Decoding Algorithm

Our joint decoding algorithm is based on extending the segment-based algorithm described in our previous work (Li and Ji, 2014). Let $x = (x_1, ..., x_m)$ be the input sentence. The decoder performs two types of actions at each token x_i from left to right:

- NODEACTION(i, j): appends a new node $\langle j, i, t \rangle$ ending at the i-th token, where $i - d_t < j \leq i$, and d_t is the maximal length of type-t nodes in training data.
- ARCACTION(i, j): for each $j < i$, incrementally creates a new arc between the nodes ending at the j-th and i-th tokens respectively: $\langle i, j, r \rangle$.

After each action, the top-k hypotheses are selected according to their features $\mathbf{f}(x, y')$ and weights $\mathbf{w}$:

$$\underset{y' \in buffer}{\text{best}_\text{k}} \ \mathbf{f}(x, y') \cdot \mathbf{w}$$

Since a relation can only occur between a pair of entity mentions, an argument arc can only occur between an entity mention and an event trigger, and each edge must obey certain entity type constraints, during the search we prune invalid ARCACTIONs by checking the types of the nodes ending at the j-th and the i-th tokens. Finally, the top hypothesis in the beam is returned as the final prediction. The upper-bound time complexity of the decoding algorithm is $O(d \cdot b \cdot m^2)$, where d is the maximum size of nodes, b is the beam size, and m is the sentence length. The actual execution time is much shorter, especially when entity type constraints are applied.

2.2 Parameter Estimation

For each training instance (x, y), the structured perceptron algorithm seeks the assignment with the highest model score:

$$z = \underset{y' \in \mathcal{Y}(x)}{\text{argmax}} \, \mathbf{f}(x, y') \cdot \mathbf{w}$$

and then updates the feature weights by using:

$$\mathbf{w}^{new} = \mathbf{w} + \mathbf{f}(x, y) - \mathbf{f}(x, z)$$

We relax the exact inference problem by the aforementioned beam-search procedure. The standard perceptron will cause invalid updates because of inexact search. Therefore we apply early-update (Collins and Roark, 2004), an instance of violation-fixing methods (Huang et al., 2012). In the rest of this paper, we override y and z to denote prefixes of structures.

In addition to the simple perceptron update, we also apply k-best MIRA (McDonald et al., 2005), an online large-margin learning algorithm. During each update, it keeps the norm of the change to feature weights $\mathbf{w}$ as small as possible, and forces the margin between y and the k-best candidate z greater or equal to their loss $L(y, z)$. It is formulated as a quadratic programming problem:

$$\begin{aligned} &\min \|\mathbf{w}^{new} - \mathbf{w}\| \\ &\text{s.t. } \mathbf{w}^{new}\mathbf{f}(x, y) - \mathbf{w}^{new}\mathbf{f}(x, z) \geq L(y, z) \\ &\forall z \in best_k(x, \mathbf{w}) \end{aligned}$$

We employ the following three loss functions for comparison:

[1] http://www.itl.nist.gov/iad/mig//tests/ace

Freq.	Relation Type	Event Type	Arg-1	Arg-2	Example
159	Physical	Transport	Artifact	Destination	He$_{(arg-1)}$ was escorted$_{(trigger)}$ into Iraq$_{(arg-2)}$.
46	Physical	Attack	Target	Place	Many people$_{(arg-1)}$ were in the cafe$_{(arg-2)}$ during the blast$_{(trigger)}$.
42	Agent-Artifact	Attack	Attacker	Instrument	Terrorists$_{(arg-1)}$ might use$_{(trigger)}$ the devices$_{(arg-2)}$ as weapons.
41	Physical	Transport	Artifact	Origin	The truck$_{(arg-1)}$ was carrying$_{(trigger)}$ Syrians fleeing the war in Iraq$_{(arg-2)}$.
33	Physical	Meet	Entity	Place	They$_{(arg-1)}$ have reunited$_{(trigger)}$ with their friends in Norfolk$_{(arg-2)}$.
32	Physical	Die	Victim	Place	Two Marines$_{(arg-1)}$ were killed$_{(trigger)}$ in the fighting in Kut$_{(arg-2)}$.
28	Physical	Attack	Attacker	Place	Protesters$_{(arg-1)}$ have been clashing$_{(trigger)}$ with police in Tehran$_{(arg-2)}$.
26	ORG-Affiliation	End-Position	Person	Entity	NBC$_{(arg-2)}$ is terminating$_{(trigger)}$ freelance reporter Peter Arnett$_{(arg-1)}$.

Table 1: Frequent overlapping relation and event types in the training set.

- The first one is F_1 loss:

$$L_1(y, z) = 1 - \frac{2 \cdot |y \cap z|}{|y| + |z|}$$

When counting the numbers, we treat each node and arc as a single unit. For example, in Figure 1, $|y| = 6$.

- The second one is 0-1 loss:

$$L_2(y, z) = \begin{cases} 1 & y \neq z \\ 0 & y = z \end{cases}$$

It does not discriminate the extent to which z deviates from y.

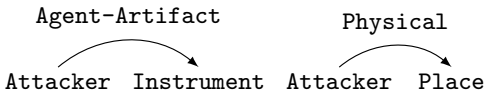
- The third loss function counts the difference between y and z:

$$L_3(y, z) = |y| + |z| - 2 \cdot |y \cap z|$$

Similar to F_1 loss function, it penalizes both missing and false-positive units. The difference is that it is sensitive to the size of y and z.

2.3 Joint Relation-Event Features

By extracting three core IE components in a joint search space, we can utilize joint features over multiple components in addition to factorized features in pipelined approaches. In addition to the features as described in (Li et al., 2013; Li and Ji, 2014), we can make use of joint features between relations and events, given the fact that relations are often ending or starting states of events (Dölling, 2011). Table 1 shows the most frequent overlapping relation and event types in our training data. In each partial structure y' during the search, if both arguments of a relation participate in an event, we compose the corresponding argument roles and relation type as a joint feature for y'. For example, for the structure in Figure 1, we obtain the following joint relation-event features:



Split	Sentences	Mentions	Relations	Triggers	Arguments
Train	7.2k	25.7k	4.8k	2.8k	4.5k
Dev	1.7k	6.3k	1.2k	0.7k	1.1k
Test	1.5k	5.3k	1.1k	0.6k	1.0k

Table 2: Data set

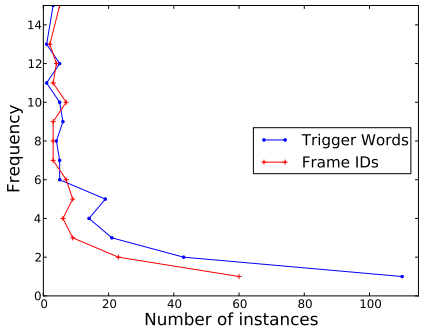


Figure 2: Distribution of triggers and their frames.

2.4 Semantic Frame Features

One major challenge of constructing information networks is the data sparsity problem in extracting event triggers. For instance, in the sentence: "*Others were* **mutilated** *beyond recognition.*" The *Injure* trigger "*mutilated*" does not occur in our training data. But there are some similar words such as "*stab*" and "*smash*". We utilize FrameNet (Baker and Sato, 2003) to solve this problem. FrameNet is a lexical resource for semantic frames. Each frame characterizes a basic type of semantic concept, and contains a number of words (lexical units) that evoke the frame. Many frames are highly related with ACE events. For example, the frame "*Cause_harm*" is closely related with *Injure* event and contains 68 lexical units such as "*stab*", "*smash*" and "*mutilate*".

Figure 2 compares the distributions of trigger words and their frame IDs in the training data. We can clearly see that the trigger word distribution suffers from the long-tail problem, while Frames reduce the number of triggers which occur only

Methods	Entity Mention (%)			Relation (%)			Event Trigger (%)			Event Argument (%)		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Pipelined Baseline	83.6	75.7	79.5	68.5	41.4	51.6	71.2	58.7	64.4	64.8	24.6	35.7
Pipeline + Li et al. (2013)				N/A			74.5	56.9	64.5	67.5	31.6	43.1
Li and Ji (2014)	85.2	76.9	80.8	68.9	41.9	52.1	N/A					
Joint w/ Avg. Perceptron	85.1	77.3	81.0	70.5	41.2	52.0	67.9	62.8	65.3	64.7	35.3	45.6
Joint w/ MIRA w/ F_1 Loss	83.1	75.3	79.0	65.5	39.4	49.2	59.6	63.5	61.5	60.6	38.9	47.4
Joint w/ MIRA w/ 0-1 Loss	84.2	76.1	80.0	65.4	41.8	51.0	65.6	61.0	63.2	60.5	39.6	47.9
Joint w/ MIRA w/ L_3 Loss	85.3	76.5	80.7	70.8	42.1	52.8	70.3	60.9	65.2	66.4	36.1	46.8

Table 3: Overall performance on test set.

once in the training data from 100 to 60 and alleviate the sparsity problem. For each token, we exploit the frames that contain the combination of its lemma and POS tag as features. For the above example, "*Cause_harm*" will be a feature for "*mutilated*". We only consider tokens that appear in at most 2 frames, and omit the frames that occur fewer than 20 times in our training data.

3 Experiments

3.1 Data and Evaluation

We use ACE'05 corpus to evaluate our method with the same data split as in (Li and Ji, 2014). Table 2 summarizes the statistics of the data set. We report the performance of extracting entity mentions, relations, event triggers and arguments separately using the standard F_1 measures as defined in (Ji and Grishman, 2008; Chan and Roth, 2011):

- An entity mention is correct if its entity type (7 in total) and head offsets are correct.
- A relation is correct if its type (6 in total) and the head offsets of its two arguments are correct.
- An event trigger is correct if its event subtype (33 in total) and offsets are correct.
- An argument link is correct if its event subtype, offsets and role match those of any of the reference argument mentions.

In this paper we focus on entity arguments while disregard values and time expressions because they can be most effectively extracted by handcrafted patterns (Chang and Manning, 2012).

3.2 Results

Based on the results of our development set, we trained all models with 21 iterations and chose the beam size to be 8. For the k-best MIRA updates, we set k as 3. Table 3 compares the overall performance of our approaches and baseline methods.

Our joint model with perceptron update outperforms the state-of-the-art pipelined approach in (Li et al., 2013; Li and Ji, 2014), and further improves the joint event extraction system in (Li et al., 2013) ($p < 0.05$ for entity mention extraction, and $p < 0.01$ for other subtasks, according to Wilcoxon Signed RankTest). For the k-best MIRA update, the L_3 loss function achieved better performance than F_1 loss and 0-1 loss on all sub-tasks except event argument extraction. It also significantly outperforms perceptron update on relation extraction and event argument extraction ($p < 0.01$). It is particularly encouraging to see the end output of an IE system (event arguments) has made significant progress (12.2% absolute gain over traditional pipelined approach).

3.3 Discussions

3.3.1 Feature Study

Rank	Feature		Weight
1	Frame=Killing	Die	0.80
2	Frame=Travel	Transport	0.61
3	Physical(Artifact, Destination)		0.60
4	w_1="home"	Transport	0.59
5	Frame=Arriving	Transport	0.54
6	ORG-AFF(Person, Entity)		0.48
7	Lemma=charge	Charge-Indict	0.45
8	Lemma=birth	Be-Born	0.44
9	Physical(Artifact,Origin)		0.44
10	Frame=Cause_harm	Injure	0.43

Table 4: Top Features about Event Triggers.

Table 4 lists the weights of the most significant features about event triggers. The 3rd, 6th, and 9th rows are joint relation-event features. For instance, *Physical(Artifact, Destination)* means the arguments of a *Physical* relation participate in a *Transport* event as *Artifact* and *Destination*. We can see that both the joint relation-event features

and FrameNet based features are of vital importance to event trigger labeling. We tested the impact of each type of features by excluding them in the experiments of "MIRA w/ L_3 loss". We found that FrameNet based features provided 0.8% and 2.2% F_1 gains for event trigger and argument labeling respectively. Joint relation-event features also provided 0.6% F_1 gain for relation extraction.

3.3.2 Remaining Challenges

Event trigger labeling remains a major bottleneck. In addition to the sparsity problem, the remaining errors suggest to incorporate external world knowledge. For example, some words act as triggers for some certain types of events only when they appear together with some particular arguments:

- "*Williams picked up the child again and this time,* **threw**$_{\texttt{Attack}}$ **her** *out the window.*"
  The word "*threw*" is used as an *Attack* event trigger because the *Victim* argument is a "*child*".
- "*Ellison to spend $10.3 billion to* **get**$_{\texttt{Merge_Org}}$ *his* **company**." The common word "*get*" is tagged as a trigger of *Merge_Org*, because its object is "*company*".
- "*We believe that the likelihood of them* **using**$_{\texttt{Attack}}$ *those* **weapons** *goes up.*"
  The word "*using*" is used as an *Attack* event trigger because the *Instrument* argument is "*weapons*".

Another challenge is to distinguish physical and non-physical events. For example, in the sentence:

- "*we are paying great attention to their ability to* **defend**$_{\texttt{Attack}}$ *on the ground.*",

our system fails to extract "*defend*" as an *Attack* trigger. In the training data, "*defend*" appears multiple times, but none of them is tagged as *Attack*. For instance, in the sentence:

- "*North Korea could do everything to* **defend** *itself.* "

"*defend*" is not an *Attack* trigger since it does not relate to physical actions in a war. This challenge calls for deeper understanding of the contexts.

Finally, some pronouns are used to refer to actual events. Event coreference is necessary to recognize them correctly. For example, in the following two sentences from the same document:

- "*It's important that people all over the world know that we don't believe in the* **war**$_{\texttt{Attack}}$.",
- "*Nobody questions whether* **this**$_{\texttt{Attack}}$ *is right or not.*"

"*this*" refers to "*war*" in its preceding contexts. Without event coreference resolution, it is difficult to tag it as an *Attack* event trigger.

4 Conclusions

We presented the first joint model that effectively extracts entity mentions, relations and events based on a unified representation: information networks. Experiment results on ACE'05 corpus demonstrate that our approach outperforms pipelined method, and improves event-argument performance significantly over the state-of-the-art. In addition to the joint relation-event features, we demonstrated positive impact of using FrameNet to handle the sparsity problem in event trigger labeling.

Although our primary focus in this paper is information extraction in the ACE paradigm, we believe that our framework is general to improve other tightly coupled extraction tasks by capturing the inter-dependencies in the joint search space.

Acknowledgments

We thank the three anonymous reviewers for their insightful comments. This work was supported by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), U.S. NSF CAREER Award under Grant IIS-0953149, U.S. DARPA Award No. FA8750-13-2-0041 in the Deep Exploration and Filtering of Text (DEFT) Program, IBM Faculty Award, Google Research Award, Disney Research Award and RPI faculty start-up grant. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Collin F. Baker and Hiroaki Sato. 2003. The framenet data and software. In *Proc. ACL*, pages 161–164.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proc. ACL*, pages 551–560.

Angel X. Chang and Christopher Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In *Proc. LREC*, pages 3735–3740.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. ACL*, pages 111–118.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*, pages 1–8.

Johannes Dölling. 2011. Aspectual coercion and eventuality structure. pages 189–226.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proc. HLT-NAACL*, pages 142–151.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proc. ACL*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proc. ACL*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proc. ACL*, pages 73–82.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. ACL*, pages 91–98.

Sebastian Riedel and Andrew McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *Proc. EMNLP*.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A markov logic approach to bio-molecular event extraction. In *Proc. the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*.

Dan Roth and Wen-tau Yih. 2007. Global inference for entity and relation identification via a lin- ear programming formulation. In *Introduction to Statistical Relational Learning*. MIT.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *Proc. CIKM Workshop on Automated Knowledge Base Construction*.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proc. ACL*, pages 1640–1649.

Event Role Extraction using Domain-Relevant Word Representations

Emanuela Boroş[†‡] **Romaric Besançon**[†] **Olivier Ferret**[†] **Brigitte Grau**[‡*]
[†]CEA, LIST, Vision and Content Engineering Laboratory, F-91191, Gif-sur-Yvette, France
[‡]LIMSI, rue John von Neumann, Campus Universitaire d'Orsay, F-91405 Orsay cedex
[*]ENSIIE, 1 square de la résistance F-91025 Évry cedex
firstname.lastname@cea.fr firstname.lastname@limsi.fr

Abstract

The efficiency of Information Extraction systems is known to be heavily influenced by domain-specific knowledge but the cost of developing such systems is considerably high. In this article, we consider the problem of event extraction and show that learning word representations from unlabeled domain-specific data and using them for representing event roles enable to outperform previous state-of-the-art event extraction models on the MUC-4 data set.

1 Introduction

In the Information Extraction (IE) field, event extraction constitutes a challenging task. An event is described by a set of participants (*i.e.* attributes or roles) whose values are text excerpts. The event extraction task is related to several subtasks: event mention detection, candidate role-filler extraction, relation extraction and event template filling. The problem we address here is the detection of role-filler candidates and their association with specific roles in event templates. For this task, IE systems adopt various ways of extracting patterns or generating rules based on the surrounding context, local context and global context (Patwardhan and Riloff, 2009). Current approaches for learning such patterns include bootstrapping techniques (Huang and Riloff, 2012a; Yangarber et al., 2000), weakly supervised learning algorithms (Huang and Riloff, 2011; Sudo et al., 2003; Surdeanu et al., 2006), fully supervised learning approaches (Chieu et al., 2003; Freitag, 1998; Bunescu and Mooney, 2004; Patwardhan and Riloff, 2009) and other variations. All these methods rely on substantial amounts of manually annotated corpora and use a large body of linguistic knowledge. The performance of these approaches is related to the amount of knowledge engineering deployed and a good choice of features and classifiers. Furthermore, the efficiency of the system relies on the a priori knowledge of the applicative domain (the nature of the events) and it is generally difficult to apply a system on a different domain with less annotated data without reconsidering the design of the features used. An important step forwards is TIER$_{light}$ (Huang and Riloff, 2012a) that targeted the minimization of human supervision with a bootstrapping technique for event roles detection. Also, PIPER (Patwardhan and Riloff, 2007; Patwardhan, 2010) distinguishes between relevant and irrelevant regions and learns domain-relevant extraction patterns using a semantic affinity measure. Another possible approach for dealing with this problem is to combine the use a restricted set of manually annotated data with a much larger set of data extracted in an unsupervised way from a corpus. This approach was experimented for relations in the context of Open Information Extraction (Soderland et al., 2010) but not for extracting events and their participants to our knowledge.

In this paper, we propose to approach the task of labeling text spans with event roles by automatically learning relevant features that requires limited prior knowledge, using a neural model to induce semantic word representations (commonly referred as *word embeddings*) in an unsupervised fashion, as in (Bengio et al., 2006; Collobert and Weston, 2008). We exploit these word embeddings as features for a supervised event role (multiclass) classifier. This type of approach has been proved efficient for numerous tasks in natural language processing, including named entity recognition (Turian et al., 2010), semantic role labeling (Collobert et al., 2011), machine translation (Schwenk and Koehn, 2008; Lambert et al., 2012), word sense disambiguation (Bordes et al., 2012) or sentiment analysis (Glorot et al., 2011; Socher et al., 2011) but has never been used, to our knowl-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1852–1857,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

edge, for an event extraction task. Our goal is two-fold: (1) to prove that using as only features word vector representations makes the approach competitive in the event extraction task; (2) to show that these word representations are scalable and robust when varying the size of the training data. Focusing on the data provided in MUC-4 (Lehnert et al., 1992), we prove the relevance of our approach by outperforming state-of-the-art methods, in the same evaluation environment as in previous works.

2 Approach

In this work, we approach the event extraction task by learning word representations from a domain-specific data set and by using these representations to identify the event roles. This idea relies on the assumption that the different words used for a given event role in the text share some semantic properties, related to their context of use and that these similarities can be captured by specific representations that can be automatically induced from the text, in an unsupervised way. We then propose to rely only on these word representations to detect the event roles whereas, in most works (Riloff, 1996; Patwardhan and Riloff, 2007; Huang and Riloff, 2012a; Huang and Riloff, 2012b), the role fillers are represented by a set of different features (raw words, their parts-of-speech, syntactic or semantic roles in the sentence).

Furthermore, we propose two additional contributions to the construction of the word representations. The first one is to exploit limited knowledge about the event types (seed words) to improve the learning procedure by better selecting the dictionary. The second one is to use a *max* operation[1] on the word vector representations in order to build noun phrase representations (since slot fillers are generally noun phrases), which represents a better way of aggregating the semantic information born by the word representations.

2.1 Inducing Domain-Relevant Word Representations

In order to induce the domain-specific word representations, we project the words into a 50-dimensional word space. We chose a single layer neural network (NN) architecture that avoids strongly engineered features, assumes little prior knowledge about the task, but is powerful enough to capture relevant domain information. Following (Collobert et al., 2011), we use an NN which learns to predict whether a given text sequence (short word window) exists naturally in the considered domain. We represent an input sequence of n words as $\langle w_i \rangle = \langle w_{i-(n/2)} \ldots, w_i, \ldots w_{i+(n/2)} \rangle$. The main idea is that each sequence of words in the training set should receive a higher score than a sequence in which one word is replaced with a random one. We call the sequence with a random word *corrupted* ($\langle \bar{w_i} \rangle$) and denote as *correct* ($\langle w_i \rangle$) all the sequences of words from the data set. The goal of the training step is then to minimize the following loss function for a word w_i in the dictionary D: $C_{w_i} = \sum_{w_i \in D} max(0, 1 - g(\langle w_i \rangle) + g(\langle \bar{w_i} \rangle))$, where $g(\cdot)$ is the scoring function given by the neural network. Further details and evaluations of these embeddings can be found in (Bengio et al., 2003; Bengio et al., 2006; Collobert and Weston, 2008; Turian et al., 2010). For efficiency, words are fed to our architecture as indices taken from a finite dictionary. Obviously, a simple index does not carry much useful information about the word. So, the first layer of our network maps each of these word indices into a feature vector, by a lookup table operation. Our first contribution intervenes in the process of the choosing the proper dictionary. (Bengio, 2009) has shown that the order of the words in the dictionary of the neural network is not indifferent to the quality of the achieved representations: he proposed to order the dictionary by frequency and select the words for the corrupted sequence according to this order. In our case, the most frequent words are not always the most relevant for the task of event role detection. Since we want to have a training more focused to the domain specific task, we chose to order the dictionary by word relevance to the domain. We accomplish this by considering a limited number of seed words for each event type that needs to be discovered in text (e.g. *attack, bombing, kidnapping, arson*). We then rate with higher values the words that are more similar to the event types words, according to a given semantic similarity, and we rank them accordingly. We use the "Leacock Chodorow" similarity from Wordnet 3.0 (Leacock and Chodorow, 1998). Initial experimental results proved that using this domain-

[1] This max operation consists in taking, for each component of the vector, the max value of this component for each word vector representation.

oriented order leads to better performance for the task than the order by frequency.

2.2 Using Word Representations to Identify Event Roles

After having generated for each word their vector representation, we use them as features for the annotated data to classify event roles. However, event role fillers are not generally single words but noun phrases that can be, in some cases, identified as named entities. For identifying the event roles, we therefore apply a two-step strategy. First, we extract the noun chunks using SENNA[2] parser (Collobert et al., 2011; Collobert, 2011) and we build a representation for these chunks defined as the maximum, per column, of the vector representations of the words it contains. Second, we use a statistical classifier to recognize the slot fillers, using this representation as features. We chose the extra-trees ensemble classifier (Geurts et al., 2006), which is a meta estimator that fits a number of randomized decision trees (*extra-trees*) on various sub-samples of the data set and use averaging to improve the predictive accuracy and control over-fitting.

3 Experiments and Results

3.1 Task Description

We conducted the experiments on the official MUC-4 training corpus that consists of 1,700 documents and instantiated templates for each document. The task consists in extracting information about terrorist events in Latin America from news articles. We classically considered the following 4 types of events: *attack*, *bombing*, *kidnapping* and *arson*. These are represented by templates containing various slots for each piece of information that should be extracted from the document (perpetrators, human targets, physical targets, etc). Following previous works (Huang and Riloff, 2011; Huang and Riloff, 2012a), we only consider the "String Slots" in this work (other slots need different treatments) and we group certain slots to finally consider the five slot types *PerpInd* (individual perpetrator), *PerpOrg* (organizational perpetrator), *Target* (physical target), *Victim* (human target name or description) and *Weapon* (instrument id or type). We used 1,300 documents (DEV) for training, 200 documents (TST1+TST2) for tuning, and 200 documents (TST3+TST4) as the blind test set. To compare with similar works, we do not evaluate the template construction and only focus on the identification of the slot fillers: for each answer key in a reference template, we check if we find it correctly with our extraction method, using head noun matching (e.g., the victim *her mother Martha Lopez Orozco de Lopez* is considered to match *Matha Lopez*), and merging duplicate extractions (so that different extracted slot fillers sharing the same head noun are counted only once). We also took into account the answer keys with multiple values in the reference, dealing with conjunctions (when several victims are named, we need to find all of them) and disjunctions (when several names for the same organization are possible, we need to find any of them). Our results are reported as Precision/Recall/F1-score for each event role separately and averaged on all roles.

3.2 Experiments

In all the experiments involving our model, we established the following stable choices of parameters: 50-dimensional vectors obtained by training on sequences of 5 words, which is consistent with previous studies (Turian et al., 2010; Collobert and Weston, 2008). All the hyper-parameters of our model (e.g. learning rate, size of the hidden layer, size of the word vectors) have been chosen by finetuning our event extraction system on the TST1+TST2 data set. For *DRVR-50* and *W2V-50*, the embeddings were built from the whole training corpus (1,300 documents) and the dictionary was made of all the words of this corpus under their inflected form.

We used the extra-trees ensemble classifier implemented in (Pedregosa et al., 2011), with hyper-parameters optimized on the validation data: forest of 500 trees and the maximum number of features to consider when looking for the best split is $\sqrt{number\_features}$. We present a 3-fold evaluation: first, we compare our system with state-of-the-art systems on the same task, then we compare our domain-relevant vector representations (*DRVR-50*) to more generic word embeddings (*C&W50*, *HLBL-50*)[3] and finally to another

[2]Code and resources can be found at `http://ml.nec-labs.com/senna/`

[3]*C&W-50* are described in (Collobert and Weston, 2008), *HLBL-50* are the Hierarchical log-bilinear embeddings (Mnih and Hinton, 2007), provided by (Turian et al., 2010), available at `http://metaoptimize.com/projects/wordreprs` induced from the Reuters-RCV1

State-of-the-art systems						
	PerpInd	**PerpOrg**	**Target**	**Victim**	**Weapon**	**Average**
(Riloff, 1996)	33/49/40	53/33/41	54/59/56	49/54/51	38/44/41	45/48/46
(Patwardhan and Riloff, 2007)	39/48/43	55/31/40	37/60/46	44/46/45	47/47/47	44/36/40
(Patwardhan and Riloff, 2009)	51/58/54	34/45/38	43/72/53	55/58/56	57/53/55	48/57/52
(Huang and Riloff, 2011)	48/57/52	46/53/50	51/73/60	56/60/58	53/64/58	51/62/56
(Huang and Riloff, 2012a)	47/51/47	60/39/47	37/65/47	39/53/45	53/55/54	47/53/50
(Huang and Riloff, 2012b)	54/57/56	55/49/51	55/68/61	63/59/61	62/64/63	58/60/59
Models based on word embeddings						
C&W-50	80/55/65	64/65/64	76/72/74	53/63/57	85/64/73	68/63/65
HLBL-50	81/53/64	63/67/65	78/72/75	53/63/58	93/64/75	69/62/66
W2V-50	79/57/66	88/71/79	74/72/73	69/75/71	97/65/78	77/68/72
DRVR-50	79/57/66	91/74/81	79/57/66	77/75/76	92/58/81	80/67/73

Table 1: Accuracy of "String Slots" on the TST3 + TST4 test set P/R/F1 (Precision/Recall/F1-Score)

word representation construction on the domain-specific data (*W2V-50*)[4].

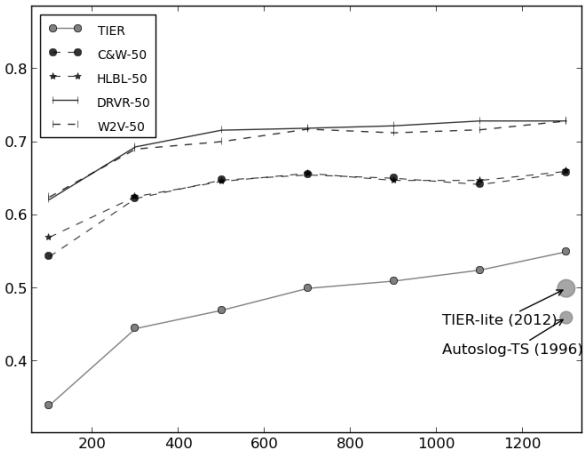


Figure 1: F1-score results for event role labeling on MUC-4 data, for different size of training data, of "String Slots" on the TST3+TST4 with different parameters, compared to the learning curve of TIER (Huang and Riloff, 2012a). The grey points represent the performances of other IE systems.

Figure 1 presents the average F1-score results, computed over the slots PerpInd, PerpOrg, Target, Victim and Weapon. We observe that models relying on word embeddings globally outperform the state-of-the-art results, which demonstrates that the word embeddings capture enough semantic information to perform the task of event role labeling on "String Slots" without using any additional hand-engineered features. Moreover, our representations (*DRVR-50*) clearly surpass the models based on generic embeddings (C&W-50 and HLBL-50) and obtain better results than W2V-50, based the competitive model of (Mikolov et al., 2013a), even if the difference is small. We can also note that the performance of our model is good even with a small amount of training data, which makes it a good candidate to easily develop an event extraction system on a new domain.

Table 1 provides a more detailed analysis of the comparative results. We can see in this table that our results surpass those of previous systems (0.73 vs. 0.59) with, particularly, a consistently higher precision on all roles, whereas recall is smaller for certain roles (Target and Weapon). To further explore the impact of these representations, we compared our word embeddings with other word embeddings (*C&W-50*, *HLBL-50*) and report the results in Figure 1 and Table 1. The results show that our model also outperforms the models using others word embeddings (F1-score of 0.73 against 0.65, 0.66). This proves that a model learned on a domain-specific data set does indeed provide better results, even if its size is much smaller (whereas it is usually considered that neural models require often important training data). Finally, we also achieve slightly better results than *W2V-50* with other word representations built on the same corpus, which shows that the choices made for the word representation construction, such as the use of domain information for word ordering, tend to have a positive impact.

newswire corpus

[4] *W2V-50* are the embeddings induced from the MUC4 data set using the negative sampling training algorithm (Mikolov et al., 2013a; Mikolov et al., 2013b; Mikolov et al., 2013c), available at `https://code.google.com/p/word2vec/`

4 Conclusions and Perspectives

We presented in this paper a new approach for event extraction by reducing the features to only use unsupervised word representations and a small set of seed words. The word embeddings induced from a domain-specific corpus bring improvement over state-of-art models on the standard MUC-4 corpus and demonstrate a good scalability on different sizes of training data sets. Therefore, our proposal offers a promising path towards easier and faster domain adaptation. We also prove that using a domain-specific corpus leads to better word vector representations for this task than using other publicly-available word embeddings (even if they are induced from a larger corpus).

As future work, we will reconsider the architecture of the neural network and we will refocus on creating a deep learning model while taking advantage of a larger set of types of information such as syntactic information, following (Levy and Goldberg, 2014), or semantic information, following (Yu and Dredze, 2014).

References

Yoshua Bengio, Rejean Ducharme, and Pascal Vincent. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Yoshua Bengio, Holger Schwenk, Jean-Sébastian Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In DawnE. Holmes and LakhmiC. Jain, editors, *Innovations in Machine Learning*, volume 194 of *Studies in Fuzziness and Soft Computing*, pages 138–186. Springer Berlin Heidelberg.

Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2(1).

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2012)*, pages 127–135.

Razvan Bunescu and Raymond J Mooney. 2004. Collective information extraction with relational markov networks. In *42nd Annual Meeting on Association for Computational Linguistics (ACL-04)*, pages 438–445.

Hai Leong Chieu, Hwee Tou Ng, and Yoong Keok Lee. 2003. Closing the gap: Learning-based information extraction rivaling knowledge-engineering methods. In *41st international Annual Meeting on Association for Computational Linguistics (ACL-2003)*, pages 216–223.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *25th International Conference of Machine learning (ICML-08)*, pages 160–167. ACM.

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

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *14th International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*.

Dayne Freitag. 1998. Information extraction from HTML: Application of a general machine learning approach. In *AAAI'98*, pages 517–523.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine Learning*, 63(1):3–42.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *28th International Conference on Machine Learning (ICML-11)*, pages 513–520.

Ruihong Huang and Ellen Riloff. 2011. Peeling back the layers: Detecting event role fillers in secondary contexts. In *ACL 2011*, pages 1137–1147.

Ruihong Huang and Ellen Riloff. 2012a. Bootstrapped training of event extraction classifiers. In *13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012)*, pages 286–295.

Ruihong Huang and Ellen Riloff. 2012b. Modeling textual cohesion for event extraction. In *26th Conference on Artificial Intelligence (AAAI 2012)*.

Patrik Lambert, Holger Schwenk, and Frédéric Blain. 2012. Automatic translation of scientific documents in the hal archive. In *LREC 2012*, pages 3933–3936.

Claudia Leacock and Martin Chodorow. 1998. Combining local context and Wordnet similarity for word sense identification. In Christiane Fellbaum, editor, *WordNet: An electronic lexical database.*, pages 265–283. MIT Press.

Wendy Lehnert, Claire Cardie, David Fisher, John McCarthy, Ellen Riloff, and Stephen Soderland. 1992. University of Massachusetts: MUC-4 test results and analysis. In *4th Conference on Message understanding*, pages 151–158.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), Short Papers*, pages 302–308, Baltimore, Maryland, June.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR 20013), workshop track*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 3111–3119.

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

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical modelling. In *24th International Conference of Machine learning (ICML 2007)*, pages 641–648. ACM.

Siddharth Patwardhan and Ellen Riloff. 2007. Effective information extraction with semantic affinity patterns and relevant regions. In *2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 717–727.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, pages 151–160.

Siddharth Patwardhan. 2010. *Widening the field of view of information extraction through sentential event recognition*. Ph.D. thesis, University of Utah.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *AAAI'96*, pages 1044–1049.

Holger Schwenk and Philipp Koehn. 2008. Large and diverse language models for statistical machine translation. In *IJCNLP 2008*, pages 661–666.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *28th International Conference on Machine Learning (ICML-11)*, pages 129–136.

Stephen Soderland, Brendan Roof, Bo Qin, Shi Xu, Mausam, and Oren Etzioni. 2010. Adapting open information extraction to domain-specific relations. *AI Magazine*, 31(3):93–102.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic ie pattern acquisition. In *41st Annual Meeting on Association for Computational Linguistics (ACL-03)*, pages 224–231.

Mihai Surdeanu, Jordi Turmo, and Alicia Ageno. 2006. A hybrid approach for the acquisition of information extraction patterns. In *EACL-2006 Workshop on Adaptive Text Extraction and Mining (ATEM 2006)*, pages 48–55.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *48th international Annual Meeting on Association for Computational Linguistics (ACL 2010)*, pages 384–394.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Automatic acquisition of domain knowledge for information extraction. In *18th Internation Conference on Computational Linguistics (COLING 2000)*, pages 940–946.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), Short Papers*, pages 545–550, Baltimore, Maryland, June.

Modeling Joint Entity and Relation Extraction with Table Representation

Makoto Miwa and **Yutaka Sasaki**
Toyota Technological Institute
2-12-1 Hisakata, Tempaku-ku, Nagoya, 468-8511, Japan
{makoto-miwa, yutaka.sasaki}@toyota-ti.ac.jp

Abstract

This paper proposes a history-based structured learning approach that jointly extracts entities and relations in a sentence. We introduce a novel simple and flexible table representation of entities and relations. We investigate several feature settings, search orders, and learning methods with inexact search on the table. The experimental results demonstrate that a joint learning approach significantly outperforms a pipeline approach by incorporating global features and by selecting appropriate learning methods and search orders.

1 Introduction

Extraction of entities and relations from texts has been traditionally treated as a pipeline of two separate subtasks: entity recognition and relation extraction. This separation makes the task easy to deal with, but it ignores underlying dependencies between and within subtasks. First, since entity recognition is not affected by relation extraction, errors in entity recognition are propagated to relation extraction. Second, relation extraction is often treated as a multi-class classification problem on pairs of entities, so dependencies between pairs are ignored. Examples of these dependencies are illustrated in Figure 1. For dependencies between subtasks, a *Live_in* relation requires *PER* and *LOC* entities, and vice versa. For in-subtask dependencies, the *Live_in* relation between "Mrs. Tsutayama" and "Japan" can be inferred from the two other relations.

Figure 1 also shows that the task has a flexible graph structure. This structure usually does not cover all the words in a sentence differently from other natural language processing (NLP) tasks such as part-of-speech (POS) tagging and dependency parsing, so local constraints are considered to be more important in the task.

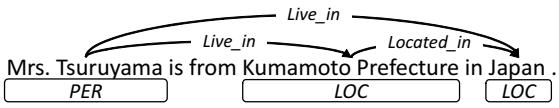


Figure 1: An entity and relation example (Roth and Yih, 2004). Person (*PER*) and location (*LOC*) entities are connected by *Live_in* and *Located_in* relations.

Joint learning approaches (Yang and Cardie, 2013; Singh et al., 2013) incorporate these dependencies and local constraints in their models; however most approaches are time-consuming and employ complex structures consisting of multiple models. Li and Ji (2014) recently proposed a history-based structured learning approach that is simpler and more computationally efficient than other approaches. While this approach is promising, it still has a complexity in search and restricts the search order partly due to its semi-Markov representation, and thus the potential of the history-based learning is not fully investigated.

In this paper, we introduce an entity and relation table to address the difficulty in representing the task. We propose a joint extraction of entities and relations using a history-based structured learning on the table. This table representation simplifies the task into a table-filling problem, and makes the task flexible enough to incorporate several enhancements that have not been addressed in the previous history-based approach, such as search orders in decoding, global features from relations to entities, and several learning methods with inexact search.

2 Method

In this section, we first introduce an entity and relation table that is utilized to represent the whole

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1858–1869,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

entity and relation structures in a sentence. We then overview our model on the table. We finally explain the decoding, learning, search order, and features in our model.

2.1 Entity and relation table

The task we address in this work is the extraction of entities and their relations from a sentence. Entities are typed and may span multiple words. Relations are typed and directed.

We use words to represent entities and relations. We assume entities do not overlap. We employ a BILOU (Begin, Inside, Last, Outside, Unit) encoding scheme that has been shown to outperform the traditional BIO scheme (Ratinov and Roth, 2009), and we will show that this scheme induces several label dependencies between words and between words and relations in §2.3.2. A label is assigned to a word according to the relative position to its corresponding entity and the type of the entity. Relations are represented with their types and directions. $\perp$ denotes a non-relation pair, and $\rightarrow$ and $\leftarrow$ denote left-to-right and right-to-left relations, respectively. Relations are defined on not entities but words, since entities are not always given when relations are extracted. Relations on entities are mapped to relations on the last words of the entities.

Based on this representation, we propose an entity and relation table that jointly represents entities and relations in a sentence. Figure 2 illustrates an entity and relation table corresponding to an example in Figure 1. We use only the lower triangular part because the table is symmetric, so the number of cells is $n(n+1)/2$ when there are n words in a sentence. With this entity and relation table representation, the joint extraction problem can be mapped to a table-filling problem in that labels are assigned to cells in the table.

2.2 Model

We tackle the table-filling problem by a history-based structured learning approach that assigns labels to cells one by one. This is mostly the same as the traditional history-based model (Collins, 2002) except for the table representation.

Let $\mathbf{x}$ be an input table, $\mathbf{Y}(\mathbf{x})$ be all possible assignments to the table, and $s(\mathbf{x}, \mathbf{y})$ be a scoring function that assesses the assignment of $\mathbf{y} \in \mathbf{Y}(\mathbf{x})$ to $\mathbf{x}$. With these definitions, we define our model to predict the most probable assignment as follows:

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in \mathbf{Y}(\mathbf{x})} s(\mathbf{x}, \mathbf{y}) \quad (1)$$

This scoring function is a decomposable function, and each decomposed function assesses the assignment of a label to a cell in the table.

$$s(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{|\mathbf{x}|} s(\mathbf{x}, \mathbf{y}, 1, i) \quad (2)$$

Here, i represents an index of a cell in the table, which will be explained in §2.3.1. The decomposed function $s(\mathbf{x}, \mathbf{y}, 1, i)$ corresponds to the i-th cell. The decomposed function is represented as a linear model, i.e., an inner product of features and their corresponding weights.

$$s(\mathbf{x}, \mathbf{y}, 1, i) = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}, 1, i) \quad (3)$$

The scoring function are further divided into two functions as follows:

$$s(\mathbf{x}, \mathbf{y}, 1, i) = s_{local}(\mathbf{x}, \mathbf{y}, i) + s_{global}(\mathbf{x}, \mathbf{y}, 1, i) \quad (4)$$

Here, $s_{local}(\mathbf{x}, \mathbf{y}, i)$ is a local scoring function that assesses the assignment to the i-th cell without considering other assignments, and $s_{global}(\mathbf{x}, \mathbf{y}, 1, i)$ is a global scoring function that assesses the assignment in the context of 1st to $(i-1)$-th assignments. This global scoring function represents the dependencies between entities, between relations, and between entities and relations. Similarly, features $\mathbf{f}$ are divided into local features $\mathbf{f}_{local}$ and global features $\mathbf{f}_{global}$, and they are defined on its target cell and surrounding contexts. The features will be explained in §2.5. The weights $\mathbf{w}$ can also be divided, but they are tuned jointly in learning as shown in §2.4.

2.3 Decoding

The scoring function $s(\mathbf{x}, \mathbf{y}, 1, i)$ in Equation (2) uses all the preceding assignments and does not rely on the Markov assumption, so we cannot employ dynamic programming.

We instead employ a beam search to find the best assignment with the highest score (Collins and Roark, 2004). The beam search assigns labels to cells one by one with keeping the top K best assignments when moving from a cell to the next cell, and it returns the best assignment when labels are assigned to all the cells. The pseudo code for decoding with the beam search is shown in Figure 3.

	Mrs.	Tsutayama	is	from	Kumamoto	Prefecture	in	Japan	.
Mrs.	B-PER								
Tsutayama	⊥	L-PER							
is	⊥	⊥	O						
from	⊥	⊥	⊥	O					
Kumamoto	⊥	⊥	⊥	⊥	B-LOC				
Prefecture	⊥	Live_in→	⊥	⊥	⊥	L-LOC			
in	⊥	⊥	⊥	⊥	⊥	⊥	O		
Japan	⊥	Live_in→	⊥	⊥	⊥	Located_in→	⊥	U-LOC	
.	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥

Figure 2: The entity and relation table for the example in Figure 1.

INPUT: $\mathbf{x}$: input table with no assignment,
K: beam size
OUTPUT: best assignment $\mathbf{y}^*$ for $\mathbf{x}$
1: $b \leftarrow [\mathbf{x}]$
2: **for** $i = 1$ to $|\mathbf{x}|$ **do**
3: $T \leftarrow \emptyset$
4: **for** $k = 1$ to $|b|$ **do**
5: **for** $a \in \mathbf{A}(i, b[k])$ **do**
6: $T \leftarrow T \cup$ append(a, $b[k]$)
7: **end for**
8: **end for**
9: $b \leftarrow$ top K tables from T using the scoring function in Equation (2)
10: **end for**
11: return b[0]

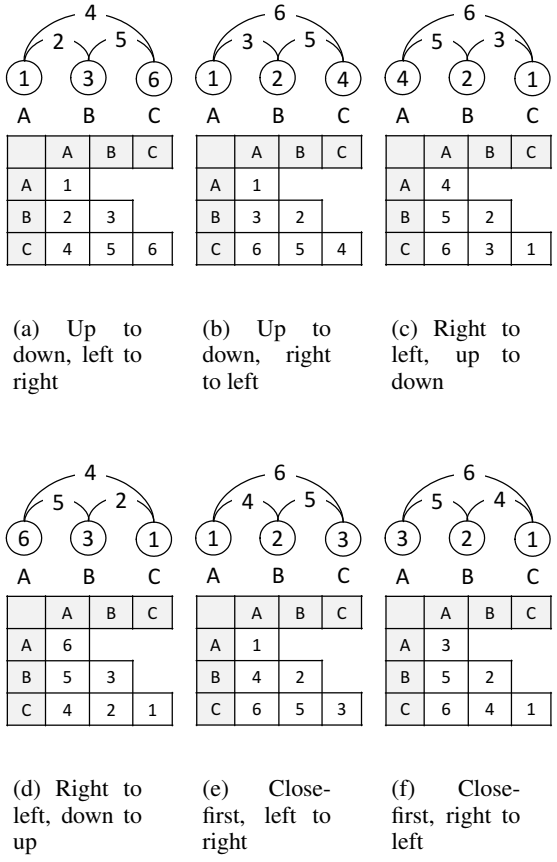
Figure 3: Decoding with the beam search. $\mathbf{A}(i, t)$ returns possible assignments for i-th cell of a table t, and append(a, t) returns a table t updated with an assignment a.

We explain how to map the table to a sequence (line 2 in Figure 3), and how to calculate possible assignments (line 6 in Figure 3) in the following subsections.

2.3.1 Table-to-sequence mapping

Cells in an input table are originally indexed in two dimensions. To apply our model in §2.2 to the cells, we need to map the two-dimensional table to a one-dimensional sequence. This is equivalent to defining a search order in the table, so we will use the terms "mapping" and "search order" interchangeably.

Since it is infeasible to try all possible mappings, we define six promising static mappings (search orders) as shown in Figure 4. Note that the "left" and "right" directions in the captions correspond to not word orders, but tables. We define two mappings (Figures 4(a) and 4(b)) with the highest priority on the "up to down" order, which checks a sentence forwardly (from the beginning of a sentence). Similarly, we also define two mappings (Figures 4(c) and 4(d)) with the highest priority on the "right to left" order, which check a sentence backwardly (from the end of a sentence). From another point of view, entities are detected before relations in Figures 4(b) and 4(c) whereas the order in a sentence is prioritized in Figures 4(a)



(a) Up to down, left to right

(b) Up to down, right to left

(c) Right to left, up to down

(d) Right to left, down to up

(e) Close-first, left to right

(f) Close-first, right to left

Figure 4: Static search orders.

Condition	Possible labels on w_i
Relation(s) on w_{i-1}	B-*, O, U-*
Relation(s) on w_i	L-*, U-*

Table 1: Label dependencies from relations to entities. * indicates any type.

Label on w_i	Relations from/to w_i
B-*, I-*, O	$\perp$
L-*, U-*	*
Label on w_{i+1}	Relations from/to w_i
I-*, L-*	$\perp$
B-*, U-*, O	*

Table 2: Label dependencies from entities to relations.

and 4(d). We further define two close-first mappings (Figures 4(e) and 4(f)) since entities are easier to find than relations and close relations are easier to find than distant relations.

We also investigate dynamic mappings (search orders) with an easy-first policy (Goldberg and Elhadad, 2010). Dynamic mappings are different from the static mappings above, since we reorder the cells before each decoding[1]. We evaluate the cells using the local scoring function, and assign indices to the cells so that the cells with higher scores have higher priorities. In addition to this naïve easy-first policy, we define two other dynamic mappings that restricts the reordering by combining the easy-first policy with one of the following two policies: entity-first (all entities are detected before relations) and close-first (closer cells are detected before distant cells) policies.

2.3.2 Label dependencies

To avoid illegal assignments to a table, we have to restrict the possible assignments to the cells according to the preceding assignments. This restriction can also reduce the computational costs.

We consider all the dependencies between cells to allow the assignments of labels to the cells in an arbitrary order. Our representation of entities and relations in §2.1 induces the dependencies between entities and between entities and relations. Tables 1-3 summarize these dependencies on the i-th word w_i in a sentence. We can further utilize dependencies between entity types and relation types if some entity types are involved in a limited number of relation types or vice versa. We note that the dependencies between entity types and relation types include not only words participating in relations but also their surrounding words. For example, the label on w_{i-1} can restrict the types of relations involving w_i. We employ these type dependencies in the evaluation, but we omit these dependencies here since these dependencies are dependent on the tasks.

[1]It is also possible to reorder the cells during decoding, but it greatly increases the computational costs.

Label on w_{i-2}	Possible labels on w_i
B-TYPE	B-*, I-TYPE, L-TYPE, O, U-*
I-TYPE	B-*, I-TYPE, L-TYPE, O, U-*
L-TYPE	B-*, I-*, L-*, O, U-*
O	B-*, I-*, L-*, O, U-*
U-TYPE	B-*, I-*, L-*, O, U-*
O/S	B-*, I-*, L-*, O, U-*
Label on w_{i-1}	Possible labels on w_i
B-TYPE	I-TYPE, L-TYPE
I-TYPE	I-TYPE, L-TYPE
L-TYPE	B-*, O, U-*
O	B-*, O, U-*
U-TYPE	B-*, O, U-*
O/S	B-*, O, U-*
Label on w_{i+1}	Possible labels on w_i
B-TYPE	L-*, O, U-*
I-TYPE	B-TYPE, I-TYPE
L-TYPE	B-TYPE, I-TYPE
O	L-*, O, U-*
U-TYPE	L-*, O, U-*
O/S	L-*, O, U-*
Label on w_{i+2}	Possible labels on w_i
B-TYPE	B-*, I-*, L-*, O, U-*
I-TYPE	B-TYPE, I-TYPE, L-*, O, U-*
L-TYPE	B-TYPE, I-TYPE, L-*, O, U-*
O	B-*, I-*, L-*, O, U-*
U-TYPE	B-*, I-*, L-*, O, U-*
O/S	B-*, I-*, L-*, O, U-*

Table 3: Label dependencies between entities. TYPE represents an entity type, and O/S means the word is outside of a sentence.

2.4 Learning

The goal of learning is to minimize errors between predicted assignments $\mathbf{y}^*$ and gold assignments $\mathbf{y}^{gold}$ by tuning the weights $\mathbf{w}$ in the scoring function in Equation 3. We employ a margin-based structured learning approach to tune the weights $\mathbf{w}$. The pseudo code is shown in Figure 5. This approach enhances the traditional structured percep-

INPUT: training sets $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$,
T: iterations
OUTPUT: weights $\mathbf{w}$

1: $\mathbf{w} \leftarrow \mathbf{0}$
2: **for** $t = 1$ to T **do**
3: **for** $\mathbf{x}, \mathbf{y} \in D$ **do**
4: $\mathbf{y}^* \leftarrow$ best assignment for $\mathbf{x}$ using decoding in Figure 3 with s' in Equation (5)
5: **if** $\mathbf{y}^* \neq \mathbf{y}^{gold}$ **then**
6: $m \leftarrow \arg\max_i \{s'(\mathbf{x}, \mathbf{y}^{gold}, 1, i) - s'(\mathbf{x}, \mathbf{y}^*, 1, i)\}$
7: $\mathbf{w} \leftarrow \text{update}(\mathbf{w}, f(\mathbf{x}, \mathbf{y}^{gold}, 1, m), f(\mathbf{x}, \mathbf{y}^*, 1, m))$
8: **end if**
9: **end for**
10: **end for**
11: return $\mathbf{w}$

Figure 5: Margin-based structured learning approach with a max-violation update. $\text{update}(\mathbf{w}, f(\mathbf{x}, \mathbf{y}^{gold}, 1, m), f(\mathbf{x}, \mathbf{y}^*, 1, m))$ depends on employed learning methods.

tron (Collins, 2002) in the following ways. Firstly, we incorporate a margin Δ into the scoring function as follows so that wrong assignments with small differences from gold assignments are penalized (lines 4 and 6 in Figure 5) (Freund and Schapire, 1999).

$$\mathbf{s}'(\mathbf{x}, \mathbf{y}) = \mathbf{s}(\mathbf{x}, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}^{gold}) \quad (5)$$

Similarly to the scoring function $\mathbf{s}$, the margin Δ is defined as a decomposable function using 0-1 loss as follows:

$$\begin{aligned} \Delta(\mathbf{y}, \mathbf{y}^{gold}) &= \sum_{i=1}^{|\mathbf{x}|} \Delta(y_i, y_i^{gold}), \\ \Delta(y_i, y_i^{gold}) &= \begin{cases} 0 & \text{if } y_i = y_i^{gold} \\ 1 & \text{otherwise} \end{cases} \end{aligned} \quad (6)$$

Secondly, we update the weights $\mathbf{w}$ based on a max-violation update rule following Huang et al. (2012) (lines 6-7 in Figure 5). Finally, we employ not only perceptron (Collins, 2002) but also AROW (Mejer and Crammer, 2010; Crammer et al., 2013), AdaGrad (Duchi et al., 2011), and DCD-SSVM (Chang and Yih, 2013) for learning methods (line 7 in Figure 5.) We employ parameter averaging except for DCD-SSVM. AROW and AdaGrad store additional information for covariance and feature counts respectively, and DCD-SSVM keeps a working set and performs additional updates in each iteration. Due to space limitations, we refer to the papers for the details of the learning methods.

2.5 Features

Here, we explain the local features $\mathbf{f}_{local}$ and the global features $\mathbf{f}_{global}$ introduced in §2.2.

2.5.1 Local features

Our focus is not to exploit useful local features for entities and relations, so we incorporate several features from existing work to realize a reasonable baseline. Table 4 summarizes the local features. Local features for entities (or words) are similar to the features used by Florian et al. (2003), but some features are generalized and extended, and gazetteer features are excluded. For relations (or pairs of words), we employ and extend features in Miwa et al. (2009).

2.5.2 Global features

We design global features to represent dependencies among entities and relations. Table 5 summarizes the global features[2]. These global features are activated when all the information is available during decoding.

We incorporate label dependency features like traditional sequential labeling for entities. Although our model can include other non-local features between entities (Ratinov and Roth, 2009), we do not include them expecting that global features on entities and relations can cover them. We design three types of global features for relations. These features are activated when all the participating relations are not $\perp$ (non-relations). Features except for the "Crossing" category are similar to global relation features in Li and Ji (2014). We further incorporate global features for both entities and relations. These features are activated when the relation label is not $\perp$. These features can act as a bridge between entities and relations.

3 Evaluation

In this section, we first introduce the corpus and evaluation metrics that we employed for evaluation. We then show the performance on the training data set with explaining the parameters used

[2]We tried other "Entity+Relation" features to represent a relation and both its participating entities, but they slightly degraded the performance in our preliminary experiments.

Target	Category	Features
Word (Entity)	Lexical	Character n-grams (n=2,3,4)
		Attributes by parsers (base form, POS)
		Word types (all-capitalized, initial-capitalized, all-digits, all-puncts, all-digits-or-puncts)
	Contextual	Word n-grams (n=1,2,3) within a context window size of 2
Word pair (Relation)	Entity	Entity lexical features of each word
	Contextual	Word n-grams (n=1,2,3) within a context window size of 2
	Shortest path	Walk features (word-dependency-word or dependency-word-dependency) on the shortest paths in parsers' outputs
		n-grams (n=2,3) of words and dependencies on the paths
		n-grams (n=1,2) of token modifier-modifiee pairs on the paths
		The length of the paths

Table 4: Local features.

Target	Category	Details
Entity	Bigram	Bigrams of labels
		Combinations of two labels and their corresponding POS tags
		Combinations of two labels and their corresponding words
	Trigram	Trigrams of labels
		Combinations of three labels and each of their corresponding POS tags
		Combinations of three labels and each of their corresponding words
	Entity	Combinations of a label and its corresponding entity
Relation	Entity-sharing	Combinations of two relation labels that share a word (i.e., relations in same columns or same rows in a table)
		Combinations of two relation labels and the shared word
		Relation shortest path features between non-shared words, augmented by a combination of relation labels and the shared word
	Cyclic	Combinations of three relation labels that make a cycle
	Crossing	Combinations of two relation labels that cross each other
Entity + Relation	Entity-relation	Relation label and the label of its participating entity
		Relation label and the label and word of its participating entity

Table 5: Global features.

for the test set evaluation, and show the performance on the test data set.

3.1 Evaluation settings

We used an entity and relation recognition corpus by Roth and Yih (2004)[3]. The corpus defines four named entity types *Location*, *Organization*, *Person*, and *Other* and five relation types *Kill*, *Live_In*, *Located_In*, *OrgBased_In* and *Work_For*.

All the entities were words in the original corpus because all the spaces in entities were replaced with slashes. Previous systems (Roth and Yih, 2007; Kate and Mooney, 2010) used these word boundaries as they were, treated the boundaries as given, and focused the entity classification problem alone. Differently from such systems, we recovered these spaces by replacing these slashes with spaces to evaluate the entity boundary detection performance on this corpus. Due to this replacement and the inclusion of the boundary detection problem, our task is more challenging than the original task, and our results are not comparable with those by the previous systems.

The corpus contains 1,441 sentences that contain at least one relation. Instead of 5-fold cross validation on the entire corpus by the previous systems, we split the data set into training (1,153 sentences) and blind test (288 sentences) data sets and

[3]conll04.corp at `http://cogcomp.cs.illinois.edu/page/resource_view/43`

developed the system on the training data set. We tuned the hyper-parameters using a 5-fold cross validation on the training data set, and evaluated the performance on the test set.

We prepared a pipeline approach as a baseline. We first trained an entity recognition model using the local and global features, and then trained a relation extraction model using the local features and global features without global "Relation" features in Table 5. We did not employ the global "Relation" features in this baseline since it is common to treat relation extraction as a multi-class classification problem.

We extracted features using the results from two syntactic parsers Enju (Miyao and Tsujii, 2008) and LRDEP (Sagae and Tsujii, 2007). We employed feature hashing (Weinberger et al., 2009) and limited the feature space to 2^{24}. The numbers of features greatly varied for categories and targets. They also caused biased predictions that prefer entities to relations in our preliminary experiments. We thus chose to re-scale the features as follows. We normalized local features for each feature category and then for each target. We also normalized global features for each feature category, but we did not normalize them for each target since normalization was impossible during decoding. We instead scaled the global features, and the scaling factor was tuned by using the same 5-fold cross validation above.

We used the F1 score on relations with entities as our primary evaluation measure and used it for tuning parameters. In this measure, a relation with two entities is considered correct when the offsets and types of the entities and the type of the relation are all correct. We also evaluated the F1 scores for entities and relations individually on the test data set by checking their corresponding cells. An entity is correct when the offset and type are correct, and a relation is correct when the type is correct and the last words of two entities are correct.

3.2 Performance on Training Data Set

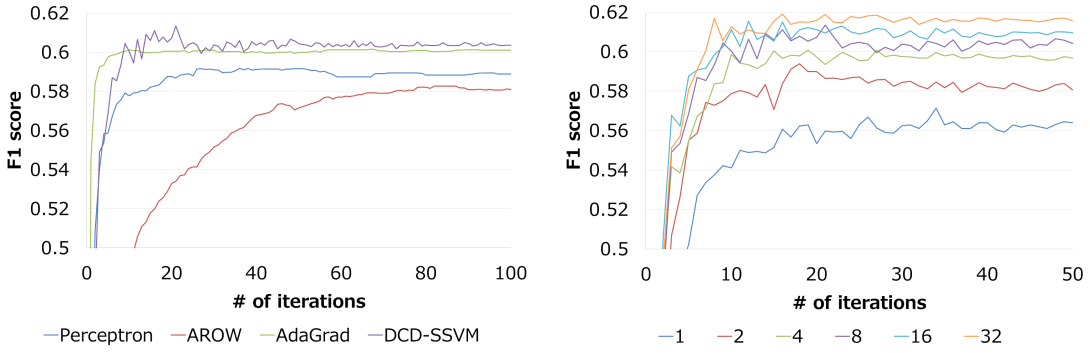
It is infeasible to investigate all the combinations of the parameters, so we greedily searched for a *default parameter setting* by using the evaluated results on the training data set. The default parameter setting was the best setting except for the beam size. We show learning curves on the training data set in Figure 6 when we varied each parameter from the default parameter setting. We employed 5-fold cross validation. The default parameter setting used DCD-SSVM as the learning method, entity-first, easy-first as the search order, local and global features, and 8 as the beam size. This section discusses how these parameters affect the performance on the training data set and explains how the parameter setting was selected for the test set.

Figure 6(a) compares the learning methods introduced in §2.4. DCD-SSVM and AdaGrad performed slightly better than perceptron, which has often been employed in history-based structured learning. AROW did not show comparable performance to the others. We ran 100 iterations to find the number of iterations that saturates learning curves. The large number of iterations took time and the performance of DCD-SSVM almost converged after 30 iterations, so we employed 50 iterations for other evaluation on the training data set. AdaGrad got its highest performance more quickly than other learning methods and AROW converged slower than other methods, so we employed 10 for AdaGrad, 90 for AROW, and 50 iterations for other settings on the test data set.

The performance was improved by widening the beam as in Figure 6(b), but the improvement was gradually diminished as the beam size increased. Since the wider beam requires more training and test time, we chose 8 for the beam size.

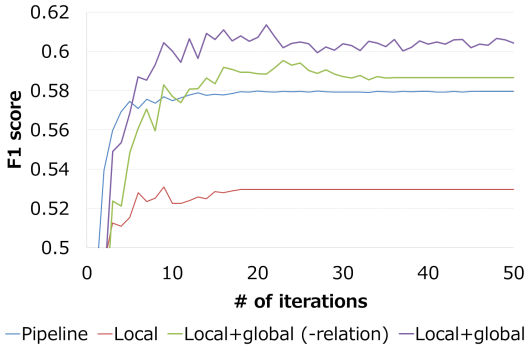
Figure 6(c) shows the effects of joint learning as well as features explained in §2.5. We show the performance of the pipeline approach (Pipeline) introduced in §3.1, and the performance with local features alone (Local), local and global features without global "Relation" features in Table 5 (Local+global ($-$relation)) and all local and global features (Local+global). We note that Pipeline shows the learning curve of relation extraction in the pipeline approach. Features in "Local+global ($-$relation)" are the same as the features in the pipeline approach, and the result shows that the joint learning approach performed slightly better than the pipeline approach. The incorporation of global "Entity" and "Entity+Relation" features improved the performance as is common with the existing pipeline approaches, and relation-related features further improved the performance.

Static search orders in §2.3.1 also affected the performance as shown in Figure 6(d), although search orders are not investigated in the joint entity and relation extraction. Surprisingly, the gap

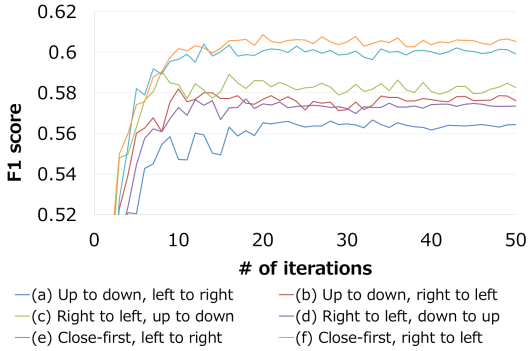


(a) Learning methods

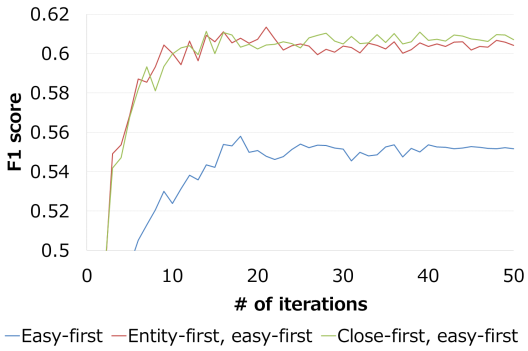
(b) Beam sizes



(c) Features and pipeline / joint approaches



(d) Static search orders



(e) Dynamic search orders

Figure 6: Learning curves of entity and relation extraction on the training data set using 5-fold cross validation.

between the performances with the best order and worst order was about 0.04 in an F1 score, which is statistically significant, and the performance can be worse than the pipeline approach in Figure 6(c). This means improvement by joint learning can be easily cancelled out if we do not carefully consider search order. It is also surprising that the second worst order (Figure 4(b)) is the most intuitive "left-to-right" order, which is closest to the order in Li and Ji (2014) among the six search orders.

Figure 6(e) shows the performance with dynamic search orders. Unfortunately, the easy-first policy did not work well on this entity and relation task, but, with the two enhancements, dynamic orders performed as well as the best static order in Figure 6(d). This shows that entities should be de-

tected earlier than relations on this data set.

3.3 Performance on Test Data Set

Table 6 summarizes the performance on the test data set. We employed the default parameter setting explained in §3.2, and compared parameters by changing the parameters shown in the first column. We performed a statistical test using the approximate randomization method (Noreen, 1989) on our primary measure ("Entity+Relation"). The results are almost consistent with the results on the training data set with a few exceptions.

Differently from the results on the training data set, AdaGrad and AROW performed significantly worse than perceptron and DCD-SSVM and they performed slightly worse than the pipeline approach. This result shows that DCD-SSVM performs well with inexact search and the selection of learning methods can significantly affect the entity and relation extraction performance.

The joint learning approach showed a significant improvement over the pipeline approach with relation-related global features, although the joint learning approach alone did not show a significant improvement over the pipeline approach. Unfortunately, no joint learning approach outperformed the pipeline approach in entity recognition. This may be partly because hyper-parameters were tuned to the primary measure. The results on the pipeline approach also indicate that the better performance on entity recognition does not necessarily improve the relation extraction performance.

Search orders also affected the performance, and the worst order (right to left, down to up) and best order (close-first, left to right) were significantly different. The performance of the worst order was worse than that of the pipeline approach, although the difference was not significant. These results show that it is necessary to carefully select the search order for the joint entity and relation extraction task.

3.4 Comparison with Other Systems

To compare our model with the other systems (Roth and Yih, 2007; Kate and Mooney, 2010), we evaluated the performance of our model when the entity boundaries were given. Differently from our setting in §3.1, we used the gold entity boundaries encoded in the BILOU scheme and assigned entity labels to the boundaries. We performed 5-fold cross validation on the data set following Roth and Yih (2007) although the split was different from theirs since their splits were not available. We employed the default parameter setting in §3.2 for this comparison.

Table 7 shows the evaluation results. Although we cannot directly compare the results, our model performs better than the other models. Compared to Table 6, Table 7 also shows that the inclusion of entity boundary detection degrades the performance about 0.09 in F-score.

4 Related Work

Search order in structured learning has been studied in several NLP tasks. Left-to-right and right-to-left orderings have been often investigated in sequential labeling tasks (Kudo and Matsumoto, 2001). Easy-first policy was firstly introduced by Goldberg and Elhadad (2010) for dependency parsing, and it was successfully employed in several tasks, such as joint POS tagging and dependency parsing (Ma et al., 2012) and co-reference resolution (Stoyanov and Eisner, 2012). Search order, however, has not been focused in relation extraction tasks.

Named entity recognition (Florian et al., 2003; Nadeau and Sekine, 2007) and relation extraction (Zelenko et al., 2003; Miwa et al., 2009) have often been treated as separate tasks, but there are some previous studies that treat entities and relations jointly in learning. Most studies built joint learning models upon individual models for subtasks, such as Integer Linear Programming (ILP) (Roth and Yih, 2007; Yang and Cardie, 2013) and Card-Pyramid Parsing (Kate and Mooney, 2010). Our approach does not require such individual models, and it also can detect entity boundaries that these approaches except for Yang and Cardie (2013) did not treat. Other studies (Yu and Lam, 2010; Singh et al., 2013) built global probabilistic graphical models. They need to compute distributions over variables, but our approach does not. Li and Ji (2014) proposed an approach to jointly find entities and relations. They incorporated a semi-Markov chain in representing entities and they defined two actions during search, but our approach does not employ such representation and actions, and thus it is more simple and flexible to investigate search orders.

5 Conclusions

In this paper, we proposed a history-based structured learning approach that jointly detects enti-

Parameter	Entity	Relation	Entity+Relation
Perceptron	0.809 / 0.809 / 0.809	0.760 / 0.547 / 0.636	0.731 / 0.527 / 0.612*
AdaGrad	0.801 / 0.790 / 0.795	0.732 / 0.486 / 0.584	0.716 / 0.476 / 0.572
AROW	0.810 / 0.802 / 0.806	0.797 / 0.468 / 0.590	0.758 / 0.445 / 0.561
DCD-SSVM†	0.812 / 0.802 / 0.807	0.783 / 0.524 / 0.628	0.760 / 0.509 / 0.610*
Pipeline	0.823 / 0.814 / 0.818	0.672 / 0.542 / 0.600	0.647 / 0.522 / <u>0.577</u>
Local	0.819 / 0.812 / 0.815	0.844 / 0.399 / 0.542	0.812 / 0.384 / 0.522
Local + global ($-$relation)	0.809 / 0.799 / 0.804	0.784 / 0.481 / 0.596	0.747 / 0.458 / 0.568
Local + global†	0.812 / 0.802 / 0.807	0.783 / 0.524 / 0.628	0.760 / 0.509 / 0.610*
(a) Up to down, left to right	0.824 / 0.801 / 0.813	0.821 / 0.433 / 0.567	0.787 / 0.415 / 0.543
(b) Up to down, right to left	0.828 / 0.808 / 0.818	0.850 / 0.461 / 0.597	0.822 / 0.445 / 0.578
(c) Right to left, up to down	0.823 / 0.799 / 0.811	0.826 / 0.448 / 0.581	0.789 / 0.427 / 0.554
(d) Right to left, down to up	0.811 / 0.784 / 0.797	0.774 / 0.445 / 0.565	0.739 / 0.425 / 0.540
(e) Close-first, left to right	0.821 / 0.806 / 0.813	0.807 / 0.522 / 0.634	0.780 / 0.504 / 0.612*
(f) Close-first, right to left	0.817 / 0.801 / 0.809	0.832 / 0.491 / 0.618	0.797 / 0.471 / 0.592
Easy-first	0.811 / 0.790 / 0.801	0.862 / 0.415 / 0.560	0.831 / 0.399 / 0.540
Entity-first, easy-first†	0.812 / 0.802 / 0.807	0.783 / 0.524 / 0.628	0.760 / 0.509 / 0.610*
Close-first, easy-first	0.816 / 0.803 / 0.810	0.796 / 0.486 / 0.603	0.767 / 0.468 / 0.581

Table 6: Performance of entity and relation extraction on the test data set (precision / recall / F1 score). The † denotes the default parameter setting in §3.2 and * represents a significant improvement over the underlined "Pipeline" baseline ($p<0.05$). Labels (a)-(f) correspond to those in Figure 4.

	Kate and Mooney (2010)	Roth and Yih (2007)	Entity-first, easy-first
Person	0.921 / 0.942 / 0.932	0.891 / 0.895 / 0.890	0.931 / 0.948 / 0.939
Location	0.908 / 0.942 / 0.924	0.897 / 0.887 / 0.891	0.922 / 0.939 / 0.930
Organization	0.905 / 0.887 / 0.895	0.895 / 0.720 / 0.792	0.903 / 0.896 / 0.899
All entities	-	-	0.924 / 0.924 / 0.924
Located_In	0.675 / 0.567 / 0.583	0.539 / 0.557 / 0.513	0.821 / 0.549 / 0.654
Work_For	0.735 / 0.683 / 0.707	0.720 / 0.423 / 0.531	0.886 / 0.642 / 0.743
OrgBased_In	0.662 / 0.641 / 0.647	0.798 / 0.416 / 0.543	0.768 / 0.572 / 0.654
Live_In	0.664 / 0.601 / 0.629	0.591 / 0.490 / 0.530	0.819 / 0.532 / 0.644
Kill	0.916 / 0.641 / 0.752	0.775 / 0.815 / 0.790	0.933 / 0.797 / 0.858
All relations	-	-	0.837 / 0.599 / 0.698

Table 7: Results of entity classification and relation extraction on the data set using the 5-fold cross validation (precision / recall / F1 score).

ties and relations. We introduced a novel entity and relation table that jointly represents entities and relations, and showed how the entity and relation extraction task can be mapped to a simple table-filling problem. We also investigated search orders and learning methods that have been fixed in previous research. Experimental results showed that the joint learning approach outperforms the pipeline approach and the appropriate selection of learning methods and search orders is crucial to produce a high performance on this task.

As future work, we plan to apply this approach to other relation extraction tasks and explore more suitable search orders for relation extraction tasks. We also plan to investigate the potential of this table representation in other tasks such as semantic parsing and co-reference resolution.

Acknowledgments

We thank Yoshimasa Tsuruoka and Yusuke Miyao for valuable discussions, and the anonymous reviewers for their insightful comments. This work was supported by the TTI Start-Up Research Support Program and the JSPS Grant-in-Aid for Young Scientists (B) [grant number 25730129].

References

Ming-Wei Chang and Wen-Tau Yih. 2013. Dual coordinate descent algorithms for efficient large margin structured prediction. *Transactions of the Association for Computational Linguistics*, 1:207–218.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, July.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.

Koby Crammer, Alex Kulesza, and Mark Dredze. 2013. Adaptive regularization of weight vectors. *Machine learning*, 91(2):155–187.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 168–171.

Yoav Freund and Robert E Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750, Los Angeles, California, June. Association for Computational Linguistics.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151, Montréal, Canada, June. Association for Computational Linguistics.

Rohit J. Kate and Raymond Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 203–212, Uppsala, Sweden, July. Association for Computational Linguistics.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, Stroudsburg, PA, USA. Association for Computational Linguistics.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland, June. Association for Computational Linguistics.

Ji Ma, Tong Xiao, Jingbo Zhu, and Feiliang Ren. 2012. Easy-first Chinese POS tagging and dependency parsing. In *Proceedings of COLING 2012*, pages 1731–1746, Mumbai, India, December. The COLING 2012 Organizing Committee.

Avihai Mejer and Koby Crammer. 2010. Confidence in structured-prediction using confidence-weighted models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 971–981, Cambridge, MA, October. Association for Computational Linguistics.

Makoto Miwa, Rune Sætre, Yusuke Miyao, and Jun'ichi Tsujii. 2009. A rich feature vector for protein-protein interaction extraction from multiple corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 121–130, Singapore, August. Association for Computational Linguistics.

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

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

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

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June. Association for Computational Linguistics.

Dan Roth and Wen-Tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 1–8, Boston, Massachusetts, USA, May. Association for Computational Linguistics.

Dan Roth and Wen-Tau Yih, 2007. *Global Inference for Entity and Relation Identification via a Linear Programming Formulation*. MIT Press.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050, Prague, Czech Republic, June. Association for Computational Linguistics.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 1–6. ACM.

Veselin Stoyanov and Jason Eisner. 2012. Easy-first coreference resolution. In *Proceedings of COLING 2012*, pages 2519–2534, Mumbai, India, December. The COLING 2012 Organizing Committee.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1113–1120, New York, NY, USA. ACM.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1640–1649, Sofia, Bulgaria, August. Association for Computational Linguistics.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Coling 2010: Posters*, pages 1399–1407, Beijing, China, August. Coling 2010 Organizing Committee.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106.

ZORE: A Syntax-based System for Chinese Open Relation Extraction

Likun Qiu and Yue Zhang
Singapore University of Technology and Design, Singapore
qiulikun@gmail.com, yue_zhang@sutd.edu.sg

Abstract

Open Relation Extraction (ORE) overcomes the limitations of traditional IE techniques, which train individual extractors for every single relation type. Systems such as ReVerb, PATTY, OLLIE, and Exemplar have attracted much attention on English ORE. However, few studies have been reported on ORE for languages beyond English. This paper presents a syntax-based Chinese (Zh) ORE system, ZORE, for extracting relations and semantic patterns from Chinese text. ZORE identifies relation candidates from automatically parsed dependency trees, and then extracts relations with their semantic patterns iteratively through a novel double propagation algorithm. Empirical results on two data sets show the effectiveness of the proposed system.

1 Introduction

Traditional Information Extraction (IE) systems train extractors for pre-specified relations (Kim and Moldovan, 1993). This approach cannot scale to the web, where target relations are not defined in advance. Open Relation Extraction (ORE) attempts to solve this problem by shallow-parsing-based, syntax-based or semantic-role-based pattern matching without pre-defined relation types, and has achieved great success on open-domain corpora ranging from news to Wikipedia (Banko et al., 2007; Wu and Weld, 2010; Nakashole et al., 2012; Etzioni et al., 2011; Moro and Navigli, 2013). Many NLP and IR applications, including selectional preference learning, commonsense knowledge and entailment rule mining, have benefited from ORE (Ritter et al., 2010). However, most existing ORE systems focus on English, and little research has been reported on other languages. In addition, existing ORE techniques are mainly concerned with the extraction of textual relations, without trying to give semantic analysis, which is the advantage of traditional IE.

Our goal in this paper is to present a syntax-based Chinese (Zh) ORE system, ZORE, which extracts relations by using syntactic dependency patterns, while associating them with explicit semantic information. An example is shown is Figure 1, where the relation *(奥巴马 (Obama) 总统 (President) , Pred[毕业 (graduate)],* 哈佛 *(Harvard)* 法学院 *(Law School))* is extracted from the given sentence "奥巴马 *(Obama)* 总统 *(President)* 毕业 *(graduate)* 于 *(from)* 哈佛 *(Harvard)* 法学院 *(Law School)*", and generalized into the syntactic-semantic pattern {*nsubj-NR(Af) Pred[毕业 (graduate)] prep-于 (from) pobj-NN(Di)*}. Here, *Af* and *Di* stand for human and institution, respectively, according to a Chinese taxonomy *Extended Cilin* (Che et al., 2010).

Rather than extracting binary relations and then generalizing them into semantic patterns, which most previous work does (Mausam et al., 2012; Nakashole et al., 2012; Moro and Navigli, 2012; Moro and Navigli, 2013), we develop a novel method that extracts relations and patterns simultaneously. A double propagation algorithm is used to make relation and pattern information reinforce each other, so that negative effects from automatic syntactic and semantic analysis errors can be mitigated. In this way, semantic pattern information is leveraged to improve relation extraction.

We manually annotate two sets of data, from news text and Wikipedia, respectively. Experiments on both data sets show that the double propagation algorithm gives better precision and recall compared to the baseline. To our knowledge, we are one of the first to report empirical results on Chinese ORE. The ZORE system, together with the two sets of test data we annotated, and the sets of 5 million relations and 344K semantic patterns extracted from news and Wikipedia, is freely re-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1870–1880,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

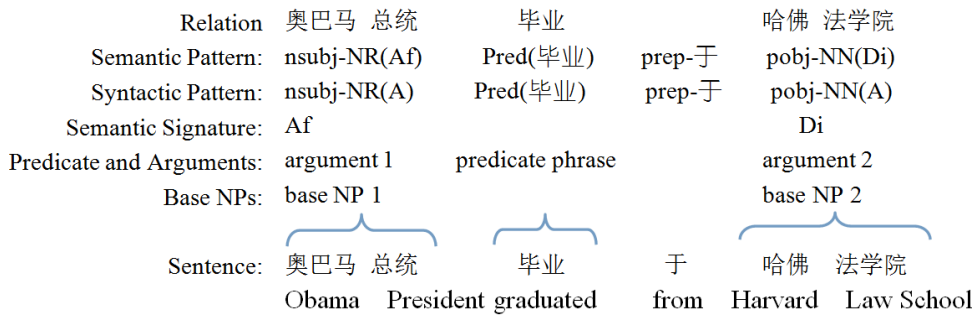


Figure 1: A sample sentence analyzed by ZORE.

leased[1].

2 Basic Definitions for Open Information Extraction

ZORE is applied to web text to extract general relations and their semantic types. Our definition of relations follow previous work on ORE (Moro and Navigli, 2013), but with language-specific adjustments. In this section, we use the sentence in Figure 1 as an instance to describe the basic definitions for ZORE.

Definition 1 (predicate phrase) A ***predicate phrase*** is a sequence of words that contains at least one verb or copula, and governs one or more noun phrases syntactically. For instance, a predicate phrase for the sentence in Figure 1 is "毕业 *(graduate)*". Following Fader et al. (2011), Mausam et al. (2012) and Nakashole et al. (2012), in case of light verb constructions, the verb and its direct object jointly serve as predicate phrase. We do not include prepositions into the predicate phrases.

Definition 2 (argument) An ***argument*** is a base noun phrase governed by a predicate phrase *directly* or *indirectly* with a preposition. For instance, "奥巴马 (Obama) 总统 (President)" and "哈佛 (Harvard) 法学院 (Law School)" are two arguments of the predicate phrase "毕业 *(graduate)*".

Definition 3 (relation) A ***binary relation*** is a triple that consists of the predicate phrase *Pred* and its two arguments *x* and *y*. Accordingly, an *n-ary relation* contains *n* arguments. For instance, the sentence in Figure 1 contains the binary relation *(奥巴马 (Obama) 总统 (President), Pred[毕业 (graduate)], 哈佛 (Harvard) 法学院 (Law School))*. In English, the two arguments of a binary relation are usually positioned on the left and right of *Pred*, respectively. Hence, shallow patterns are highly useful for English relation extraction (Banko et al., 2007). In Chinese, however, the two arguments can be both on the left, both on the right or one on the left and one on the right of the predicate, and the resulting binary relation can be either *(x, y, Pred)*, *(Pred, x, y)* and *(x, Pred, y)*, depending on the sentence. This makes the detection of relation phrases more complicated.

Definition 4 (syntactic pattern) A ***syntactic pattern*** is the syntactic abstraction of a relation. A relation can be generalized into the combination of words, POS-tags and syntactic dependency labels (Nakashole et al., 2012). For instance, the syntactic pattern of the sentence in Figure 1 is {*nsubj-NR(A) Pred[毕业] prep-于 pobj-NN(A)*}. It consists of four sub-patterns. The first, *nsubj-NR(A)*, denotes that the current phrase acts as the subject of the predicate phrase with the POS-tag *NR* (proper nouns). Here, "*(A)*" means that the phrase is an argument of the extracted relation. The second sub-pattern denotes that the predicate phrase of the example is "毕业 *(graduate)*". Note that the words between the predicate and arguments (e.g., *prep-于*) are included into the pattern directly (Nakashole et al., 2012; Mausam et al., 2012).

Definition 5 (semantic signature) The ***semantic signature*** of a relation consists of the semantic categories of the arguments. The semantic signature of Figure 1 is *(Af, Di)*, where *Af* and *Di* denotes *human* and *institute*, respectively.

Definition 6 (semantic pattern) A ***semantic pattern*** is the semantic abstraction of a relation. It is the combination of a syntactic pattern and a semantic signature. For instance, the syntactic pattern {*nsubj-NR(A) Pred[毕业] prep-于 pobj-NN(A)*}, combined with the semantic signature *(Af, Di)*, results in the semantic pattern {*nsubj-NR(Af) Pred[毕业] prep-于 pobj-NN(Di)*}.

[1]https://sourceforge.net/projects/zore/

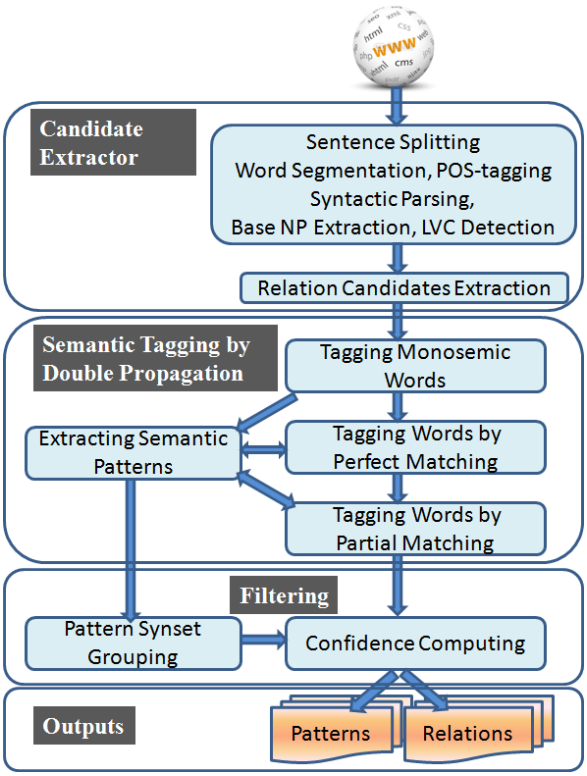


Figure 2: Architecture of ZORE.

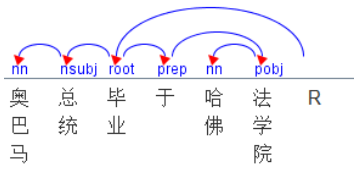


Figure 3: Parsing result of the example sentence in Figure 1, in Stanford dependencies.

3 ZORE

The architecture of ZORE is shown in Figure 2. It consists of three components. The first is a relation candidate extractor, which consumes input text and performs sentence segmentation, word segmentation, POS tagging, syntactic parsing, base NP extraction, light verb structure (LVC) detection and relation candidate extraction. The output is a set of relation candidates. The second component tags relations and extracts semantic patterns by a double propagation algorithm. In the third component, extracted patterns are grouped into synsets, and relations are filtered by confidence scores.

3.1 Extracting Relation Candidates

3.1.1 Parsing and Base NP Extraction

ZORE analyzes the syntactic structures of input texts by applying a pipeline of NLP tools. Each sentence is segmented into a list of words by using the Stanford segmenter (Chang et al., 2008), and parsed by using ZPar (Zhang and Clark, 2011), with POS tags and constituent structures by the CTB standard (Xue et al., 2005). The resulting constituent trees are transformed into projective trees with Stanford dependencies by using the Stanford parser (Chang et al., 2009). Figure 4 shows the parse tree of the sentence in Figure 1.

Next, base noun phrases (NPs) are extracted from the dependency tree. Here a base NP is a maximum phrase whose words can only have POS from the first row of Table 1. The head word of a base NP can be either a noun, a pronoun, a number or a measure word (the second row of Table 1). The dependency labels within a base NP can only be from the third row of Table 1. Obviously, a base NP does not contain other base NPs, and is also not contained by any other base NP.

3.1.2 Detecting Light Verb Constructions

In linguistics, a *light verb* is a verb that has little semantic content of its own, and typically forms a predicate with a noun (Butt, 2003). Example predicates by light verb constructions (LVC) include "*is a capital of*" and "*claim responsibility for*", where "is" and "claim" are light verbs. Improper handling of LVC can cause a significant problem by uninformative extractions (Etzioni et al., 2011). For example, if "is" and "claim" are extracted as predicates, the resulting relations (such as (*Hamas, claimed, responsibility*) from the sentence "Hamas claimed responsibility for the Gaza attack") might not bare useful information. ReVerb (Etzioni et al., 2011) handles this problem by hard syntactic constraints, taking the noun phrase (e.g., responsibility) between a verb phrase (e.g., "claim") and a preposition (e.g., "for") as a part of the predicate phrase rather than an argument, leading to the relation *(Hamas, claimed responsibility for, the Gaza attack)*.

In Chinese, LVCs are highly frequent and should be handled properly in order to ensure that the extracted relations are informative. However, the syntactic constraints in ReVerb can not be transferred to Chinese directly, because the word orders of English and Chinese are quite different.

	Labels
Base NP modifier	*NN (common noun), M (measure word), CD (cardinal number), OD (ordinal number), PN (pronoun), NR (proper noun), NT (temporal noun), JJ (other noun-modifier), or PU (punctuation)*
Base NP head	*NN (common noun), M (measure word), CD (cardinal number), OD (ordinal number), PN (pronoun), NR (proper noun), NT (temporal noun)*
Labels in base NPs	*nn (noun compound modifier), conj (conjunct), nummod (number modifier), cc (coordinating conjunction), clf (classifier modifier), det (determiner), ordmod (ordinal number modifier), punct (punctuation), dep (other dependencies), or amod (adjectival modifier)*
Labels from base NPs to predicate phrase	*nsubj (nominal subject), conj (conjunct), dobj (direct object), advmod (adverbial modifier), prep (prepositional modifier), pobj (prepositional object), lobj (localizer object), range (dative object that is a quantifier phrase), tmod (temporal modifier), plmod (localizer modifier of a preposition), attr (attributive), loc (localizer), top (topic), xsubj (controlling subject), ba ("ba" construction), nsubjpass (nominal passive subject)*

Table 1: Constraints on POS-tags and dependency labels. Labels in the top three rows are used for base NP extraction, while labels in the last row for traversing from a base NP to the predicate phrase.

In Chinese, prepositions acting as the modifier of a verb can be on both the left and right of the verb. For instance, the sentence "奥巴马 *(Obama)* 总统 *(President)* 于 *(from)* 哈佛 *(Harvard)* 法学院 *(Law School)* 毕业 *(graduate)*" is a paraphrase of the sentence in Figure 1, with the preposition 于 *(from)* on the left of the predicate phrase.

Chinese LVCs can be classified into two types, which we refer to as dummy-LVCs and common LVCs, respectively. For the first type, the predicate is a dummy verb such as "进行 *(do)*" and "予以 *(give)*", which has a noun phrase as its object. Since dummy verbs in Chinese are a closed set, we detect this type of LVCs (such as *"进行 (do) 会谈 (talk)"*) by finding the dummy verb from a lexicon. For the second type of LVCs, the predicate is a common verb, which has a nominalized structure or a common noun as its object. For instance, "展开 *(launch)* 调查 *(investigation)*" belongs to this type of construction.

Common LVCs are more difficult to detect than dummy-LVCs. We detect common LVCs by the context. Besides the NPs in the LVC itself, a common LVC typically governs two NPs, with the latter being connected to the predicate phrase by an LVC-related preposition such as "对 (for), 对于 (for), 针对 (for), 向 (to), 同 (with), 与 (with), 和 (with) ". Based on the observation, a basic idea of identifying common LVCs is to find verb-object structures that frequently co-occur with a LVC-related preposition in a large-scale corpus parsed automatically. For a given verb-object v, let f^v and f^p denote the frequency of v and the frequency of v co-occurring with an LVC-related preposition, respectively. We define the statistical strength of v to be an LVC as the ratio f^p/f^v. If the statistical strength of v exceeds a threshold t^{lvc}, we identify v as a LVC. Table 2 illustrates some high-frequency LVCs extracted by the method automatically.

3.1.3 Extracting Relation Candidates

ZORE tries to extract relation candidates from sentences that contain two or more base NPs. Given two base NPs, we traverse the dependency tree to obtain the shortest path that connects them. The path can contain only dependency labels in the fourth row of Table 1, and should contain at least one of the labels from "nsubj" and "dobj" to ensure that a predicate phrase is included in the path. If such a path is acquired, other base NPs governed by the same predicate phrase are included into the target relation, resulting in a n-ary relation candidates with each base NP being an argument. According to the predicate phrase, relation candidates can be classified into the following classes.

Common and dummy LVC relations. In this type of relations, the predicate phrase of the path is an LVC (e.g., a light verb and a nominal object). The two base NPs can be the subject or prepositional object of the light verb. For instance, in the sentence "霍迪尼 *(Houdini)* 对 *(to)* 我的 *(my)* 事业 *(career)* 有 *(have)* 很大 *(big)* 影响 *(influence)*", "有 *(have)*"and "影响 *(influence)*" are combined into a common LVC and taken as the predicate phrase, resulting the relation *(霍迪尼 (Houdini), Pred[有 (have) 影响 (influence)], 我的 (my) 事业 (career))*. In the corresponding English sentence, the predicate phrase "*be a big influence in*" is also an LVC structure.

Verb relations. In this type of relations, a verb acts as the predicate phrase. For instance, the relation *(奥巴马 (Obama) 总统 (President), Pred[毕业 (graduate)], 哈佛 (Harvard) 法学院 (Law School))* extracted from the sentence in Figure 1 is a typical verb relation.

Relative-clause relations. In an relative-clause

Verb	Noun
进行 (do) (*)	发行 (distribution), 分析 (analysis), 收集 (collection), 修改 (modification), 访问 (visit), 处罚 (punishment)
有 (have) (*)	影响(effect), 贡献 (contribution), 兴趣 (interest), 帮助 (help), 认识 (understanding), 期望 (expectation)
产生 (generate) (**)	影响 (effect), 兴趣 (interest), 怀疑(doubt), 冲击 (shock), 好感 (good feeling), 恐惧 (fear)
造成 (cause) (**)	影响 (effect), 破坏(destruction), 伤害 (harm), 威胁 (threat), 压力 (pressure), 干扰 (distraction)
表示 (express) (**)	满意 (satisfaction), 欢迎 (welcome), 尊重 (respect), 担忧 (worry), 哀悼 (mourning), 感谢 (gratitude)
展开 (launch) (**)	调查 (investigation), 攻击 (attack),攻势 (offensive), 批评 (criticism), 批判 (negotiation), 诉讼 (lawsuit)

Table 2: Instances of dummy-LVCs (*) and common LVCs (**). A verb in the left column is combined with a noun in the right column to form an LVC, which serves as the predicate phrase.

relation, the head word is a noun, modified by an relative clause, but acting as an argument of the predicate of the relative clause semantically. The sentence "毕业 *(graduate)* 于 *(from)* 哈佛 *(Harvard)* 法学院 *(Law School)* 的 *(de, an auxiliary word)* 奥巴马 *(Obama)* 总统 *(president)*" is a paraphrase of the sentence in Figure 1, with the same predicate phrase and arguments. However, the relation extracted from this phrase is an relative-clause relation *(Pred[*毕业 *(graduate)],* 哈佛 *(Harvard)* 法学院 *(Law School),* 奥巴马 *(Obama)* 总统 *(president))*, which belongs to the same pattern synset as the relation of Figure 1.

3.2 Semantic Tagging by Double Propagation

The basic idea of our approach is to identify relations and patterns iteratively through semantically tagging the head words of arguments in relation candidates. Given a set of relation candidates and a semantic taxonomy, the propagation consists of three steps. In Step 1, monosemic arguments in candidate relations are tagged with a semantic category, such as *Af* and *Di*, to obtain semantic patterns. In Step 2 and Step 3, untagged ambiguous and unknown words are tagged by perfect matching and partial matching, respectively. In the end of each step, semantic patterns are generalized from extracted and tagged relations, and then used to help relation tagging in the next step. Because of the two-way information exchange, we call this method *double propagation*. The method can also be treated as similar to bootstrapping (Yangarber et al., 2000; Qiu et al., 2009).

3.2.1 Step 1: Tagging Monosemic Arguments

Each argument in a relation candidate is a base NP. Since base NPs are endocentric, we can take the semantic category of the head word of a base NP as the semantic category of the base NP. In a taxonomy, each word is associated with one or more semantic categories. In this step, however, only monosemic words are tagged, while both ambiguous words and unknown words are left untagged.

Most named entities are not included in the taxonomy. However, after POS-tagging, most of them are detected as NR (proper noun). As a result, they are taken as ambiguous words that can be person names, organization names or location names. The named entities that are not included in the taxonomy are tagged in Steps 2 and 3.

After this step, all the arguments in some relation candidates have been tagged with semantic categories. We refer to these relation candidates as *tagged relation candidates*, and the remaining relation candidates as *untagged relation candidates*. Tagged relation candidate are generalized into semantic patterns, consisting of syntactic patterns and semantic signatures, as illustrated in Figure 1 and Section 2. We call the set of resulting semantic patterns Set^{SemPat}.

3.2.2 Step 2: Tagging by Perfect Pattern Matching

In this step, the arguments in the untagged relation candidates are tagged by semantic pattern matching. Given an untagged relation candidate *r*, we acquire a set of *possible* semantic categories for each argument with an *ambiguous* head word. For the arguments with *unknown* head words, we acquire a set of *possible* semantic categories according to their characters. Qiu et al. (2011) demonstrate that 98% Chinese words have at least one synonym, which shares at least one character. For Chinese nouns, the set of synonyms usually shares the last one or two characters. According to this, our strategy for acquiring possible semantic categories for an unknown word is as follows.

Given an unknown word w^u, if we find a known word w^k that shares the last two character with w^u, the semantic categories of w^k will be used as the possible semantic categories of w^u. Otherwise, if we find a known word w^k that share the last one character with w^u, the semantic categories of w^k will be used as the possible categories of w^u.

We then acquire possible semantic signatures of untagged relation candidates, of which all the arguments are tagged with possible semantic categories. As in Step 1, we generalize relation *r* into a syntactic pattern pat^{syn}, and then combine pat^{syn} with each *possible* semantic signature of *r* to generate *possible* semantic patterns. In case one or more possible semantic patterns of *r* exist in Set^{SemPat}, if the highest frequency of these patterns is above a threshold t^{sem}, the corresponding pattern will be taken as the semantic pattern of *r*, from which we infer the semantic signature for *r* and then the semantic category for the head word of each argument of *r*. After this step, the frequency of each semantic pattern in Set^{SemPat} is updated according to the newly tagged relation candidates.

3.2.3 Step 3: Tagging by Partial Pattern Matching

In this step, we tag the ambiguous and unknown words by partial matching rather than perfect matching of the whole semantic pattern. This can be treated as a back-off of the last step.

We first split *n*-ary semantic patterns in Set^{SemPat} into binary semantic patterns, and calculate their frequencies. Second, we split each untagged relation candidate *r* into several binary sub-relations and then search for the corresponding semantic patterns as in Step 2 — for each binary sub-relation, we obtain a binary semantic signature with the highest frequency. By combining the binary semantic signatures, we obtain one *n*-ary semantic signature for *r*, based on which all the unknown and ambiguous words can be tagged with a semantic categories. If all the arguments of a relation candidate *r* are tagged, *r* is treated as tagged. Finally, according to the newly tagged relations, statistics in Set^{SemPat} are updated.

3.3 Grouping Patterns into Synsets

In this step, we group semantic patterns from Set^{SemPat} into *pattern synsets*, based on a single-pass clustering process (Papka and Allan, 1998). Given two semantic patterns $SemPat_i$ and $SemPat_j$, we refer to their corresponding syntactic pattern, semantic signature and predicate phrases as $SynPat_i$ and $SynPat_j$, $SemSig_i$ and $SemSig_j$, $Pred_i$ and $Pred_j$, respectively. Not taking the predicate phrase into account, $SynPat_i$ and $SynPat_j$ are identical, and we call them *loosely identical* ($\approx$).

The algorithm in Figure 4 is used to group

if $Pred_i$ = "是 (is)" **or** $Pred_j$ = "是 (is)": **return** *false*.
else if ARGCOUNT($SemPat_i$) = 2 **and** ARGCOUNT($SemPat_j$) = 2 **and** SEMCAT(arg_1) = SEMCAT(arg_2):
  if $SynPat_i \approx SynPat_j$ **and** ISSONONYM ($Pred_i$, $Pred_j$): **return** *true*.
  else if $Pred_i = Pred_j$: **return** *true*.
  else: **return** *false*.
else if $Pred_i = Pred_j$ **and** $SemSig_i = SemSig_j$: **return** *true*.
else if ISSYNONYM($Pred_i$, $Pred_j$) **and** $SemSig_i = SemSig_j$ **and** $SynPat_i \approx SynPat_j$: **return** *true*.
else: **return** *false*.

Figure 4: Algorithm for pattern synset grouping.

Type	Feature	Weight
Base	*r* covers all words in *c*	0.96
Base	There are commas within *r*	-0.47
Base	LENGTH(*r*)<10 words	0.35
Base	10 words≤LENGTH(*r*)<20 words	0.11
Base	20 words≤LENGTH(*r*)	-1.06
Base	COUNT(*arguments*)=2	0.14
Base	COUNT(*arguments*)=3	0.33
Base	COUNT(*arguments*)=4	-0.60
Base	COUNT(*arguments*)> 4	-0.46
SemPat	Being tagged in Step 3	0.87
SemPat	Being tagged before Step 3	0.75
SemPat	50≤SIZE(*SemPat*) and untagged	-0.05
SemPat	50≤SIZE(*SemPat*) and tagged	0.65
SemPat	10≤SIZE(*SemPat*)<50 and untagged	-0.16
SemPat	10≤SIZE(*SemPat*)<50 and tagged	0.39
SemPat	5≤SIZE(*SemPat*)<10 and untagged	-0.22
SemPat	5≤SIZE(*SemPat*)<10 and tagged	0.36
SemPat	SIZE(*SemPat*) <5 and untagged	-0.92
SemPat	SIZE(*SemPat*) <5 and tagged	-0.64

Table 3: Features of the logistic regression classifier with weights trained on Wiki-500 dataset.

patterns, where ARGCOUNT($SynPat_j$) denotes the number of arguments in $SemPat_i$, SEMCAT(arg_1) indicates the semantic category of the first argument, and ISSYNONYM ($Pred_i$, $Pred_j$) returns whether two predicates are synonyms. In similarity-based single-pass clustering, the topic excursion problem is common (Papka and Allan, 1998). But since our similarity measure is symmetric, we do not suffer from this problem.

3.4 Computing the Confidence for Relations

Without filtering, the extraction algorithm in the previous sections may yield false relations. Following previous ORE systems, we make a balance between recall and precision by using a confidence threshold (Fader et al., 2011). A logistic regression classifier is used to give a confidence score to each relation, with features shown in Table 3. In the table, *c*, *r*, *arguments* and *SemPat* denote

Dataset	Source	#Sen	#Rel
Wiki-500	Chinese Wikipedia	500	561
Sina-500	Sina News	500	707

Table 4: Annotated relation datasets.

clause, relation, arguments in a relation, and semantic pattern, respectively. LENGTH(r), COUNT(*arguments*) and SIZE(*SemPat*) indicate the number of words in r, the number of arguments in r, and the number of relations that belong to the same semantic pattern *SemPat* as r. Because semantic patterns from the double propagation algorithm are used as features in the classifier, they participate in relation extraction also. Their effect on relation extraction can directly demonstrate the effectiveness of double propagation.

4 Experiments

4.1 Experimental Setup

We run ZORE on two difference corpora: the Chinese edition of Wikipedia (Wiki), which contains 4.3 million sentences (as of March 29, 2014), and a corpus from the Sina News archive (Sina News), which includes 6.1 million sentences from January 2013 to May 2013. The sentences that do not end with punctuations are filtered. The Chinese taxonomy *Extended Cilin*[2] (*Cilin*) (Che et al., 2010) is used to give semantic categories for each word. *Cilin* contains 77,492 Chinese words, organized into a five-level hierarchy. There are 12 categories in the top level, 94 in the second and 1492 in the third. In this paper, the second level is used for semantic categories. We create two test sets, containing 500 sentences from Wiki and 500 sentences from Sina News, respectively (see Table 4), annotated by two independent annotators using the annotation strategy of Fader et al. (2011). The thresholds t^{lvc} and t^{sem} for pattern matching are set as 0.4 and 5, tuned on 100 sentences from Wiki-500 dataset, respectively.

4.2 Evaluation of Relation Extraction

First, we compare ZORE with a baseline system to illustrate the effectiveness of the double propagation algorithm. The baseline system does not have the double propagation tagging component in Figure 2, using the logistic regression classifier in Section 3.4.1 with the 9 base features to filter extracted relation candidates. It is similar to the architecture of ReVerb (Fader et al., 2011). We

[2]http://ir.hit.edu.cn/demo/ltp/Sharing_Plan.htm

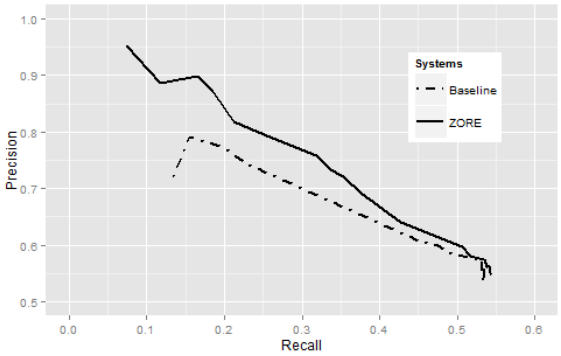


Figure 5: Performance on Wiki.

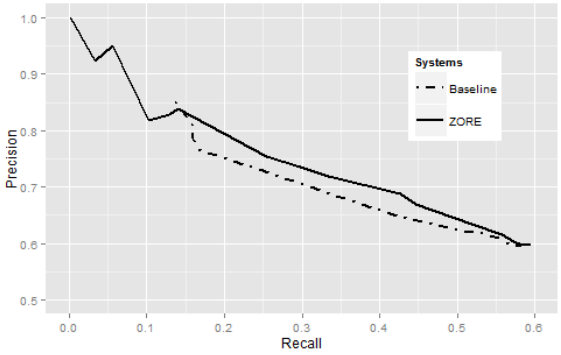


Figure 6: Performance on Sina News.

measure the precision and recall of the extracted relations. An extracted relation is considered correct only when the predicate phrase and all the arguments match the the gold set. On each data set, we perform 5-fold cross-validation test and take the average as the final precision and recall.

Figures 5 and 6 show the comparison of the two systems on Wiki and Sina News, respectively. On Wiki, ZORE has higher precision than the baseline at all levels of recall. When the recall is 0.3, the precision of ZORE is 0.77, 0.11 higher than the baseline. The result on Sina News is similar. The second column of Table 3 shows the weights of all features trained on the Wiki data set, which indicates that the semantic pattern features can give a positive effect on relation filtering.

Second, we compare the intermediate results at Steps 1, 2, and 3 in Section 3.2, respectively. The precision, recall and F1 of the three steps with different numbers of Wiki sentences (from 10K to 5M sentences) are shown in Table 5. This figure shows that Step 2 achieves higher precision than Step 1 at all levels of recall, indicating that the word sense tagging method in step 2 is useful for

Sentences	Step 1			Step 2			Step 3		
	P	R	F1	P	R	F1	P	R	F1
10K	0.947	0.032	0.062	**0.960**	0.043	0.082	0.933	0.075	0.139
50K	0.894	0.075	0.138	0.922	0.105	0.189	0.907	0.139	0.241
100K	0.897	0.093	0.169	0.924	0.130	0.228	0.909	0.160	0.272
200K	0.901	0.114	0.202	0.926	0.157	0.268	0.892	0.191	0.315
500K	0.891	0.146	0.251	0.909	0.196	0.322	0.860	0.230	0.363
1M	0.860	0.164	0.275	0.885	0.219	0.351	0.842	0.248	0.383
2M	0.797	0.182	0.296	0.819	0.250	0.383	0.788	0.278	0.411
3M	0.784	0.187	0.302	0.802	0.253	0.385	0.778	0.282	0.414
4M	0.739	0.178	0.287	0.801	0.258	0.390	0.778	0.287	0.419
5M	0.779	0.189	0.304	0.798	0.260	0.392	**0.768**	**0.289**	**0.420**

Table 5: Accuracies on different numbers Wiki sentences.

a significant boost of recall, together with a little improvement in precision. In particular, Step 2 can extract about 20% relations with relatively high precision (about 90%). The result of Step 3 is better to that of Step 2 in terms of F1-measure, with the highest F1-measure achieved by this step.

4.3 Evaluation of Patterns

ZORE acquires 122K and 222K patterns from Wiki and Sina News, clustered into 59K and 118K pattern synsets, respectively. The frequency distribution of the Wiki patterns is shown in Figure 7, which conforms to Zipf's law.

To assess the accuracy of pattern extraction, we rank the extracted patterns by the size, and evaluated the precision of the top 100 and a random set of 100 pattern synsets. Two annotators were shown a pattern synset with its semantic signature and a few example relations, and then asked to judge whether it indicates a valid semantic relation or not. The results are shown in Table 6. The averaged precision is 92% for the top 100 set, and 85% for the random 100 set.

The patterns in a pattern synset can be taken as paraphrases (Barzilay and Lee, 2003). We observe that two synonymous patterns might differ in three aspects. First, two patterns can differ by the predicates, which are synonyms. For instance, the verbs "担任, 当, 任, 出任, 为, 做" are synonyms, meaning "to hold the appointment of". Second, two patterns in the same synset can belong to different syntactic patterns, and therefore are paraphrases in the syntactic level. For instance, the semantic patterns of the two sentences "毕业 *(graduate)* 于 *(from)* 哈佛 *(Harvard)* 法学院 *(Law School)* 的 *(de, an auxiliary word)* 奥巴马 *(Obama)* 总统 *(president)*" and "奥巴马 *(Obama)* 总统 *(president)* 从*(from)* 哈佛 *(Harvard)* 法学院 *(Law School)* 毕业 *(graduate)*" are both synonymous to that of the sentence in Figure 1; all the three patterns are found in the same synset obtained by ZORE. Third, two patterns can differ only by the POS-tag. For instance, "奥巴马 *(Obama)* 从*(from)* 哈佛 *(Harvard)* 法学院 *(Law School)* 毕业 *(graduate)*" and "那个 *(That)* 律师 *(attorney)* 从*(from)* 哈佛 *(Harvard)* 法学院 *(Law School)* 毕业 *(graduate)*" are synonyms with different POS-tags for the first argument (i.e. N-R and NN). According to the grouping algorithm in Section 3.3, all the three types of paraphrases are grouped in a pattern synset, which makes some synsets very large. The largest synset contains 110 patterns, while the top 100 synsets contain more than 20 patterns.

4.4 Error analysis

We analyze the incorrect extractions (precision loss) and missed correct relations (recall loss) returned by Step 2, running on 500K sentences. Table 7 summarizes the types of correct relations that are missed by ZORE. 40% missed relations are due to the minimum frequency constraint on semantic patterns, which is used for a balance between precision and recall. Another main source of failure is the incorrect identification of the predicate phrase due to parsing errors, which account for 37% of the total errors. Other sources of failures include redundant arguments and segmentation errors. Most redundant arguments are related to prepositions such as "按照 (according to)" and "根据 (on the basis of)". For instance, in the sentence "按照 (according to) 这个 (the) 观点 (point of view) ， (,) 根本 (fundamental) 问题 (problem) 是 (is)", an incorrect binary relation *(这个 (the) 观点 (point of view), 根本 (fundamental) 问题 (problem), Pred[是 (is)])* is extracted, because the prepositional object "这个 (the) 观点 (point of view)" is tagged as an argument of the predicate phrase "是 (is)".

Table 8 summarizes the major types of incor-

Corpus	Patterns	Synsets	Top100	Random100
Wiki	122,723	59,298	0.93	0.87
Sina	222,773	118,923	0.91	0.83

Table 6: Precision of pattern synsets.

40%	Relations filtered by semantic pattern constraint
37%	Could not identify correct predicates because of preprocessing errors
12%	Too many arguments because of parsing errors
11%	Segmentation and POS tagging errors

Table 7: Relations missed by ZORE.

rect relations, 56% of which were caused by parsing errors, and 34% of which were due to word segmentation and POS tagging errors. Although many errors have been filtered by ZORE, the biggest source of errors is still syntactic analysis, which is very important for high quality of ORE.

5 Related Work

English has been the major language on which ORE research has been conducted. Previous work on English ORE has evolved from shallow-syntactic (Banko et al., 2007; Fader et al., 2011; Merhav et al., 2012) to full-syntactic (Nakashole et al., 2012; Mausam et al., 2012; Moro and Navigli, 2013; Xu et al., 2013) and semantic (Johansson and Nugues, 2008) systems.

It has been shown that a full-syntactic system based on dependency grammar can give significantly better results than shallow syntactic systems based on surface POS-patterns, yet enjoy higher efficiency compared with semantic systems (Mesquita et al., 2013). Our investigation on Chinese ORE takes root in full dependency syntax and is hence able to identify patterns that involve long-range dependencies. Considering the characteristics of the Chinese language, such as the lack of morphology and function words, and the high segmentation and word sense ambiguities, we incorporate semantic ontology information into the design of the system to improve the output quality without sacrificing efficiency.

The state-of-the-art systems most closely related to our approach are PATTY (Nakashole et al., 2012) and the system of Moro and Navigli (2013). Both, however, extract relations first, and then defines patterns based on extracted relations. This paper differs in that patterns and relations are extracted in a simultaneous process and so they can improve each other. Previous studies show that pattern generalization benefit from relation extrac-

56%	Parsing errors
17%	Segmentation errors
17%	POS tagging errors
6%	Redundant arguments
6%	Other, including base NP extraction errors

Table 8: Incorrect extractions by ZORE.

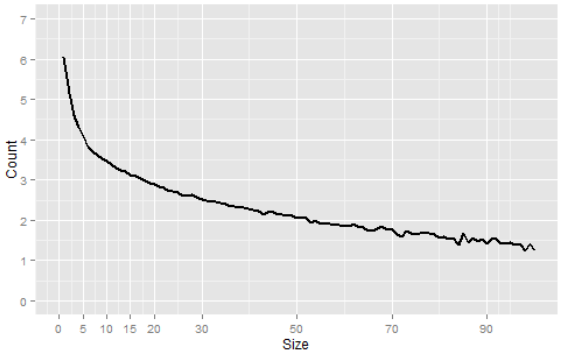


Figure 7: The frequency distribution of patterns extracted from Wiki. Size and Count denote the number of relations that belong to a semantic pattern and the logarithmic number of semantic patterns that have the same size, respectively.

tion (Nakashole et al., 2012; Moro and Navigli, 2013), and relation extraction can benefit from pattern generalization (Mausam et al., 2012). By using double propagation, not only can we make relation and pattern extraction benefit from each other, but we can also tag relations and patterns with semantic categories in a joint process.

There has been a line of research on Chinese relation extraction, where both feature-based (Zhou et al., 2005; Li et al., 2008) and kernel-based (Zhang et al., 2006; Che et al., 2005) methods have been applied. In addition, semantic ontologies such as *Extended Cilin* have been shown useful for Chinese relation extraction (Liu et al., 2013). However, these studies have focused on traditional IE, with pre-defined relations. In contrast, we investigate ORE for Chinese, finding that semantic ontologies useful for this task also. Tseng et al. (2014) is the only previous research focusing on Chinese ORE. Their system can be considered as a pipeline of word segmentation, POS-tagging and parsing, while our work gives semantic interpretation and explicitly deals with statistical errors in parsing by a novel double propagation algorithm between patterns and relations.

6 Conclusion and Future Work

We presented a Chinese ORE system that integrates relation extraction with semantic pattern generalization by double propagation. Experimental results on two datasets demonstrated the effectiveness of the proposed algorithm. We make the ZORE system, together with the large scale relations and pattern synsets extracted by ZORE, freely available at (`https://sourceforge.net/projects/zore/`). Another version of ZORE (ZORE-PMT), which is based on the dependency tagset from PMT1.0 (Qiu et al., 2014), is also provided.

Our error analysis demonstrates that the quality of syntactic parsing is crucial to the accuracy of syntax-based Chinese ORE. Improvements to syntactic analysis is likely to lead to improved ORE. In addition, the idea of double propagation can be generalized into information propagation between relation extraction and syntactic analysis. We plan to investigate the use of ORE in improving syntactic analysis in future work.

Acknowledgments

We gratefully acknowledge the invaluable assistance of Ji Ma, Wanxiang Che and Yijia Liu. We also thank the anonymous reviewers for their constructive comments, and gratefully acknowledge the support of the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301, the start-up grant SRG ISTD 2012 038 from Singapore University of Technology and Design, the National Natural Science Foundation of China (No. 61103089), National High Technology Research and Development Program of China (863 Program) (No. 2012AA011101), Major National Social Science Fund of China (No. 12&ZD227), Scientific Research Foundation of Shandong Province Outstanding Young Scientist Award (No. BS2013DX020) and Humanities and Social Science Projects of Ludong University (No. WY2013003).

References

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction for the web. In *IJCAI*, volume 7, pages 2670–2676.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 16–23. Association for Computational Linguistics.

Miriam Butt. 2003. The light verb jungle. In *Workshop on Multi-Verb Constructions*, pages 1–28.

Pi-Chuan Chang, Michel Galley, and Christopher D Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 224–232. Association for Computational Linguistics.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D Manning. 2009. Discriminative reordering with Chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*, pages 51–59. Association for Computational Linguistics.

WX Che, Jianmin Jiang, Zhong Su, Yue Pan, and Ting Liu. 2005. Improved-edit-distance kernel for Chinese relation extraction. In *IJCNLP*, pages 132–137.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A Chinese language technology platform. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 13–16. Association for Computational Linguistics.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, pages 3–10. AAAI Press.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of propbank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 69–78. Association for Computational Linguistics.

Jun-Tae Kim and Dan I Moldovan. 1993. Acquisition of semantic patterns for information extraction from corpora. In *Artificial Intelligence for Applications, 1993. Proceedings., Ninth Conference on*, pages 171–176. IEEE.

Wenjie Li, Peng Zhang, Furu Wei, Yuexian Hou, and Qin Lu. 2008. A novel feature-based approach to Chinese entity relation extraction. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 89–92. Association for Computational Linguistics.

Dandan Liu, Zhiwei Zhao, Yanan Hu, and Longhua Qian. 2013. Incorporating lexical semantic similarity to tree kernel-based Chinese relation extraction. In *Chinese Lexical Semantics*, pages 11–21. Springer.

Michael Schmitz Mausam, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. pages 523–534.

Yuval Merhav, Filipe Mesquita, Denilson Barbosa, Wai Gen Yee, and Ophir Frieder. 2012. Extracting information networks from the blogosphere. *ACM Transactions on the Web (TWEB)*, 6(3):11.

Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and efficiency of open relation extraction. *New York Times*, 500:150.

Andrea Moro and Roberto Navigli. 2012. Wisenet: Building a wikipedia-based semantic network with ontologized relations. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1672–1676. ACM.

Andrea Moro and Roberto Navigli. 2013. Integrating syntactic and semantic analysis into the open information extraction paradigm. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2148–2154. AAAI Press.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: a taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145. Association for Computational Linguistics.

Ron Papka and James Allan. 1998. On-line new event detection using single pass clustering. *University of Massachusetts, Amherst*.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2009. Expanding domain sentiment lexicon through double propagation. In *IJCAI*, volume 9, pages 1199–1204.

Likun Qiu, Yunfang Wu, and Yanqiu Shao. 2011. Combining contextual and structural information for supersense tagging of Chinese unknown words. In *Computational Linguistics and Intelligent Text Processing*, pages 15–28. Springer.

Likun Qiu, Yue Zhang, Peng Jin, and Houfeng Wang. 2014. Multi-view Chinese treebanking. In *Proceedings of COLING 2014*, pages 257–268.

Alan Ritter, Oren Etzioni, et al. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434. Association for Computational Linguistics.

Yuen-Hsien Tseng, Lung-Hao Lee, Shu-Yen Lin, Bo-Shun Liao, Mei-Jun Liu, Hsin-Hsi Chen, Oren Etzioni, and Anthony Fader. 2014. Chinese open relation extraction for knowledge acquisition. In *EACL 2014*, pages 12–16.

Fei Wu and Daniel S Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127. Association for Computational Linguistics.

Ying Xu, Mi-Young Kim, Kevin Quinn, Randy Goebel, and Denilson Barbosa. 2013. Open information extraction with tree kernels. In *Proceedings of NAACL-HLT*, pages 868–877.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn Chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 940–946. Association for Computational Linguistics.

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

Min Zhang, Jie Zhang, Jian Su, and Guodong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 825–832. Association for Computational Linguistics.

GuoDong Zhou, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 427–434. Association for Computational Linguistics.

Coarse-grained Candidate Generation and Fine-grained Re-ranking for Chinese Abbreviation Prediction

Longkai Zhang Houfeng Wang Xu Sun
Key Laboratory of Computational Linguistics (Peking University)
Ministry of Education, China
zhlongk@qq.com, wanghf@pku.edu.cn, xusun@pku.edu.cn

Abstract

Correctly predicting abbreviations given the full forms is important in many natural language processing systems. In this paper we propose a two-stage method to find the corresponding abbreviation given its full form. We first use the contextual information given a large corpus to get abbreviation candidates for each full form and get a coarse-grained ranking through graph random walk. This coarse-grained rank list fixes the search space inside the top-ranked candidates. Then we use a similarity sensitive re-ranking strategy which can utilize the features of the candidates to give a fine-grained re-ranking and select the final result. Our method achieves good results and outperforms the state-of-the-art systems. One advantage of our method is that it only needs weak supervision and can get competitive results with fewer training data. The candidate generation and coarse-grained ranking is totally unsupervised. The re-ranking phase can use a very small amount of training data to get a reasonably good result.

1 Introduction

Abbreviation Prediction is defined as finding the meaningful short subsequence of characters given the original fully expanded form. As an example, "HMM" is the abbreviation for the corresponding full form "Hidden Markov Model". While the existence of abbreviations is a common linguistic phenomenon, it causes many problems like spelling variation (Nenadić et al., 2002). The different writing manners make it difficult to identify the terms conveying the same concept, which will hurt the performance of many applications, such as information retrieval (IR) systems and machine translation (MT) systems.

Previous works mainly treat the Chinese abbreviation generation task as a sequence labeling problem, which gives each character a label to indicate whether the given character in the full form should be kept in the abbreviation or not. These methods show acceptable results. However they rely heavily on the character-based features, which means it needs lots of training data to learn the weights of these context features. The performance is good on some test sets that are similar to the training data, however, when it moves to an unseen context, this method may fail. This is always true in real application contexts like the social media where there are tremendous new abbreviations burst out every day.

A more intuitive way is to find the full-abbreviation pairs directly from a large text corpus. A good source of texts is the news texts. In a news text, the full forms are often mentioned first. Then in the rest of the news its corresponding abbreviation is mentioned as an alternative. The co-occurrence of the full form and the abbreviation makes it easier for us to mine the abbreviation pairs from the large amount of news texts. Therefore, given a long full form, we can generate its abbreviation candidates from the given corpus, instead of doing the character tagging job.

For the abbreviation prediction task, the candidate abbreviation must be a sub-sequence of the given full form. An intuitive way is to select all the sub-sequences in the corpus as the candidates. This will generate large numbers of irrelevant candidates. Instead, we use a contextual graph random walk method, which can utilize the contextual information through the graph, to select a coarse grained list of candidates given the full form. We only select the top-ranked candidates to reduce the search space. On the other hand, the candidate generation process can only use limited contextual information to give a coarse-grained ranked list of candidates. During generation, can-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1881–1890,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

didate level features cannot be included. Therefore we propose a similarity sensitive re-ranking method to give a fine-grained ranked list. We then select the final result based on the rank of each candidate.

The contribution of our work is two folds. Firstly we propose an improved method for abbreviation generation. Compared to previous work, our method can perform well with less training data. This is an advantage in the context of social media. Secondly, we build a new abbreviation corpus and make it publicly available for future research on this topic.

The paper is structured as follows. Section 1 gives the introduction. In section 2 we describe the abbreviation task. In section 3 we describe the candidate generation part and in section 4 we describe the re-ranking part. Experiments are described in section 5. We also give a detailed analysis of the results in section 5. In section 6 related works are introduced, and the paper is concluded in the last section.

2 Chinese Abbreviation Prediction System

Chinese Abbreviation Prediction is the task of selecting representative characters from the long full form[1]. Previous works mainly use the sequence labeling strategies, which views the full form as a character sequence and give each character an extra label 'Keep' or 'Skip' to indicate whether the current character should be kept in the abbreviation. An example is shown in Table 1. The sequence labeling method assumes that the character context information is crucial to decide the keep or skip of a character. However, we can give many counterexamples. An example is "北京大学"(Peking University) and "清华大学"(Tsinghua University), whose abbreviations correspond to "北大" and '清华' respectively. Although sharing a similar character context, the third character '大' is kept in the first case and is skipped in the second case.

We believe that a better way is to extract these abbreviation-full pairs from a natural text corpus where the full form and its abbreviation co-exist. Therefore we propose a two stage method. The first stage generates a list of candidates given a large corpus. To reduce the search space, we adopt graph random walk to give a coarse-grained ranking and select the top-ranked ones as the candidates. Then we use a similarity sensitive re-ranking method to decide the final result. Detailed description of the two parts is shown in the following sections.

[1]Details of the difference between English and Chinese abbreviation prediction can be found in Zhang et al. (2012).

Full form	香	港	大	学
Status	Skip	Keep	Keep	Skip
Result		港	大	

Table 1: The abbreviation "港大" of the full form "香港大学" (Hong Kong University)

3 Candidate Generation through Graph Random Walk

3.1 Candidate Generation and Graph Representation

Chinese abbreviations are sub-sequences of the full form. We use a brute force method to select all strings in a given news article that is the sub-sequence of the full form. The brute force method is not time consuming compared to using more complex data structures like trie tree, because in a given news article there are a limited number of sub-strings which meet the sub-sequence criteria for abbreviations. When generating abbreviation candidates for a given full form, we require the full form should appear in the given news article at least once. This is a coarse filter to indicate that the given news article is related to the full form and therefore the candidates generated are potentially meaningful.

The main motivation of the candidate generation stage in our approach is that the full form and its abbreviation tend to share similar context in a given corpus. To be more detailed, given a word context window w, the words that appear in the context window of the full form tend to be similar to those words in the context window of the abbreviations.

We use a bipartite graph $G(V_{word}, V_{context}, E)$ to represent this phenomena. We build bipartite graphs for each full form individually. For a given full form v_{full}, we first extract all its candidate abbreviations V_C. We have two kinds of nodes in the bipartite graph: the word nodes and the context nodes. We construct the word nodes as $V_{word} = V_C \cup \{v_{full}\}$, which is the node set of the full form and all the candidates. We construct the context nodes $V_{context}$ as the words that appear

in a fixed window of V_{word}. To reduce the size of the graph, we make two extra assumptions: 1) We only consider the nouns and verbs in the context and 2) context words should appear in the vocabulary for more than a predefined threshold (i.e. 5 times). Because G is bipartite graph, the edges E only connect word node and context nodes. We use the number of co-occurrence of the candidate and the context word as the weight of each edge and then form the weight matrix W. Details of the bipartite graph construction algorithm are shown in Table 2. An example bipartite graph is shown in figure 1.

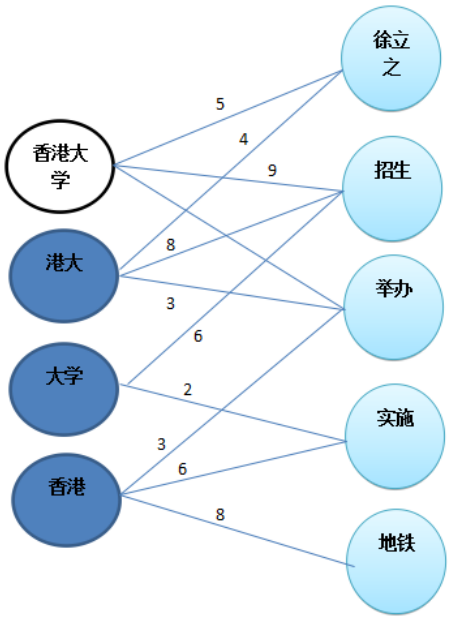


Figure 1: An example of the bipartite graph representation. The full form is "香港大学"(Hong Kong University), which is the first node on the left. The three candidates are "港大", "香港", "大学", which are the nodes on the left. The context words in this example are "徐立之"(Tsui Lap-chee, the headmaster of Hong Kong University), "招生"(Enrollment), "举办"(Hold), "实施"(Enact), "地铁"(Subway), which are the nodes on the right. The edge weight is the co-occurrence of the left word and the right word.

3.2 Coarse-grained Ranking Using Random Walks

We perform Markov Random Walk on the constructed bipartite graph to give a coarse-grained ranked list of all candidates. In random walk, a *walker* starts from the full form source node S (in later steps, v_i) and randomly *walks* to another node v_j with a transition probability p_{ij}. In random walk we assume the *walker* do the *walking* n times and finally stops at a final node. When the walking is done, we can get the probability of each node that the *walker* stops in the end. Because the *destination* of each step is selected based on transition probabilities, the word node that shares more similar contexts are more likely to be the final stop. The random walk method we use is similar to those defined in Norris (1998); Zhu et al. (2003); Sproat et al. (2006); Hassan and Menezes (2013); Li et al. (2013).

The transition probability p_{ij} is calculated using the weights in the weight matrix W and then normalized with respect to the source node v_i with the formula $p_{ij} = \frac{w_{ij}}{\sum_l w_{il}}$. When the graph random walk is done, we get a list of coarse-ranked candidates, each with a confidence score derived from the context information. By performing the graph random walk, we reduce the search space from exponential to the top-ranked ones. Now we only need to select the final result from the candidates, which we will describe in the next section.

4 Candidate Re-ranking

Although the coarse-grained ranked list can serve as a basic reference, it can only use limited information like co-occurrence. We still need a re-ranking process to decide the final result. The reason is that we cannot get any candidate-specific features when the candidate is not fully generated. Features such as the length of a candidate are proved to be useful to rank the candidates by previous work. In this section we describe our second stage for abbreviation generation, which we use a similarity sensitive re-ranking method to find the final result.

4.1 Similarity Sensitive Re-ranking

The basic idea behind our similarity sensitive re-ranking model is that we penalize the mistakes based on the similarity of the candidate and the reference. If the model wrongly selects a less similar candidate as the result, then we will attach a large penalty to this mistake. If the model wrongly chooses a candidate but the candidate is similar to the reference, we slightly penalize this mistake. The similarity between a candidate and the reference is measured through character similarity, which we will describe later.

Input: the full form v_{full}, news corpus U
Output: bipartite graph $G(V_{word}, V_{context}, E)$

Candidate vector $V_c = \emptyset$, $V_{context} = \emptyset$
for each document d in U
  if d contains v_{full}
  add all words w in the window of v_{full} into $V_{context}$
  for each n-gram s in d
  if s is a sub-sequence of v_{full}
  add s into V_c
  add all word w in the window of s into $V_{context}$
  end if
  end for
  end if
end for
$V_{word} = V_c \cup \{v_{full}\}$
for each word v_i in V_{word}
  for each word v_j in $V_{context}$
  calculate edge weight in E based on co-occurrence
  end for
end for
Return $G(V_{word}, V_{context}, E)$

Table 2: Algorithm for constructing bipartite graphs

We first give some notation of the re-ranking phase.

1. $f(x, y)$ is a scoring function for a given combination of x and y, where x is the original full form and y is an abbreviation candidate. For a given full form x_i with K candidates, we assume its corresponding K candidates are y_i^1,y_i^2,...,y_i^K.

2. evaluation function $s(x, y)$ is used to measure the similarity of the candidate to the reference, where x is the original full form and y is one abbreviation candidate. We require that $s(x, y)$ should be in $[0, 1]$ and $s(x, y) = 1$ if and only if y is the reference.

One choice for $s(x, y)$ may be the indicator function. However, indicator function returns zero for all false candidates. In the abbreviation prediction task, some false candidates are much closer to the reference than the rest. Considering this, we use a Longest Common Subsequence(LCS) based criterion to calculate $s(x, y)$. Suppose the length of a candidate is a, the length of the reference is b and the length of their LCS is c, then we can define precision P and recall R as:

$$\begin{aligned} P =& \frac{c}{a}, \\ R =& \frac{c}{b}, \\ F =& \frac{2 * P * R}{P + R} \end{aligned} \tag{1}$$

It is easy to see that F is a suitable $s(x, y)$. Therefore we can use the F-score as the value for $s(x, y)$.

3. $\phi(x, y)$ is a feature function which returns a m dimension feature vector. m is the number of features in the re-ranking.

4. $\vec{w}$ is a weight vector with dimension m. $\vec{w}^T\phi(x, y)$ is the score after re-ranking. The candidate with the highest score will be our final result.

Given these notations, we can now describe our re-ranking algorithm. Suppose we have the training set $X = \{x_1, x_2, ..., x_n\}$. We should find the weight vector $\vec{w}$ that can minimize the loss function:

$$\begin{aligned} Loss(\vec{w}) = \sum_{i=1}^{n} \sum_{j=1}^{k} ((s(x_i, y_i^1) - s(x_i, y_i^j)) \\ * I(\vec{w}^T\phi(x_i, y_i^j) \geq \vec{w}^T\phi(x_i, y_i^1))) \end{aligned} \tag{2}$$

$I(x)$ is the indicator function. It equals to 1 if and only if $x \geq 0$. $I(j) = 1$ means that the candidate which is less 'similar' to the reference is ranked higher than the reference. Intuitively, $Loss(\vec{w})$ is the weighted sum of all the wrongly ranked candidates.

It is difficult to optimize $Loss(\vec{w})$ because $Loss(\vec{w})$ is discontinuous. We make a relaxation here[2]:

$$\begin{aligned} L(\vec{w}) = & \sum_{i=1}^{n}\sum_{j=1}^{k}((s(x_i, y_i^1) - s(x_i, y_i^j)) \\ & * \frac{1}{1 + e^{-\vec{w}^T(\phi(x_i, y_i^j) - \phi(x_i, y_i^1))}}) \\ & \leq \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{k}((s(x_i, y_i^1) - s(x_i, y_i^j)) \\ & * I(\vec{w}^T\phi(x_i, y_i^j) \geq \vec{w}^T\phi(x_i, y_i^1))) \\ & = \frac{1}{2}Loss(\vec{w}) \end{aligned} \tag{3}$$

From the equations above we can see that $2L(\vec{w})$ is the upper bound of our loss function $Loss(\vec{w})$. Therefore we can optimize $L(\vec{w})$ to approximate $Loss(\vec{w})$.

We can use optimization methods like gradient descent to get the $\vec{w}$ that minimize the loss function. Because L is not convex, it may go into a local minimum. In our experiment we held out 10% data as the develop set and try random initialization to decide the initial $\vec{w}$.

4.2 Features for Re-ranking

One advantage of the re-ranking phase is that it can now use features related to candidates. Therefore, we can use a variety of features. We list them as follows.

1. **The coarse-grained ranking score from the graph random walk phase**. From the description of the previous section we know that this score is the probability a 'walker' 'walk' from the full form node to the current candidate. This is a coarse-grained score because it can only use the information of words inside the window. However, it is still informative because in the re-ranking phase we cannot collect this information directly.
2. **The character uni-grams and bi-grams in the candidate**. This kind of feature cannot be used in the traditional character tagging methods.
3. **The language model score of the candidate**. In our experiment, we train a bi-gram language model using Laplace smoothing on the Chinese Gigaword Data[3].
4. **The length of the candidate**. Intuitively, abbreviations tend to be short. Therefore length can be an important feature for the re-ranking.
5. **The degree of ambiguity of the candidate**. We first define the degree of ambiguity d_i of a character c_i as the number of identical words that contain the character. We then define the degree of ambiguity of the candidate as the sum of all d_i in the candidates. We need a dictionary to extract this feature. We collect all words in the PKU data of the second International Chinese Word Segmentation Bakeoff[4].
6. **Whether the candidate is in a word dictionary**. We use the PKU dictionary in feature 5.
7. **Whether all bi-grams are in a word dictionary**. We use the PKU dictionary in feature 5.
8. **Adjacent Variety(AV) of the candidate**. We define the left AV of the candidate as the probability that in a corpus the character in front of the candidate is a character in the full form. For example if we consider the full form "北京大学"(Peking University) and the candidate "京大", then the left AV of "京大" is the probability that the character preceding "京大" is '北' or '京' or '大' or '学' in a corpus. We can similarly define the right AV, with respect to characters follow the candidate.

The AV feature is very useful because in some cases a substring of the full form may have a confusingly high frequency. In the example of "北京大学"(Peking University), an article in the corpus may mention "北京大学"(Peking University) and

[2]To prove this we need the following two inequalities: 1) when $x \geq 0$, $I(x) \leq \frac{2}{1+e^{-x}}$ and 2) $s(x_i, y_i^1) - s(x_i, y_i^j) \geq 0$.

[3]http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2003T09

[4]http://www.sighan.org/bakeoff2005/

“东京大学”(Tokyo University) at the same time. Then the substring “京大学” may be included in the candidate generation phase for “北京大学” with a high frequency. Because the left AV of “京大学” is high, the re-ranker can easily detect this false candidate.

In practice, all the features need to be scaled in order to speed up training. There are many ways to scale features. We use the most intuitive scaling method. For a feature value x, we scale it as $(x-mean)/(max-min)$. Note that for language model score and the score of random walk phase, we scale based on their log value.

5 Experiments

5.1 Dataset and Evaluation metrics

For the dataset, we collect 3210 abbreviation pairs from the Chinese Gigaword corpus. The abbreviation pairs include noun phrases, organization names and some other types. The Chinese Gigaword corpus contains news texts from the year 1992 to 2007. We only collect those pairs whose full form and corresponding abbreviation appear in the same article for at least one time. For full forms with more than one reasonable reference, we keep the most frequently used one as its reference. We use 80% abbreviation pairs as the training data and the rest as the testing data.

We use the top-K accuracy as the evaluation metrics. The top-K accuracy is widely used as the measurement in previous work (Tsuruoka et al., 2005; Sun et al., 2008, 2009; Zhang et al., 2012). It measures what percentage of the reference abbreviations are found if we take the top k candidate abbreviations from all the results. In our experiment, we compare the top-5 accuracy with baselines. We choose the top-10 candidates from the graph random walk are considered in re-ranking phase and the measurement used is top-1 accuracy because the final aim of the algorithm is to detect the exact abbreviation, rather than a list of candidates.

5.2 Candidate List

Table 3 shows examples of the candidates. In our algorithm we further reduce the search space to only incorporate 10 candidates from the candidate generation phase.

K	Top-K Accuracy
1	6.84%
2	19.35%
3	49.01%
4	63.70%
5	73.60%

Table 4: Top-5 accuracy of the candidate generation phase

5.3 Comparison with baselines

We first show the top-5 accuracy of the candidate generation phase Table 4. We can see that, just like the case of using other feature alone, using the score of random walk alone is far from enough. However, the first 5 candidates contain most of the correct answers. We use the top-5 candidates plus another 5 candidates in the re-ranking phase.

We choose the character tagging method as the baseline method. The character tagging strategy is widely used in the abbreviation generation task (Tsuruoka et al., 2005; Sun et al., 2008, 2009; Zhang et al., 2012). We choose the ‘SK’ labeling strategy which is used in Sun et al. (2009); Zhang et al. (2012). The ‘SK’ labeling strategy gives each character a label in the character sequence, with ‘S’ represents ’Skip’ and ‘K’ represents ‘Keep’. Same with Zhang et al. (2012), we use the Conditional Random Fields (CRFs) model in the sequence labeling process.

The baseline method mainly uses the character context information to generate the candidate abbreviation. To be fair we use the same feature set in Sun et al. (2009); Zhang et al. (2012). One drawback of the sequence labeling method is that it relies heavily on the character context in the full form. With the number of new abbreviations grows rapidly (especially in social media like Facebook or twitter), it is impossible to let the model ‘remember’ all the character contexts. Our method is different from theirs, we use a more intuitive way which finds the list of candidates directly from a natural corpus.

Table 5 shows the comparison of the top-5 accuracy. We can see that our method outperforms the baseline methods. The baseline model performs well when using character features (Column 3). However, it performs poorly without the character features (Column 2). In contrast, without the character features, our method (Column 4) works much better than the sequence labeling method.

Full form	Reference	Generated Candidates	#Enum	#Now
国际政治系 (Department of International Politics)	国政系	国政系,政治系,国际政治,国政治,国政,政治,国际	30	7
无核武器国家 (Non-nuclear Countries)	无核国	核国,无核,核武,核国家,无核国,武器国,无核国家,核武器国,无核武器,核武器国家,核武器,国家,武器	62	13
贩卖毒品 (Drug trafficking)	贩毒	卖毒品,贩毒品,贩卖,毒品,贩毒	14	5
长江经济联合发展股份有限公司 (Yangtze Joint River Economic Development Inc.)	长发公司	合股,长发,长发公司,长江公司,长江经济,联合发展,经济发展,经济联合,长江联合发展,长江发展公司,长江经济联合,经济联合发展,长江经济联合发展,股份有限公司,有限公司,长江,公司,股份,联合,经济	16382	20

Table 3: Generated Candidates. #Enum is the number of candidates generated by enumerating all possible candidates. #Now is the number of candidates generated by our method.

When we add character features, our method (Column 5) still outperforms the sequence labeling method.

K	CRF-char	Our-char	CRF	Our
1	38.00%	**48.60%**	53.27%	**55.61%**
2	38.16%	70.87%	65.89%	73.10%
3	39.41%	81.78%	72.43%	81.96%
4	55.30%	87.54%	78.97%	87.57%
5	62.31%	89.25%	81.78%	89.27%

Table 5: Comparison of the baseline method and our method. CRF-char ('-' means minus) is the baseline method without character features. CRF is the baseline method. Our-char is our method without character features. We define character features as the features that consider the characters from the original full form as their parts.

5.4 Performance with less training data

One advantage of our method is that it only requires weak supervision. The baseline method needs plenty of manually collected full-abbreviation pairs to learn a good model. In our method, the candidate generation and coarse-grained ranking is totally unsupervised. The re-ranking phase needs training instances to decide the parameters. However we can use a very small amount of training data to get a reasonably good model. Figure 2 shows the result of using different size of training data. We can see that the performance of the baseline methods drops rapidly when there are less training data. In contrast, when using less training data, our method does not suffer that much.

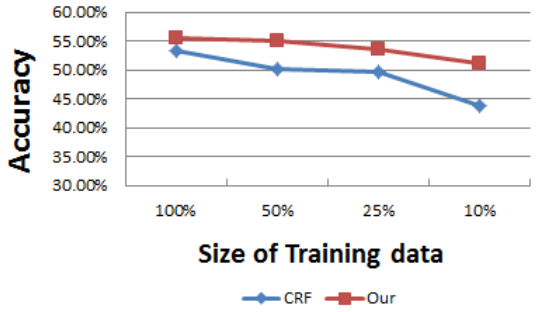


Figure 2: Top-1 accuracy when changing the size of training data. For example, "50%" means "using 50% of all the training data".

5.5 Comparison with previous work

We compare our method with the method in the previous work DPLVM+GI in Sun et al. (2009), which outperforms Tsuruoka et al. (2005); Sun et al. (2008). We also compare our method with the web-based method CRF+WEB in Zhang et al. (2012). Because the comparison is performed on different corpora, we run the two methods on our data. Table 6 shows the top-1 accuracy. We can see that our method outperforms the previous

methods.

System	Top-K Accuracy
DPLVM+GI	53.29%
CRF+WEB	54.02%
Our method	**55.61%**

Table 6: Comparison with previous work. The search results of CRF+WEB is based on March 9, 2014 version of the Baidu search engine.

5.6 Error Analysis

We perform cross-validation to find the errors and list the two major errors below:

1. Some full forms may correspond to more than one acceptable abbreviation. In this case, our method may choose the one that is indeed used as the full form's abbreviation in news texts, but not the same as the standard reference abbreviations. The reason for this phenomenon may lie in the fact that the verification data we use is news text, which tends to be formal. Therefore when a reference is often used colloquially, our method may miss it. We can relieve this by changing the corpus we use.

2. Our method may provide biased information when handling location sensitive phrases. Not only our system, the system of Sun et al. (2009); Zhang et al. (2012) also shows this phenomenon. An example is the case of "香港民主同盟" (Democracy League of Hong Kong). Because most of the news is about news in mainland China, it is hard for the model to tell the difference between the reference "港同盟" and a false candidate "民盟"(Democracy League of China).

   Another ambiguity is "清华大学"(Tsinghua University), which has two abbreviations "清大" and "清华". This happens because the full form itself is ambiguous. Word sense disambiguation can be performed first to handle this kind of problem.

6 Related Work

Abbreviation generation has been studied during recent years. At first, some approaches maintain a database of abbreviations and their corresponding "full form" pairs. The major problem of pure database-building approach is obvious. It is impossible to cover all abbreviations, and the building process is quite laborious. To find these pairs automatically, a powerful approach is to find the reference for a full form given the context, which is referred to as "abbreviation generation".

There is research on using heuristic rules for generating abbreviations Barrett and Grems (1960); Bourne and Ford (1961); Taghva and Gilbreth (1999); Park and Byrd (2001); Wren et al. (2002); Hearst (2003). Most of them achieved high performance. However, hand-crafted rules are time consuming to create, and it is not easy to transfer the knowledge of rules from one language to another.

Recent studies of abbreviation generation have focused on the use of machine learning techniques. Sun et al. (2008) proposed an SVM approach. Tsuruoka et al. (2005); Sun et al. (2009) formalized the process of abbreviation generation as a sequence labeling problem. The drawback of the sequence labeling strategies is that they rely heavily on the character features. This kind of method cannot fit the need for abbreviation generation in social media texts where the amount of abbreviations grows fast.

Besides these pure statistical approaches, there are also many approaches using Web as a corpus in machine learning approaches for generating abbreviations. Adar (2004) proposed methods to detect such pairs from biomedical documents. Jain et al. (2007) used web search results as well as search logs to find and rank abbreviates full pairs, which show good result. The disadvantage is that search log data is only available in a search engine backend. The ordinary approaches do not have access to search engine internals. Zhang et al. (2012) used web search engine information to rerank the candidate abbreviations generated by statistical approaches. Compared to their approaches, our method only uses a fixed corpus, instead of using collective information, which varies from time to time.

Some of the previous work that relate to abbreviations focuses on the task of "abbreviation disambiguation", which aims to find the correct abbreviation-full pairs. In these works, machine learning approaches are commonly used (Park and Byrd, 2001; HaCohen-Kerner et al., 2008; Yu et al., 2006; Ao and Takagi, 2005). We focus on another aspect. We want to find the abbreviation

given the full form. Besides, Sun et al. (2013) also works on abbreviation prediction but focuses on the negative full form problem, which is a little different from our work.

One related research field is text normalization, with many outstanding works (Sproat et al., 2001; Aw et al., 2006; Hassan and Menezes, 2013; Ling et al., 2013; Yang and Eisenstein, 2013). While the two tasks share similarities, abbreviation prediction has its identical characteristics, like the sub-sequence assumption. This results in different methods to tackle the two different problems.

7 Conclusion

In this paper, we propose a unified framework for Chinese abbreviation generation. Our approach contains two stages: candidate generation and re-ranking. Given a long term, we first generate a list of abbreviation candidates using the co-occurrence information. We give a coarse-grained rank using graph random walk to reduce the search space. After we get the candidate lists, we can use the features related to the candidates. We use a similarity sensitive re-rank method to get the final abbreviation. Experiments show that our method outperforms the previous systems.

Acknowledgments

This research was partly supported by National Natural Science Foundation of China (No.61370117,61333018,61300063),Major National Social Science Fund of China(No.12&ZD227), National High Technology Research and Development Program of China (863 Program) (No. 2012AA011101), and Doctoral Fund of Ministry of Education of China (No. 20130001120004). The contact author of this paper, according to the meaning given to this role by Key Laboratory of Computational Linguistics, Ministry of Education, School of Electronics Engineering and Computer Science, Peking University, is Houfeng Wang. We thank Ke Wu for part of our work is inspired by his previous work at KLCL.

References

Adar, E. (2004). Sarad: A simple and robust abbreviation dictionary. *Bioinformatics*, 20(4):527–533.

Ao, H. and Takagi, T. (2005). Alice: an algorithm to extract abbreviations from medline. *Journal of the American Medical Informatics Association*, 12(5):576–586.

Aw, A., Zhang, M., Xiao, J., and Su, J. (2006). A phrase-based statistical model for sms text normalization. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 33–40. Association for Computational Linguistics.

Barrett, J. and Grems, M. (1960). Abbreviating words systematically. *Communications of the ACM*, 3(5):323–324.

Bourne, C. and Ford, D. (1961). A study of methods for systematically abbreviating english words and names. *Journal of the ACM (JACM)*, 8(4):538–552.

HaCohen-Kerner, Y., Kass, A., and Peretz, A. (2008). Combined one sense disambiguation of abbreviations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 61–64. Association for Computational Linguistics.

Hassan, H. and Menezes, A. (2013). Social text normalization using contextual graph random walks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1577–1586, Sofia, Bulgaria. Association for Computational Linguistics.

Hearst, M. S. (2003). A simple algorithm for identifying abbreviation definitions in biomedical text.

Jain, A., Cucerzan, S., and Azzam, S. (2007). Acronym-expansion recognition and ranking on the web. In *Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on*, pages 209–214. IEEE.

Li, J., Ott, M., and Cardie, C. (2013). Identifying manipulated offerings on review portals. In *EMNLP*, pages 1933–1942.

Ling, W., Dyer, C., Black, A. W., and Trancoso, I. (2013). Paraphrasing 4 microblog normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 73–84, Seattle, Washington, USA. Association for Computational Linguistics.

Nenadić, G., Spasić, I., and Ananiadou, S. (2002). Automatic acronym acquisition and term variation management within domain-specific texts.

In *Third International Conference on Language Resources and Evaluation (LREC2002)*, pages 2155–2162.

Norris, J. R. (1998). *Markov chains*. Number 2008. Cambridge university press.

Park, Y. and Byrd, R. (2001). Hybrid text mining for finding abbreviations and their definitions. In *Proceedings of the 2001 conference on empirical methods in natural language processing*, pages 126–133.

Sproat, R., Black, A. W., Chen, S., Kumar, S., Ostendorf, M., and Richards, C. (2001). Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

Sproat, R., Tao, T., and Zhai, C. (2006). Named entity transliteration with comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 73–80. Association for Computational Linguistics.

Sun, X., Li, W., Meng, F., and Wang, H. (2013). Generalized abbreviation prediction with negative full forms and its application on improving chinese web search. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 641–647, Nagoya, Japan. Asian Federation of Natural Language Processing.

Sun, X., Okazaki, N., and Tsujii, J. (2009). Robust approach to abbreviating terms: A discriminative latent variable model with global information. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 905–913. Association for Computational Linguistics.

Sun, X., Wang, H., and Wang, B. (2008). Predicting chinese abbreviations from definitions: An empirical learning approach using support vector regression. *Journal of Computer Science and Technology*, 23(4):602–611.

Taghva, K. and Gilbreth, J. (1999). Recognizing acronyms and their definitions. *International Journal on Document Analysis and Recognition*, 1(4):191–198.

Tsuruoka, Y., Ananiadou, S., and Tsujii, J. (2005). A machine learning approach to acronym generation. In *Proceedings of the ACL-ISMB Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*, pages 25–31. Association for Computational Linguistics.

Wren, J., Garner, H., et al. (2002). Heuristics for identification of acronym-definition patterns within text: towards an automated construction of comprehensive acronym-definition dictionaries. *Methods of information in medicine*, 41(5):426–434.

Yang, Y. and Eisenstein, J. (2013). A log-linear model for unsupervised text normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 61–72, Seattle, Washington, USA. Association for Computational Linguistics.

Yu, H., Kim, W., Hatzivassiloglou, V., and Wilbur, J. (2006). A large scale, corpus-based approach for automatically disambiguating biomedical abbreviations. *ACM Transactions on Information Systems (TOIS)*, 24(3):380–404.

Zhang, L., Li, S., Wang, H., Sun, N., and Meng, X. (2012). Constructing Chinese abbreviation dictionary: A stacked approach. In *Proceedings of COLING 2012*, pages 3055–3070, Mumbai, India. The COLING 2012 Organizing Committee.

Zhu, X., Ghahramani, Z., Lafferty, J., et al. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919.

Type-Aware Distantly Supervised Relation Extraction with Linked Arguments

Mitchell Koch **John Gilmer** **Stephen Soderland** **Daniel S. Weld**
Department of Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
{mkoch,jgilme1,soderlan,weld}@cs.washington.edu

Abstract

Distant supervision has become the leading method for training large-scale relation extractors, with nearly universal adoption in recent TAC knowledge-base population competitions. However, there are still many questions about the best way to learn such extractors. In this paper we investigate four orthogonal improvements: integrating named entity linking (NEL) and coreference resolution into argument identification for training and extraction, enforcing type constraints of linked arguments, and partitioning the model by relation type signature.

We evaluate sentential extraction performance on two datasets: the popular set of NY Times articles partially annotated by Hoffmann et al. (2011) and a new dataset, called GoReCo, that is comprehensively annotated for 48 common relations. We find that using NEL for argument identification boosts performance over the traditional approach (named entity recognition with string match), and there is further improvement from using argument types. Our best system boosts precision by 44% and recall by 70%.

1 Introduction

Relation extractors are commonly trained by distant supervision (also known as knowledge-based weak supervision (Hoffmann et al., 2011)), an autonomous technique that creates a labeled training set by heuristically matching the contents of a knowledge base (KB) to *mentions* (substrings) in a textual corpus. For example, if a KB contained the ground tuple BornIn(Albert Einstein, Ulm) then

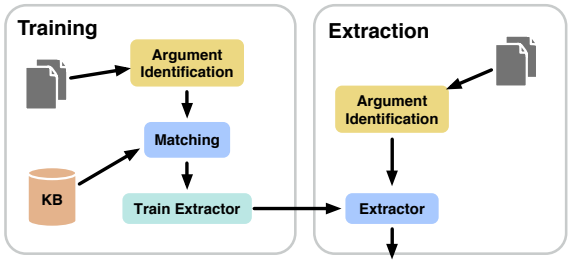


Figure 1: Distantly supervised extraction pipeline.

a distant supervision system might label the sentence "While $[\text{Einstein}]_1$ was born in $[\text{Ulm}]_2$, he moved to Munich at an early age." as a positive training instance of the BornIn relation. Although distant supervision is a simple idea and often creates data with false positives, it has become ubiquitous; for example, all top-performing systems in recent TAC-KBP slot filling competitions used the method.

Surprisingly, however, many aspects of distant supervision are poorly studied. In response we perform an extensive search of ways to improve distant supervision and the extraction process, including using named entity linking (NEL) and coreference to identify arguments for distant supervision and extraction, as well as using type constraints and partitioning the trained model by relation type signatures.

The first step in the distant supervision process is *argument identification* (Figure 1) — finding textual mentions referring to entities that might be in some relation. Next comes *matching*, where KB facts, e.g. tuples such as $R(e_1, e_2)$, are associated with sentences mentioning entities e_1 and e_2 in the assumption that many of these sentences describe the relation R. Most previous systems perform these steps by first using named entity recognition (NER) to identify possible arguments and then using a simple string match, but this crude

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1891–1901,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

approach misses many possible instances. Since the separately-studied task of named entity linking (NEL) is precisely what is needed to perform distant supervision, it is interesting to see if today's optimized linkers lead to improved performance when used to train extractors.

Coreference, the task of clustering mentions that describe the same entity, may also be useful for increasing the number of candidate arguments. Consider the following variant of our previous example: "While $[\text{he}]_1$ was born in $[\text{Ulm}]_2$, $[\text{Einstein}]_3$ moved to Munich at an early age." Since mentions 1 and 3 corefer, one could consider using either the pair $\langle 1, 2\rangle$ or $\langle 3, 2\rangle$ (or both) for training. Intuitively, it seems that $\langle 1, 2\rangle$ is more representative of BornIn and might generalize better, so we consider the use of coreference at both training and extraction time.

Semantic relations often have selectional preferences (also known as type signatures); for example, BornIn holds between people and locations. Therefore, it seems promising to include entity types, whether coarse or fine grained in the distantly supervised relation extraction process. We consider two ways of adding this information. By using NEL to get linked entities, we can impose type constraints on the relation extraction system to only allow relations over appropriately typed mentions. We also investigate using coarse types from NER to learn separate models for different relation type signatures in order to make the models more effective.

In summary, this paper represents the following contributions:

- We explore several dimensions for improving distantly supervised relation extraction, including better argument identification during training and extraction using both NEL and coreference, partitioning the model by relation type signatures, and enforcing type constraints of linked arguments as a post-processing step. While some of these ideas may seem straightforward, to our knowledge they have not been systematically studied. And, as we show, they lead to dramatic improvements.

- Since previous datasets are incapable of measuring an extractor's true recall, we introduce GoReCo, a new exhaustively-labeled dataset with gold annotations for sentential instances of 48 relations across 128 newswire documents from the ACE 2004 corpus (Doddington et al., 2004).

- We demonstrate that NEL argument identification boosts both precision and recall, and using type constraints with linked arguments further boosts precision, yielding a 43% increase in precision and 27% boost to recall. Using coreference during training argument identification gives an additional 7% improvement to precision and further boosts recall by 9%. Partitioning the model by relation type signature offers further benefits, so our best system yields a total boost of 44% to precision and 70% to recall.

2 Distantly Supervised Extraction

At a sentence-level, the goal for relation extraction is to determine for each sentence, what facts are expressed. We describe these as *relation annotations* of the form $s{\rightarrow}R(m_1, m_2)$, where s is a sentence, $R \in \mathcal{R}$ is a relation name, $\mathcal{R}$ is our finite set of target relations, and m_1 and m_2 are *grounded entity mentions* of the form (s, t_1, t_2, e), where t_1 and t_2 delimit a text span in the sentence, and e is a grounded entity.

2.1 Training

During training, the contents of the KB are heuristically matched to the training corpus according to the *distant supervision hypothesis*: if a relation holds between two entities, any sentence containing those two entities is likely to express that relation.

The training KB Δ contains *fact tuples* of form $R(e_1, e_2)$, where $R \in \mathcal{R}$ is a relation name, $\mathcal{R}$ is our finite set of target relations, and e_1 and e_2 are ground entities. The training text corpus Σ contains documents, which contain sentences. *Argument identification* is performed over the text corpus to get grounded mentions m. Then during sentential instance generation, *sentential instances* of the form (s, m_1, m_2) are generated representing a sentence with two grounded mentions. At this point, these sentential instances can be matched to the seed KB, yielding *candidate relation annotations* of the form $s{\rightarrow}R(m_1, m_2)$ by finding all relations that hold over the entities in a sentential instance. These candidate relation annotations are all positive instances to use for training. Negative instance generation is also performed, generating

negative examples of the form $s \rightarrow NA(m_1, m_2)$ indicating that no relation holds between m_1 and m_2. There are several heuristics for generating negative instances, and the number of negative examples and how they are treated can greatly affect performance (Min et al., 2013).

Because the distant supervision hypothesis often does not hold, this training data is noisy. That a fact is in the KB does not imply that the sentence in question is expressing the relation. There has been much work in combating noise in distant supervision training data, but one of the most successful ideas is to train a multi-instance classifier which assumes at-least-one relation holds for positive bags. We use Hoffmann et al. (2011)'s MULTIR system, which uses a probabilistic graphical model to jointly reason at the corpus-level and sentence-level, handles overlapping relations in the KB so that multiple relations can hold over an entity pair, and scales to large datasets.

2.2 Extraction

The trained relation extractor can assign a most likely relation and a confidence score to a sentential instance (s, m_1, m_2). To get these sentential instances, argument identification and sentential instance generation are applied to new documents. Then the relation extractor potentially yields a relation annotation of the form $s \rightarrow R(m_1, m_2)$, or potentially no relation. At extraction time a mention m might have a NIL link if a corresponding ground entity was not found during argument identification (meaning the entity is not in the KB). The relation annotations have associated confidence scores, so a threshold can be chosen to only use high-confidence relation annotations.

3 Argument Identification

An important piece of relation extraction is determining what can be an argument, and how to form a semantic representation of it. We define an argument identification function $\textit{ArgIdent}_{\Delta}(D)$, which takes a document D, finds potential arguments, and links them to entities in Δ if possible, yielding $\mathbf{m}$, a set of grounded mentions in D. Previous relation extraction systems have based this on NER. We evaluate NER-based argument identification against argument identification based on NEL, as well as NEL with coreference.

3.1 Named Entity Recognition

Named entity recognition (NER) tags spans of tokens with basic types such as PERSON, ORGANIZATION, LOCATION, and MISC. This is a high accuracy tagging task often performed using a sequence classifier (Finkel et al., 2005). Relation extraction systems can base their argument identification on NER, by using NER to identify text spans indicating entities and then find corresponding entities in the KB through exact string match (Riedel et al., 2010). Some downsides of using NER with exact string match for relation extraction is that it does not allow for overlapping mentions, it can only capture arguments with full names, and it can only capture arguments with types of the NER system, e.g., "politician" might not be captured.

3.2 Named Entity Linking

Named entity linking (NEL) is the task of grounding textual mentions to entities in a KB, such as Wikipedia. Thus "named entity" here, has a somewhat broader definition than in NER — these are any entities in the KB, not just those expressed with proper names. Hachey et al. (2013) define three stages that NEL systems take to perform this task: extraction (mention detection), search (generating candidate KB entities for a mention), and disambiguation (selecting the best entity for a mention). There has been much work on the task of NEL in recent years (Milne and Witten, 2008; Kulkarni et al., 2009; Ratinov et al., 2011; Cheng and Roth, 2013).

Our definition of a function $\textit{ArgIdent}(D)$ is completely served by an NEL system. It can find any entity in the KB, and those entities are grounded. Additionally, NEL can have overlapping mentions as well as support for abbreviated mentions like "Obama", or acronyms like "US". NEL does not seek to capture anaphoric mentions, however.

3.3 Coreference Resolution

Coreference resolution is the task of clustering mentions of entities together, typically within a single document. Using coreference, we can find even more mentions than NEL, since it can find pronouns and anaphoric mentions. We seek to use coreference to add additional arguments to those found by NEL, and we refer to this combined argument identification method as *NEL+Coref*. Tak-

ing in arguments from NEL argument identification and coreference clusters, we ground the clusters by picking the most common grounded entity from NEL mentions that occur in a coreference cluster. A difficulty is that mentions from NEL and coreference can have small differences in text spans, such as whether determiners are included. We try to assign each NEL argument to a coreference cluster, first looking for an exact span match, then by looking for the shortest coreference mention that contains it. If the coreference cluster already has matched an NEL argument through exact span match that is different from the one found by looking for the shortest containing coreference mention, the new NEL argument is not added. This gives for each coreference cluster a possible grounding to an entity in the KB. What is provided as final arguments for NEL+Coref argument identification are, in order, grounded NEL arguments, grounded coreference arguments that do not overlap with previous arguments, NIL arguments from NEL that do not overlap with previous arguments, and NIL arguments from coreference that do not overlap with previous arguments.

4 Type-Awareness

Relations have expected types for each argument. Entity types, whether coarse-grained, such as from NER, or fine-grained, such as from Freebase entities, are an important source of information that can be useful for making decisions in relation extraction. We bring type-awareness into the system through partitioning the model, as well as by enforcing type constraints on output relation annotations.

Model Partitioning Instead of building a single relation extractor that can generate sentential instances and then relation annotations with arguments of any type, we can instead build separate relation extractors for each possible coarse type signature, e.g., (PERSON, PERSON), (PERSON, LOCATION), etc., and combine the extractions from the extractor for each type signature. This modification allows each trained model to only handle instances of specific types, and thus relations of that type signature, allowing each to do a better job of choosing relations within the type signature.

Type Constraints We can additionally reject relation annotations where the types of the arguments do not agree with the expected types of the relation. That is, we only accept a relation annotation $s \rightarrow R(m_1, m_2)$ when *EntityTypes*$(e_1) \cap \tau_1 \neq \emptyset$ and *EntityTypes*$(e_2) \cap \tau_2 \neq \emptyset$, where m_1 is linked to e_1, m_2 is linked to e_2, *EntityTypes* provides the set of valid types for an entity, τ_1 is the set of allowed types for the first argument of target relation r, and τ_2 for the second argument.

5 Evaluation Setups

Relation extraction is often evaluated from a macro-reading perspective (Mitchell et al., 2009), in which the extracted facts, $R(e_1, e_2)$, are judged true or false independent of any supporting sentence. For these experiments, however, we take a micro-reading approach in order to strictly evaluate whether a relation extractor is able to extract every fact expressed by a sentence $s \rightarrow R(m_1, m_2)$. Micro-reading is more difficult, but it provides fully semantic information at the sentence and document level allowing detailed justifications, and, for our purposes, allows us to better understand the effects of our modifications. In order to fairly evaluate different systems, even those using different methods of argument identification, we want to use gold evaluation data allowing for varying mention types. We additionally use Hoffmann et al. (2011)'s sentential evaluation as-is in order to better compare with prior work. For our training corpus, we use the TAC-KBP 2009 (McNamee and Dang, 2009) English newswire corpus containing one million documents with 27 million sentences.

5.1 Hoffmann et al. Sentential Evaluation

Hoffmann et al. (2011) generated their gold data by taking the union of sentential instances where some system being evaluated extracted a relation as well as the sentential instances matching arguments in the KB. They took a random sample of these sentential instances and manually labeled them with either a single relation or *NA*. Although this process provides good coverage, since is is sampled from extractions over a large corpus, it does not allow one to measure true recall. Indeed, Hoffmann's method *significantly overestimates recall*, since the random sample is only over sentential instances where a program detected an extraction or a KB match was found. Furthermore, this test set only contains sentential instances in which arguments are marked using NER, which makes it impossible to determine if the use of NEL or

coreference confers any benefit.

Finally, it does not allow for the possibility that there may be multiple relations that should be extracted for a pair of arguments. For example, a CeoOf relation, and an EmployedBy relation might both be present for (Larry Page, Google). To address these issues, we manually annotate a full set of documents with relation annotations. Because we are evaluating changing various aspects of the distant supervision process, we cannot use Riedel et al. (2010)'s distant supervision data as-is as others did on the Hoffmann et al. (2011) sentential evaluation. Instead, we use the TAC-KBP data described above.

5.2 GoReCo Evaluation

In order to allow for variations on mentions (NER, NEL, and coreference each has its own definition of what a mention boundary should be), we want gold relation annotations over coreference clusters broadly defined to allow mentions obtained from NER and NEL, as well as gold coreference mentions. So as long as a relation extraction system extracts a relation annotation $s \rightarrow R(m_1, m_2)$ where m_1 and m_2 are allowed options (based on text spans), it will get credit for extracting the relation annotation. We introduce the GoReCo (gold relations and coreference) evaluation to satisfy these constraints.

We start with an existing gold coreference dataset, ACE 2004 (Doddington et al., 2004) newswire, consisting of 128 documents. To get relation annotations over coreference clusters, we define two human annotation tasks and use the BRAT (Stenetorp et al., 2012) tool for visualization and relation and coreference annotations.

Relation Annotation The annotator is presented with a document with gold mentions indicated and asked to determine for each sentence, what facts involving target relations are expressed by the sentence. They draw an arrow for each fact and label it with the relation. They also have the ability to add mentions not present (ReAnn mentions).

Supplemental Coreference Mentions from NER and NEL are displayed along with ACE and ReAnn mentions from the previous task. The annotator draws coreference links from NER or NEL mentions to an ACE or ReAnn mention if they are coreferent.

We randomly shuffle the 128 ACE 2004 newswire documents and use 64 as a development set and 64 as a test set. To complete annotations of these documents, we only used one original human annotator (hired using the oDesk crowdsourcing platform) and found mistakes by having others check the work, as well as checking false positives of relation extractors on the development set to find patterns of annotation mistakes. On average, there are 7 relation annotations per document.

For the GoReCo evaluation, we define our train/test split (with the separate TAC-KBP corpus used for training) such that each has a different set of documents and entities, in order to evaluate how well the system performs on unseen entities. To do this, we remove entities found in the gold evaluation set from the training KB. (We do not remove entities for the Hoffmann et al. (2011) evaluation, since they do not.) We choose the threshold confidence score for each system using the development set to optimize for F1 and report results on the test set.

5.2.1 Target Relations

Since we use a different evaluation, we also seek to choose a more comprehensive and interesting set of relations than prior work. Riedel et al. (2010), whose train and test data is also used by Hoffmann et al. (2011) and Surdeanu et al. (2012), use Freebase properties under domains /people, /business, and /location. Since /location relations such as /location/location/contains dominate the results (and are relatively uninteresting in that they rarely change), we do not use any /location relations, and instead use the domains /people, /business, and /organization (Google, 2012).

Since many Freebase properties are between an entity and a table instead of another entity, we also use joined relations, such as /people/person/employment_history $\bowtie$ /business/employment_tenure/company , in this case representing employment. We bring in an additional 20 relations of this form, also under /person, /business, and /organization. Additionally we use NELL (Carlson et al., 2010a) relations mapped to Freebase by Zhang et al. (2012).

We only include a relation in our set of target relations if both of its entity arguments are contained in the set of entities found via NER with exact string match or NEL over the training corpus. We also remove inverse relations, since they represent needless duplication. This gives us a set

$\mathcal{R}$ of 105 target relations based on joins and unions of Freebase properties. Of the 105 target relations, 48 were used at least once in the GoReCo data.

6 Experiments and Results

We conduct experiments to determine how changing distantly supervised relation extraction along various dimensions affects performance. We examine the choice of argument identification during training and extraction, as well as the effects of model type partitioning, and type constraints. We consider the space of all combinations of these dimensions, but focus on specific combinations where we find improvements.

6.1 Relation Extraction Setup

We use and modify Hoffmann et al. (2011)'s system MultiR to control our experiments and as a baseline. For NER argument identification as well as for the use of NER in the features, we use use Stanford NER (Finkel et al., 2005). For NEL argument identification we use Wikipedia Miner with the default threshold 0.5, and allowing repeated mentions (Milne and Witten, 2008). Since Wikipedia Miner does not support NIL links, we use non-overlapping NER mentions as NIL links. For coreference, we use Stanford's sieve-based deterministic coreference system (Lee et al., 2013). For sentential instance generation, we take all pairs of non-overlapping arguments in a sentence (in either order). If the arguments have KB links, we do not allow sentential instances where both arguments represent the same entity. We use the same lexical and syntactic features as MultiR, based on the features of Mintz et al. (2009). As required for features, we use Stanford CoreNLP's tokenizer, part of speech tagger (Toutanova et al., 2003), and dependency parser (de Marneffe and Manning, 2008), and use the Charniak Johnson constiuent parser (Charniak and Johnson, 2005). For negative training generation, we take a similar approach to Riedel et al. (2010) and define a percentage parameter n of the number of negative instances divided by the number of total instances. Experimenting with $n \in \{0, 20\%, 80\%\}$, we find that $n = 20\%$ works best for our evaluations, optimizing for F1, although using 80% negative training gives high precision at lower recall. We use frequency-based feature selection to eliminate features that appear less than 10 times, which is helpful both for reducing overfitting as well as constraining memory usage. Since the perceptron learning of MultiR is sensitive to instance ordering, we perform 10 random shuffles and average the models.

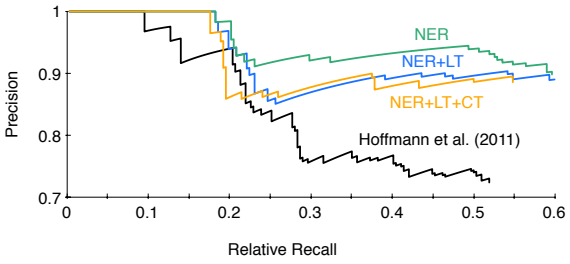


Figure 2: Methods evaluated in the context of Hoffmann et al. (2011)'s sentential extraction evaluation. NER: our NER baseline used for training and extraction, LT: use NEL for training only, CT: use coreference for training only. (NER+LT+CT means we use NER for extraction, and NEL+Coref for training.)

For model type partitioning, when training with NER, we ensure that the NER types match the coarse relation type signatures. For NEL, we attempt to use NER for coarse types of arguments, but if an NER type is not present, we map the Freebase type to its FIGER type (Ling and Weld, 2012) to its coarse type. For type constraints, we use Freebase's expected input and output types for relations. For NIL links, we use the NER type of PERSON, ORGANIZATION, or LOCATION, if available, mapping it to appropriate Freebase types.

6.2 NER Baseline

As a result of a larger training set, as well as model averaging, our baseline, which is otherwise equivalent to the methods of Hoffmann et al. (2011) and uses their MultiR system, has slightly higher precision as shown in Figure 2, curve NER. It is also higher than that of Xu et al. (2013), who achieved higher performance than Hoffmann et al. (2011); our baseline gets 89.9% precision and 59.6% relative recall, while Xu et al. (2013)'s system gets 84.6% precision and 56.1% relative recall. See Figure 3 and Table 1 for results on GoReCo.

6.3 NEL and Type Constraints

On GoReCo, using NEL argument identification increases recall and gives higher precision over the entire curve. We further find that filtering results using type constraints gives a large boost in pre-

cision at a small cost to recall. Note the increase in performance from NER to NEL to NEL+TC in Figure 3a, as well as in Table 1. Using NEL gives more recall, since it is able to capture arguments that NER cannot, such as professions like "paleontologist". The decrease in recall from type constraints comes from false positives in the type constraints process including from non-ideal links, e.g., "paleontologist" might get linked to the entity Paleontology, so will not have the type required for the Profession relation.

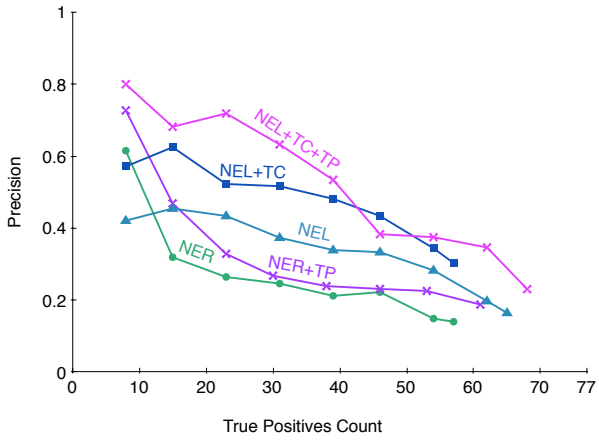
On the Hoffmann et al. (2011) sentential evaluation, we were not able to use NEL argument identification at extraction time, because the instances in the test set are from NER argument identification. We tried using NEL only at training time and found that it got similar performance to using NER (Figure 2, curve NER+LT). Doing the same on GoReCo yielded slightly lower recall, because of the mismatch of features learned from NEL arguments (Figure 3b, curve NER+LT).

6.4 NEL+Coref Argument Identification

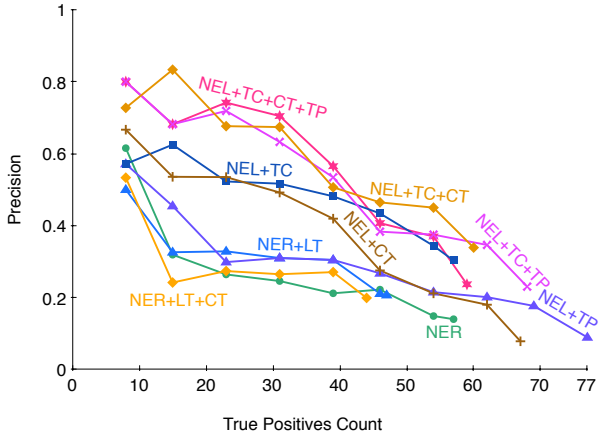
Using NEL+Coref for both training and extraction (see Table 1) introduces noise from arguments not encountered during training time, but using NEL+Coref just for training results in a decrease in recall but similar precision (Figures 2 and 3b).

We found using NEL+Coref at test time unhelpful for this dataset, because there were no examples we could find where coreference could recover arguments that NEL could not. There were three true positives from NEL+Coref involving pronouns in the GoReCo development set, but there were also proper name versions of the arguments nearby in the same sentences, making coreference unnecessary. Additionally, coreference brings in many mentions such as times like "Friday" or "1954" that do not have corresponding KB matches during training time. These sentential instances have similar features to others involving coreference mentions, and there are not negative instances to weigh against these, since these types do not appear in the training data. Better features more suited to coreference mentions could be helpful here.

At both training and extraction time, coreference can cluster together mentions that can be considered to be separate, such as in "Brian Kain, a 33-year-old accountant", "Brian Kain" and "accountant" are coreferent in the gold ACE 2004



(a)



(b)

Figure 3: Precision versus true positives count curves for different versions of the system evaluated on the GoReCo test set, containing 470 gold instances. NER/NEL: argument identification used in training and extraction, LT: use NEL for training only, CT: use coreference for training only, TC: type constraints, TP: model type partitioning.

dataset. This means that type constraints will disregard a Profession annotation between these when it should not, because "Brian Kain" (which would have been a NIL link) gets the link of "accountant". This effect contributes to the decrease in recall.

6.5 Model Type Partitioning

Using type partitioning helps both NER and NEL based models as shown with the +TP curves in Figure 3). Partitioning by type signature results in each model being able to better choose relations for sentential instances of that type signature. In the Partitioned columns of Table 1, removing type partitioning from the best system (NEL training

	Single			Partitioned		
	R	P	F1	R	P	F1
NER training						
NER extraction	7.9	21.8	11.6	11.3	21.0	14.7
NEL extraction	8.5	21.4	12.2	9.8	19.7	13.1
NEL training						
NER extraction	9.6	21.1	13.2	8.9	25.1	13.2
NEL extraction	10.0	30.5	15.1	**15.3**	16.7	16.0
NEL w/TC extraction	11.7	31.1	17.0	13.4	31.3	**18.8**
NEL+Coref training						
NER extraction	9.4	19.2	12.6	6.8	28.3	11.0
NEL extraction	12.1	27.5	16.8	11.1	21.6	14.6
NEL w/TC extraction	**12.8**	**33.3**	**18.5**	12.1	**34.1**	17.9
NEL+Coref extraction	10.6	20.4	14.0	10.0	12.9	11.3
NEL+Coref w/TC extraction	9.4	22.7	13.3	7.9	19.1	11.1

Table 1: Evaluation of different versions of the relation extraction system on the GORECO test set. For nearly all systems, partitioning the model by argument types boosts F1, as does using NEL at either training or extraction time, and using coreference at training time with type constraints (w/TC) raises F1 except with coreference at extraction time and when combined with type partitioning.

and extraction, with type constraints, Partitioned) results in a decrease in F1 from 18.8% to 17.0%. Table 2 shows by-relation performance results for the best system (curve NEL+TC+TP in Figure 3a).

6.6 Other Dimensions Explored

We also experimented with adding generalized features that replaced lexemes with WordNet classes (Fellbaum, 1998), which had uneven results. We observed a small but consistent improvement on the NER baseline (11.6% F1 to 12.7% F1 on GORECO), but after introducing NEL argument identification and partitioning, we no longer observed the improvement. For some relations, there was a small gain in recall that was offset by a loss in precision, but for others, the gain in recall outweighed the loss of precision.

We experimented with a negative instance feedback loop that ran a trained extractor over the training corpus and tested whether each extraction made was in fact a negative example. Even though the training corpus contains one million documents, this method only yielded a few thousand new negative instances due to the difficulty of being certain an extraction should be negative. A naïve approach would simply ensure that both entities participate in a relation in the KB; this is troublesome, because of KB incompleteness and because of type errors. For example Freebase contains BornIn(Barack Obama, Honolulu), but our extractor extracted BornIn(Barack Obama, Hawaii). To avoid labeling this true extraction as a negative instance we have to be robust about location semantics. We selected new negative instances $NA(e_1, e_2)$ from our initial extractor that had e_1 in the knowledge base, with e_1 participating as the first argument in the extracted relation but without e_2 as the second argument. The results were promising for some relations but overall inconclusive as identifying true negatives is quite difficult.

Relation	#Extractions	#TP	#FP
Nationality	**50**	11	38
Profession	43	**23**	20
EmployedBy	27	17	10
Spouse	22	2	20
LivedIn	6	4	2
OrgInCitytown	4	3	1
AthletePlaysForTeam	2	2	0
OrgType	1	1	0

Table 2: By-relation evaluation of the best system (NEL with type constraints and type partitioning) on the GORECO test set. The true positives (TP) are the number of gold relations over coreference clusters that matched, so multiple extractions can match a single true positive.

7 Related Work

There has been much recent work on distantly supervised relation extraction. Mintz et al. (2009) use Freebase to train relation extractors over Wikipedia without labeled data using multi-class logistic regression and lexical and syntactic features. Hoffmann et al. (2011) use a probabilistic graphical model for multi-instance, multi-label

learning and extract over newswire text using Freebase relations. Surdeanu et al. (2012) take a similar approach and use soft constraints and logistic regression. Riedel et al. (2013) integrate open information extraction with schema-based, proposing a universal schema approach, including using features based on latent types. There has also been recent work on reducing noise in distantly supervised relation extraction (Nguyen and Moschitti, 2011; Takamatsu et al., 2012; Roth et al., 2013; Ritter et al., 2013). Xu et al. (2013) and Min et al. (2013) improve the quality of distant supervision training data by reducing false negative examples.

Distant supervision is related to semi-supervised bootstrap learning work such as Carlson et al. (2010b) and many others. Note that distant supervision can be viewed as a subroutine of bootstrap learning; bootstrap learning can continue the process of distant supervision by taking the new tuples found and then training on those again, and repeating the process.

There has also been work on performing NEL and coreference jointly (Cucerzan, 2007; Hajishirzi et al., 2013), however these systems do not perform relation extraction. Singh et al. (2013) performs joint relation extraction, NER, and coreference in a fully-supervised manner. They get slight improvement by adding coreference, but do not use NEL. Ling and Weld (2013) extend MULTIR to find meronym relations in a biology textbook. They get slight improvement over NER by using coreference to pick the best mention of an entity in the sentence for the meronym relation at training and extraction time.

8 Conclusions and Future Work

Given the growing importance of distant supervision, a comprehensive understanding of its variants is crucial. While some of the optimizations we propose may seem intuitive, they have not previously been systematically explored. Our experiments show that NEL, type constraints, and type partitioning are extremely important in order to best take advantage of the seed KB during training as well as known information at extraction time. Our best system results in a 44% increase in precision, and a 70% increase in recall over our NER baseline using GORECO. While we were not able to evaluate all our methods on Hoffmann et al. (2011)'s sentential evaluation, our baseline performs significantly better than previous methods, especially in precision, and training-only modifications perform similarly in both evaluations.

Future work will explore the use of NEL in distantly supervised relation extraction further, tuning a confidence parameter for the NEL system, and determining whether different confidence parameters should be used for training and extraction. Another possible direction is interleaving NEL with relation extraction by using newly extracted facts to try to improve NEL performance.

We freely distribute GORECO a new gold standard evaluation for relation extraction consisting of exhaustive annotations of the 128 documents from ACE 2004 newswire for 48 relations. The source code of our system, its output, as well as our gold data are available at
`http://cs.uw.edu/homes/mkoch/re`.

Acknowledgements

We thank Raphael Hoffmann, Luke Zettlemoyer, Mausam, Xiao Ling, Congle Zhang, Hannaneh Hajishirzi, Leila Zilles, and the anonymous reviewers for helpful feedback. Additionally, we thank Anand Mohan and Graeme Britz for annotations and revisions of the GORECO dataset. This work was supported by Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181, ONR grant N00014-12-1-0211, a gift from Google, a grant from Vulcan, and the WRF / TJ Cable Professorship. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1256082.

References

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010a. Toward an architecture for never-ending language learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-10)*.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka, Jr., and Tom M. Mitchell. 2010b. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 101–110, New York, NY, USA. ACM.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *EMNLP*.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, pages 708–716.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Google. 2012. Freebase data dumps. `https://developers.google.com/freebase/data`.

Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. 2013. Evaluating entity linking with wikipedia. *Artif. Intell.*, 194:130–150, January.

Hannaneh Hajishirzi, Leila Zilles, Daniel S. Weld, and Luke S. Zettlemoyer. 2013. Joint coreference resolution and named-entity linking with multi-pass sieves. In *EMNLP*, pages 289–299. ACL.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL-HLT*, pages 541–550.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 457–466, New York, NY, USA. ACM.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Comput. Linguist.*, 39(4):885–916, December.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI)*.

Xiao Ling and Daniel S. Weld. 2013. Extracting meronyms for a biology knowledge base using distant supervision. In *Automated Knowledge Base Construction (AKBC) 2013: The 3rd Workshop on Knowledge Extraction at CIKM*.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the tac 2009 knowledge base population track. In *Text Analysis Conference (TAC)*, volume 17, pages 111–113.

David Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 509–518, New York, NY, USA. ACM.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of NAACL-HLT*, pages 777–782.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-2009)*, pages 1003–1011.

Tom M. Mitchell, Justin Betteridge, Andrew Carlson, Estevam Hruschka, and Richard Wang. 2009. Populating the semantic web by macro-reading internet text. In *The Semantic Web-ISWC 2009*, pages 998–1002. Springer.

Truc-Vien T. Nguyen and Alessandro Moschitti. 2011. End-to-end relation extraction using distant supervision from external semantic repositories. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 277–282, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1375–1384, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECML/PKDD (3)*, pages 148–163.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. 2013. Relation extraction with matrix factorization and universal schemas. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL '13)*, June.

Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. 2013. Modeling missing data in distant supervision for information extraction. *TACL*, 1:367–378.

Benjamin Roth, Tassilo Barth, Michael Wiegand, and Dietrich Klakow. 2013. A survey of noise reduction methods for distant supervision. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, AKBC '13, pages 73–78, New York, NY, USA. ACM.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *CIKM Workshop on Automated Knowledge Base Construction (AKBC)*.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a Web-based Tool for NLP-Assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, Stroudsburg, PA, USA, April. Association for Computational Linguistics.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 721–729, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wei Xu, Zhao Le, Raphael Hoffmann, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *Proceedings of the 2013 Conference of the Association for Computational Linguistics (ACL 2013)*, Sofia, Bulgaria, July. Association for Computational Linguistics.

Congle Zhang, Raphael Hoffmann, and Daniel S. Weld. 2012. Ontological smoothing for relation extraction with minimal supervision. In *AAAI*.

Automatic Inference of the Tense of Chinese Events Using Implicit Linguistic Information

Yuchen Zhang
Brandeis University
415 South Street
Waltham, MA
yuchenz@brandeis.edu

Nianwen Xue
Brandeis University
415 South Street
Waltham, MA
xuen@brandeis.edu

Abstract

We address the problem of automatically inferring the tense of events in Chinese text. We use a new corpus annotated with Chinese semantic tense information and other implicit Chinese linguistic information using a "distant annotation" method. We propose three improvements over a relatively strong baseline method – a statistical learning method with extensive feature engineering. First, we add two sources of implicit linguistic information as features – eventuality type and modality of an event, which are also inferred automatically. Second, we perform joint learning on semantic tense, eventuality type, and modality of an event. Third, we train artificial neural network models for this problem and compare its performance with feature-based approaches. Experimental results show considerable improvements on Chinese tense inference. Our best performance reaches 68.6% in accuracy, outperforming a strong baseline method.

1 Introduction

As a language with no grammatical tense, Chinese does not encode the temporal location of an event directly in a verb, while in English, the grammatical tense of a verb is a strong indicator of the temporal location of an event. In this paper we address the problem of inferring the semantic tense, or the temporal location of an event (e.g., present, past, future) in Chinese text. The semantic tense is defined relative to the utterance time or document creation time, and it does not always agree with the grammatical tense in languages like English where there is grammatical tense. Inferring semantic tense potentially benefits natural language processing tasks such as Machine Translation and Information Extraction (Xue, 2008; Reichart and Rappoport, 2010; Ye et al., 2006; Ye, 2007; Liu et al., 2011), but previous work has shown that automatic inference of the semantic tense of events in Chinese is a very challenging task (Xue, 2008; Ye et al., 2006; Liu et al., 2011).

There are at least two reasons why this is a difficult problem. First, since Chinese does not have grammatical tense which could serve as an important clue when annotating the semantic tense of an event, generating consistent annotation for Chinese semantic tense has proved to be a challenge. Xue and Zhang (2014) use a "distant annotation" method to address this problem. They take advantage of an English-Chinese parallel corpus with manual word alignments (Li et al., 2012) , and perform annotation on the English side, which provides more explicit information such as grammatical tense that helps annotators decide the appropriate semantic tense. The annotations are then projected to the Chinese side via the word alignments. They show consistent annotation agreements on semantic tense. Second, the lack of grammatical tense also makes automatic inference of Chinese semantic tense challenging since the grammatical tense would be an important source of information for predicting the semantic tense. Previous work has shown that it is very difficult to achieve high accuracy using standard machine learning techniques such as Maximum Entropy and Conditional Random Field classifiers combined with extensive feature engineering.

We address these challenges in two ways. First of all, we take advantage of the newly annotated corpus described in (Xue and Zhang, 2014) in which semantic tense is annotated together with eventuality type and modality using the distant annotation method. This makes it possible to use these two additional sources of information to help predict tense. Eventuality type and modality are intricately tied to tense. For example, Smith and

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1902–1911,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Erbaugh (2005) show that states by default hold in the present but (episodic) events occur by default in the past. This means knowing the eventuality type of an event would help determine the tense. Eventuality type and modality are also annotated on the English side and then projected onto the Chinese side via manual word alignments, taking advantage of the rich morphosyntactic clues in English. High inter-annotator agreement scores are also reported on eventuality type and modality.

We experimented with two ways of using eventuality type and modality information. In the first approach, we first train statistical machine learning models to predict eventuality type and modality and then use these two sources of information as features to predict semantic tense. In the second approach we trained joint learning models between semantic tense and eventuality type, and between semantic tense and modality. We show both approaches improve the tense inference accuracy over a baseline where these two sources of information are not used. Second, in our statistical machine learning experiments on tense inference using feature engineering, we find that the design of feature templates has great influence on the results. So in order to explore more possible feature combinations and mitigate the feature engineering work, we apply artificial neural network models to this problem. This shows improvements on tense inference accuracy as well in some of the experiment settings.

The rest of the paper is organized as follows. Section 2 discusses related work in automatic tense inference. Section 3 briefly introduces the distant annotation method. In section 4, we describe our experiments and analyze the experimental results. We conclude this paper in section 7.

2 Related Work

Inferring the semantic tense of events in Chinese text is not a new topic. There have been several attempts at it, yet high accuracy in this task has proved to be elusive. Using a corpus with tense annotated directly in Chinese text, Xue (2008) performed extensive feature engineering in a machine learning framework to address this problem. They used both local lexical features and structured features extracted from manually annotated syntactic parsing trees. In our baseline method, we adopt most of their features as the baseline, only on a new corpus in which semantic tenses are not annotated directly on Chinese events but projected from annotations from the English side of a parallel Chinese-English corpus. In our experiments, we also use structural features extracted from automatic parse trees, so our experimental settings are more realistic.

Ye et al. (2006) took a similar approach in which they predict tense with feature engineering in a statistical learning framework. They also used a Chinese-English parallel corpus and projected tense for English events onto Chinese events via human alignments. The main difference between their data and ours is that they used the grammatical tense of the English events, while we use human-annotated semantic tense which we believe are more "transferrable" across languages as it is free of the language-specific idiosyncrasies of grammatical tense. In addition, they also used human annotated linguistic information as "latent" features in their work, which are similar to our implicit linguistic features. However, the "latent" features that they used in their system are human-annotated, while the eventuality type and modality features in our system are predicted automatically. Another difference is that they ignored events that are not verbs. For example, they excluded verbal expressions in Chinese that are translated into nominal phrases in English. In contrast, we kept all events in our data, and they can be realized as verbs, nouns, as well as words in other parts of speech. We performed separate experiments on events realized as verbal expressions and events not in verbal expressions to investigate their impact on semantic tense inference.

Liu et al. (2011) introduced more global features in a machine learning framework, and on top of that proposed an iterative learning algorithm which better handles noisy data, but they also ignored events that are not realized as verbal expressions, or events that are verbal expressions but have more than one verb in them. They mainly focused on events that are one-verb expressions.

In a similar work on inferring tense in English text, Reichart and Rappoport (2010) aimed at inferring fine-grained semantic tenses for events in English. They introduced a fine-grained sense taxonomy for tense in a more general Tense Sense Disambiguation (TSD) task to annotate and disambiguate semantic tenses. The underlying senses include "things that are always true", "general and repeated actions and habits", "plans, expectations

and hopes", etc., which encode a combination of tense, eventuality type and modality. In the corpus that we use, the same information is organized in a more structured manner along three dimensions – semantic tense, eventuality type, and modality.

3 Distant Annotation

Figure 1 shows the distant annotation procedure from (Xue and Zhang, 2014). Starting with a word-aligned parallel English-Chinese corpus, all sentences are part-of-speech (POS) tagged first and then all verb instances in the English text as well as expressions aligned with verb instances on the Chinese side are targeted for annotation. As we will show in Section 4, these expressions include verbs as well as nouns, prepositions and word sequences "headed" by a verb. We consider those expressions as events. Annotators work only on the English side and tag every event with a pre-defined semantic tense label. These labels are then projected from the English side to the Chinese side via word alignments. The resulting corpus contains events annotated with semantic tense labels in both languages. Categories for semantic tense are "Past", "Present", "Future", "Relative Past", "Relative Present", "Relative Future", and "None".

Events annotated with relative tenses are also linked to another event that serves as the temporal reference for the event in question. In some cases the relative tense can be resolved to an absolute tense. For example, if an event is annotated with a "relative past" tense to a reference event that is annotated with a present tense, then the semantic tense of that event can resolve to an absolute "past" tense. In other cases, they can not be resolved. For example, if an event is labeled with a "relative future" tense and the reference event has a past tense, then its tense cannot be resolved to an absolute tense, which is defined with regard to the utterance time or document creation time. In our work, where possible, we resolve these links and keep only absolute tense labels. For events with relative tenses that can not be resolved (i.e. events which are "Relative Future" to "Past" events, or events which are "Relative Past" to "Future" events), we use "None" as the default label.

Eventuality type and modality are labeled in the same way as auxiliary annotation that can help with the inference of tense. Labels for eventuality type include "Episodic", "Habitual", "State", "Progressive", "Completed", and "None". Labels for modality are "Actual", "Intended", "Hypothetical", "Modalized", and "None". Readers are refered to (Xue and Zhang, 2014) for detailed explanations of each label.

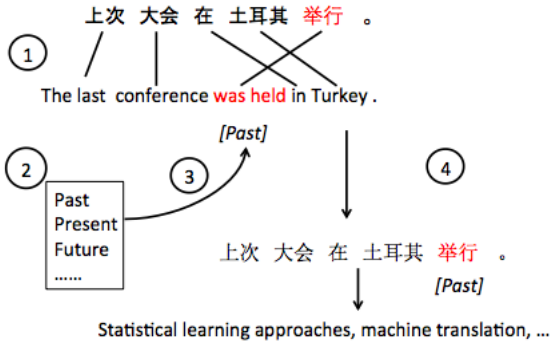


Figure 1: Distant annotation procedure.

As we mentioned in Section 2, in this corpus not only verbs but also their counterparts on the opposite language are considered as events, yielding events that may not be verbs. For example, in the following sentence pair (1), the Chinese verb (VV) "利用" is aligned with an English noun (NN) "use". In the sentence pair (2), the English verb (VBG) "opening" is aligned with an Chinese noun (NN) "开放".

(1) Statistics show that , in the past five years , Guangxi's foreign trade and its **use** of foreign investments has expanded rapidly.

统计资料显示，过去五年广西对外贸易和**利用(li4yong4)**外资规模迅速扩大。

(2) Beihai has already become a bright star arising from China's policy of **opening** up to the outside world.

北海已成为中国对外**开放(kai1fang4)**中升起的一颗明星。

In this corpus, events could be either one verb, or a verb compound, or a verb sequence "headed" by a verb, or even nouns and words of other parts of speech.

4 Experiments

4.1 Experimental Setting

Xue and Zhang (2014) annotated semantic tense, eventuality type and modality on top of the Parallel Aligned Treebank (Li et al., 2012), a corpus

of word-aligned Chinese-English sentences treebanked based on the Penn TreeBank (Marcus et al., 1993) and the Chinese TreeBank (Xue et al., 2005) standards. Human annotation of tense is performed on the newswire and webblog sections of this corpus. They report that the average pairwise agreement among three annotators consistently stays above 80% and the average Kappa score consistently exceeds 70%, indicating reliable annotation.

Apart from using the entire corpus, we also conducted experiments on three different subsets of the corpus. An examination of the data indicates that newswire data is grammatically more formal and complete than webblog data, so we also conducted separate experiments on newswire data only. Considering that the diversity of the parts of speech of the events may affect the inference accuracy and that most of our features extracted from the parse trees assume that our events being verbs, we also conducted experiments exclusively on "v_events". "v_events" consist of two parts. One part is events that are realized as a single word and the word is a verb; the other one is events which have multiple words but there is only one verb among them. In the latter case, we stripped off words tagged with other parts of speech and only keep the verbs as events. This makes it more effective to use features from previous work that are designed for single verbs. One such feature is the aspect marker. Distinctions between newswire and webblog data and between v_events and other events are further explored in Section 5.1 and Section 5.2. Table 1 presents the statistics for each subset of the experimental data.

dataset	# of v_events	# of all events
nw	6,686	8,268
all	17,153	20,885

Table 1: Statistics of four subsets of the annotated corpus (Chinese side). "nw" denotes the newswire data. "v_events" denotes events that consist of or can be reduced to only a single verb.

For each subset, randomly selected 80% were used as the training set, while 10% were used as the development set and 10% were used as the test set.

4.2 Baseline

Based on previous approaches on Chinese tense inference, we used a Maximum Entropy model with extensive feature engineering as our baseline method. We use the implementation of the Maximum Entropy algorithm in Mallet [1] for our experiments. The corpus is parsed using the Berkeley Parser for the purpose of extracting structure features. Since the Parallel Alignment TreeBank is a subset of the Chinese TreeBank (CTB) 8.0, we automatically parsed the CTB 8.0 by doing a 10-fold cross validation. The bracketing F-score is 80.5%. Feature extractions are performed on the automatic parse trees. Adopted features include previous word and its POS tag, next word and its POS tag, aspect marker following the event, 得following the event, the governing verb of the event, the character string of the main verb in the previous clause that is coordinated with the clause the event is in, whether the event is in quote, and left modifiers of the event including head of adverbial phrases, temporal noun phrases, prepositional phrases, localizer phrases, as well as subordinating clauses. Readers are referred to (Xue, 2008) for details of these features. Since in this corpus an event can span over more than one verb, we also use the character string and the POS string of the entire event instead of one word and one POS tag as features.

- The character string of an event – it could be one or more words. In our corpus, only 69.7% events consist of single word (e.g. "居住", "live"), the other 30.3% of the events are expressed with two or more words (e.g. "引发+了", "have caused").
- The POS string of an event – it could be verbs, nouns, or POS sequences of other word sequences. Table 2 shows the top ten POS tag or POS tag sequences with example word or word sequences.

Other features that we used in the baseline system are as follows.

- DEC – if the word immediately following an event has the POS tag "DEC", use its character string as a feature. In most cases, "DEC" is the POS tag for "的" when it used as a complementizer marking the boundary in a relative clause. This feature implies that an event

[1] http://mallet.cs.umass.edu/

POS	freq	examples
VV	48.2%	居住(live)
NN	5.8%	开放(opening)
VC	5.2%	是(is)
VV+AS	5.2%	引发+了(have caused)
VV+DEC	3.2%	隔绝+的(isolated)
AD+VV	3.0%	正在+建议(is suggesting)
VA	3.0%	大(is big)
AD	2.0%	似乎(seemed)
VE	1.9%	有(there is)
P	1.8%	根据(according to)

Table 2: Frequencies and examples of the ten most frequent POS tag or POS tag sequences for events in our corpus.

is inside a relative clause modifying a noun phrase and it is more often stative than eventive.

- Determiners – we find the subject of an event from its parse tree and extract the determiner of the subject, if there is one, as a feature. This feature indicates different types of agents, and different types of agents often signal different types of events. For example, individual agents tend to perform one-time episodic actions which are by default located in the past or described by a state in the present, while multiple agents tend to involved in habitual actions that spans over a long period of time.

Baseline results are reported in Table 5 and Table 6, in MaxEnt_b rows.

4.3 Eventuality Type and Modality as Features

Xue and Zhang (2014) reports that gold eventuality type and modality labels significantly help the inference of tense in Chinese, improving the accuracy by more than 20%. However, it is unrealistic to expect to have human annotated eventuality type and modality labels in a random new data set if we want to use these two sources of implicit linguistic information in any Chinese text. So we trained statistical learning models to automatically extract these two labels. We trained Maximum Entropy models and ran a 10-fold cross validation on the entire corpus in order to get automatic labels for every event. Feature used for labeling modality are as follows. Table 3 shows the average accuracies for automatic modality labeling.

- The character string of an event.
- The POS string of an event.
- The character string of an event's governing verb and its POS tag.
- Whether the event is in a conditional clause. If an event is in a subtree with the functional tag "CND", return "True"; otherwise, return "False". This feature indicates that the event's modality label is "Hypothetical".
- Whether the event is in a purpose or reason clause. If an event is in a subtree with the functional tag "PRP", return "True"; otherwise, return "False" as a feature. This feature indicates the event's modality label is "Intended".
- Whether the event string is the start of a sentence. If an event is the start of a sentence, return "True"; otherwise, return "False". Sentences that start with an event is often imperative, and the event generally has "modalized" modality label.

dataset	v_events	all events
nw	81.1%	81.2%
all	75.4%	76.4%

Table 3: Average modality labeling accuracy, using a 10-fold cross validation.

Statistics show that the five labels for modality have a skewed distribution in this corpus. Among all events, 67.3% of them fall in the "Actual" category, while the events of all the other categories are around or less than 10%. Similar distributions are found in all four subsets of the data. Still, compared with always choosing the most frequent label (around 67% accuracy), we still get a big improvement from our statistical model, even though only a very simple set of features are used.

Features used for labeling eventuality type are as follows. Table 4 shows the average accuracies for automatic eventuality type labeling.

- The character string of an event.
- The POS string of an event.

- Adverbs on the left that modifies the event.
- Aspect marker following the event
- Whether the event is Inside a relative clause. If an event is in a CP subtree with the word "的" and POS tag "DEC" as its last node, return "True"; otherwise, return "False". Events in relative clauses modifying a noun phrase and tend to be more often stative than eventive.

dataset	v_events	all events
nw	68.7%	67.7%
all	65.3%	65.1%

Table 4: Average automatic eventuality type labeling accuracy using a 10-fold cross validation.

The six labels of eventuality type are also distributed unevenly. The first group of columns in Figure 3 shows the distribution of all events. Over 65% of events are either "Episodic" or "State", while the other types of events are less than 15%. There are two categories that are even less than 5%. However, even though we only use some simple features, our model still beats the most frequent label baseline (around 35% accuracy) by a big margin, as shown in Table 4.

Tense inference accuracies using automatic eventuality type and/or modality features are reported in Table 5 and Table 6, in MaxEnt_e, MaxEnt_m, and MaxEnt_em rows.

4.4 Joint Learning

Apart from using eventuality type and modality labels as features, we also conducted joint learning experiments on them. Joint learning are applied on 1) tense and eventuality type, and 2) tense and modality. Features used are the union of the two sets of features in inferring each single label. MaxEnt_jle and MaxEnt_jlm rows in Table 5 and Table 6 present the experimental results on joint learning.

4.5 Artificial Neural Network

For each of the experiments using the maximum entropy algorithm, we conducted a neural network experiment using the same setting in order to explore more possible feature combinations and mitigate the feature engineering work. We convert the features in each of our tense inference methods into feature vectors. If a feature is not a word, we use a one-hot representation for that feature (a vector with all 0s except for a 1 at the place of the feature's index in our feature lexicon). If a feature is a word, we convert it into a word embedding. To get a dictionary of word embeddings, we use the word2vec tool [2] (Mikolov et al., 2013) and train it on the Chinese Gigaword corpus (LDC2003T09). For each word embedding, a 300-dimensional vector is used. Artificial neural networks are built using the theano package [3] (Bergstra et al., 2010). We use 5000 hidden units for all networks and set the learning rate α = 0.01. Experimental results are presented in the ANN rows of Tables 5 and 6.

5 Results Analysis

A comparison of the baseline accuracy for the four different subsets of the data shows that (1) tense inference is slightly better on v_events than on all events, but the difference is not substantial; and (2) tense inference on newswire data performs better than on all data by around 8% on v_events and around 5% on all events, verifying our assumption that automatic tense inference is easier on newswire data than webblog data. Although our experiments are performed on different data sets from that of previous work, our baseline method still shows strong results compared with previous work (Xue, 2008; Ye et al., 2006; Liu et al., 2011).

Adding automatic eventuality type and modality labels as features for semantic tense inference leads to improvements over the baseline on all four data subsets. In fact they provide considerable improvements (around 2% increase) on newswire v_events dataset. MaxEnt_e rows report results when only automatic eventuality type is added as a feature, and MaxEnt_m rows report results when only automatic modality is added as a feature. They both outperform (or, in several datasets, match) the baseline results on all datasets. MaxEnt_em rows report results when both automatic linguistic labels are added as features, and they show further improvements over when only one source of information is used. Analysis of the results shows again that tense inference accuracy is higher than webblog data under this experiment condition. The results also show that after adding eventuality type and modality as features,

[2] http://code.google.com/p/word2vec/
[3] http://deeplearning.net/software/theano/

method	all data	nw data
MaxEnt_b	58.9%	66.8%
MaxEnt_e	59.5%	67.9%
MaxEnt_m	59.5%	67.1%
MaxEnt_em	59.6%	**68.6%**
MaxEnt_jle	59.6%	63.5%
MaxEnt_jlm	**60.5%**	66.9%
MaxEnt_ge	74.6%	**77.4%**
MaxEnt_gm	66.6%	70.0%
MaxEnt_gem	**76.2%**	76.9%
ANN_b	63.4%	67.2%
ANN_e	62.6%	66.1%
ANN_m	**63.4%**	59.8%
ANN_em	59.7%	**68.3%**
ANN_jle	62.7%	64.5%
ANN_jlm	62.0%	65.6%

Table 5: Accuracy of tense inference on v_events. Best performances for each group of methods are in bold.

method	all data	nw data
MaxEnt_b	59.7%	65.1%
MaxEnt_e	59.9%	65.1%
MaxEnt_m	59.9%	65.4%
MaxEnt_em	59.9%	65.5%
MaxEnt_jle	59.7%	62.7%
MaxEnt_jlm	**60.4%**	**65.6%**
MaxEnt_ge	75.3%	**76.1%**
MaxEnt_gm	67.1%	69.0%
MaxEnt_gem	**76.2%**	75.9%
ANN_b	63.0%	64.0%
ANN_e	**63.2%**	**66.9%**
ANN_m	60.1%	64.7%
ANN_em	57.8%	66.1%
ANN_jle	61.4%	63.0%
ANN_jlm	62.9%	63.5%

Table 6: Accuracy of tense inference on all events. Best performances for each group of methods are in bold.

the improvements on v_events (0.7% and 1.8%) are much bigger than that on all events (0.2% and 0.4%), regardless of the data genre (newswire or weblog).

In order to test the potential for these two new features, we also conducted experiments using gold eventuality type and/or modality labels as features for the Maximum Entropy models (Table 5 and Table 6, MaxEnt_ge, MaxEnt_gm, and MaxEnt_gem rows.). They outperform our best MaxEnt results by around 10% on newswire data and around 15% on all data, indicating strong potentials for more accurately classified automatic eventuality type and modality labels.

Results also show that joint learning with modality proves to be working better than the baseline (Table 5 and Table 6, MaxEnt_jle, MaxEnt_jlm). In fact, on the datasets with all events, joint learning with modality produces the highest accuracy among all approaches. However, joint learning with eventuality is even worse than the baseline. One possible explanation is that the lower eventuality type classification accuracy affects the tense inference accuracy. We also believe there is still room for improvement with features tuned for the joint learning model. Simply adding the features may not be the best strategy.

On the entire dataset, regardless of v_events or other events, results of the neural network models show improvements over the maximum entropy models under most experimental conditions. A clear trend is that artificial neural networks help more on all data than on newswire data only, indicating greater potentials of the neural network models to select and combine features with carefully trained parameters, given noisier but larger training sets.

Experimental results also show significant differences in accuracy between newswire data and webblog data, and smaller but still recognizable difference between v_events and all events. Therefore, we specifically look into distinctions between these data sets.

5.1 Newswire Data vs. Webblog Data

Considering the big gap in accuracy between newswire and webblog data in our baseline results, we delve deeper into the data and found several major distinctions between these two domains that might have contributed to the rather significant difference in performance on tense inference. First, we look into the word frequency distribution of the two datasets. Here by "word" we mean the character string of an event. We find that both datasets have a small portion of words with high frequencies, but the webblog dataset contains much more low-frequent words than the newswire dataset. In Figure 2, the x-axis shows possible frequencies of words and the y-axis shows the number of words at a particular frequency. It can be seen that the num-

ber of words that appear only once in the webblog dataset is about three times as large as that in the newswire dataset. The entire newswire dataset has a vocabulary of only 2671 entries, while the webblog dataset has a vocabulary size of 6117. This greatly reduces the coverage of features extracted from the training dataset on the events in the test dataset.

Second, webblog data contains more events that are "inherently" ambiguous on temporal location. Among four possible labels for tense in this corpus, "None" is for events whose temporal locations are not clear even to human annotators. Statistics show that in webblog data about 13.4% of the events are tagged as "None", while in newswire data only around 6.7% are "None". Another piece of evidence showing webblog data is harder to process is the different inter-annotator agreement scores for tense annotation on newswire and webblog data reported by (Xue and Zhang, 2014). Newswire data has a 89.0% agreement score and a 84.9% kappa score, while webblog data only has a 81.0% agreement score and a 72.7% kappa. Third, automatic parse trees for newswire data is also more accurate than that for webblog data. The bracketing F-score of automatically parsed newswire data is 83.0% while it is only 80.4% for weblog data. Moreover, sentences in newswire data are more grammatically complete. Analysis shows that webblog data has more dropped constituents in sentences. There are around 40.5% sentences in newswire data that have nominal empty categories, while in webblog data the number is 48.1%. Dropped constituents affect the structures of parse trees and some of the features, which can affect tense inference accuracy.

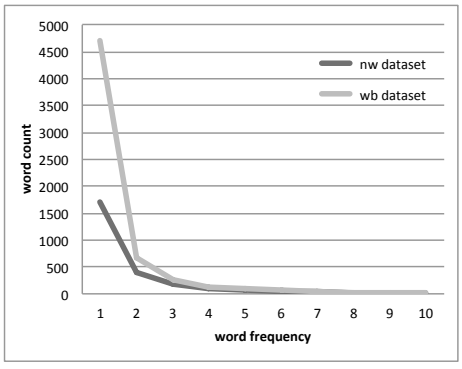


Figure 2: Word frequency distribution in newswire and webblog datasets.

5.2 V_events vs. All Events

In our definition, v_events are (1) events that are single verbs (example 1, 3, 7, 9 in Table 2), and (2) events that are multi-word sequences but only one word among them is a verb and any non-verb words are stripped off (verbs in example 4, 5, 6 in Table 2). Conversely, events that do not fall into this definition include (1) events that have no verbs in their surface form (example 2, 8, 10 in Table 2), and (2) events that have more than one verb in their surface form (e.g. "使+成为(shi3+cheng2wei2)", VV+VV, "make it become"). So from the point of view of a statistical learning algorithm, every v_event has one and only one verb. This makes sure that all features that we used are applicable to v_events. For other events, however, some features may be not applicable. For example, for an event which has a nominal expression, aspect marker, DER, and DEC features are all "None" because these features are only applicable to verbs. Another major distinction between v_events and "other events" is that the distributions of eventuality type labels on them are very different, presented in the second and third groups of columns in Figure 3. There is a rather high percentage of "State" among "other events" and very low percentage of "Completed" and "None". The highly uneven distribution of eventuality type labels make it less effective as a feature for tense inference.

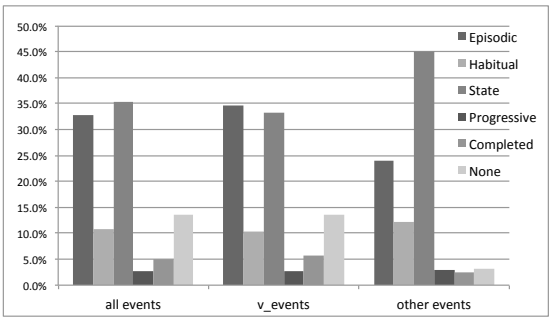


Figure 3: Statistics of eventuality types on different events.

We also find that, on newswire datasets, maximum entropy models and neural network models do not show much difference in performance. To understand this result better, we plot learning curves of the artificial neural network model, trained and tested on newswire v_events dataset. In Figure 4, the black line represents the error rate on training set, and the grey line represents the er-

ror rate on test set. As the size of training data grows, the error rate on the training set gets larger because with more training examples the training set becomes noisier and it gets harder to model all samples with the same number of features; and the error rate on the test set gets smaller because a bigger training set reduces the data sparsity and trains the parameters better. Both lines end at a rather high error rate (around 30%, i.e. only around 70% in accuracy) which means the current network is general enough to cover most cases in the test set, but it is under-fitting the training data. The current model is not specific enough to better capture the fine distinctions between the tense categories. The black line being not very smooth is also understandable, given that there are only around 6000 training examples in the newswire v_events dataset.

Figure 4: Learning curves of the artificial neural network model, trained and tested on newswire v_events dataset.

6 Error Analysis

In order to get a better understanding of the use of eventuality type and modality, we look into the error rates for each error type in greater detail. In Table 7, "Pa" stands for "Past", "Pre" is short for "Present", "Fu" is for "Future", and "No" is "None". For each error type, the left-hand side is the gold-standard tense, and the right-hand side is the wrongly assigned label. Statistics are collected on the newswire v_events data test set. Table 7 compares the different error types between the baseline method and the MaxEnt_em method, the best approach for this dataset. We can see that (1) "Present" and "Past" is the most frequently confused tense pair, and (2) eventuality type and modality information help disambiguate "Present" and "Past" events greatly, and reduce the errors due to mis-classifying "Past" as "Future", or "Future" as "Present", or "None" as "Present".

error type	MaxEnt_b	MaxEnt_em
Pre → Pa	11.7%	11.2%
Pa → Pre	9.8%	9.2%
Pa → Fu	2.5%	2.3%
No → Pa	2.0%	2.0%
Fu → Pre	1.9%	1.6%
Pre → Fu	1.4%	1.4%
Fu → Pa	1.4%	1.4%
No → Pre	1.6%	1.2%
No → Fu	0.5%	0.5%
Pa → No	0.3%	0.3%
Pre → No	0.2%	0.2%
Fu → No	0.0%	0.0%

Table 7: Tense inference error rates for different error types on newswire v_events test set.

A closer examination of the sentences in which events are assigned the wrong tense reveals that "Pre → Pa" error is prone to occur on events in relative clauses. The Chinese verb implies a past episodic event, while the event is actually a present state or habitual event. As a good example, the "生产(sheng1chan3)" event in Sentence (3) is wrongly labeled as "Past" by MaxEnt_b but correctly classified as "Present" by MaxEnt_em with eventuality type "Habitual" and modality tag "Actual" (the underlined part in the Chinese sentence is the relative clause). It is also found that most "Pa → Pre" errors occur on events that are more stative. It is reasonable since classifiers tend to assign "Present" to states and "Past" to episodic events. MaxEnt_em managed to correct some with "episodic" as their correct eventuality type.

(3) 目前该区**生产(sheng1chan3)**此疫苗的 普康公司已形成年产五百万人份的生产规模，这对有效地控制甲肝流行具有重大意义。

At present , the Pu Kang Company , which **produces** the vaccine in this zone , has already formed a production scale of 5 million doses per year , which has great significance in effectively controlling the hepatitis A epidemic .

We are also surprised to see that over 2% "Past" events are classified as "Future" events, ranking

third among all error types. This mistake seems very unlikely, but it is still possible when performing tense inference on a language with no grammatical tense at all. Take the following sentence pair (4) as an example. In the Chinese sentence, MaxEnt_b classifies "讨论(tao3lun4)" as "Future" because there is no grammatical indicator in the Chinese sentence implying that the "discussion" has already happened and it is reasonable to assume the "discussion" is in the near future. However, with eventuality type "Episodic" and modality label "Actual", MaxEnt_em classifies it as "Past" correctly, because episodic events tend to occur in the past and future events tend to get "Intended" or "Hypothetical" modality labels.

(4) 他还说，法国政府"甚至指示它的代表，在联合国安理会**讨论(tao3lun4)** 制裁古巴的议案时不要投赞成票"。

He also said, the French government "even directed its representative not to vote Yes when the Security Council **discussed** the resolution on sanctions on Cuba".

7 Conclusion and Future Work

In this paper, we address the problem of automatic inference of Chinese semantic tense. We took advantage of a new corpus annotated with rich linguistic information, and experimented with three approaches. In the first approach, we use two sources of implicit linguistic information, eventuality type and modality, automatically derived, as features in tense inference. We then conducted joint learning on tense and each of these two information types. Finally, we experimented with using artificial neural networks to train models for tense prediction. All three approaches outperformed a strong baseline, a maximum entropy model with extensive engineering. Our future work will include exploring ways to improve automatic eventuality type and modality labeling accuracy to further improve tense inference accuracy.

Acknowledgments

We would like to thank the three anonymous reviewers for their suggestions and comments. This work is supported by the National Science Foundation via Grant No. 0910532 entitled "Richer Representations for Machine Translation". All views expressed in this paper are those of the authors and do not necessarily represent the view of the National Science Foundation.

References

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June. Oral Presentation.

Xuansong Li, Stephanie Strassel, Stephen Grimes, Safa Ismael, Mohamed Maamouri, Ann Bies, and Nianwen Xue. 2012. Parallel Aligned Treebanks at LDC: New Challenges Interfacing Existing Infrastructures. In *Proceedings of LREC-2012*, Istanbul, Turkey.

Feifan Liu, Fei Liu, and Yang Liu. 2011. Learning from chinese-english parallel data for chinese tense prediction. In *Proceedings of the 5th International Conference on Natural Language Processing*, pages 1116–1124, November.

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

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

Roi Reichart and Ari Rappoport. 2010. Tense sense disambiguation: A new syntactic polysemy task. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 325–334, Cambridge, MA, October. Association for Computational Linguistics.

Carlota S. Smith and Mary Erbaugh. 2005. Temporal interpretation in Mandarin Chinese. *Linguistics*, 43(4):713–756.

Nianwen Xue and Yuchen Zhang. 2014. Buy one get one free: Distant annotation of chinese tense, event type, and modality. In *Proceedings of LREC-2014*, Reykjavik, Iceland.

Nianwen Xue, Fei Xia, Fu dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering*, 11(2):207–238.

Nianwen Xue. 2008. Automatic Inference of the Temporal Location of Situations in Chinese Text. In *EMNLP-2008*, Honolulu, Hawaii.

Yang Ye, Victoria Li Fossum, and Steven Abney. 2006. Latent features in automatic tense translation between Chinese and English. In *The Proceedings of the 5th SIGHAN Workshop on Chinese Language Processing*, Sydney, Australia.

Yang Ye. 2007. *Automatic Tense and Aspect Translation between Chinese and English*. Ph.D. thesis, University of Michigan.

Joint Inference for Knowledge Base Population

Liwei Chen[1], **Yansong Feng** [1*], **Jinghui Mo**[1], **Songfang Huang**[2], and **Dongyan Zhao**[1]
[1]ICST, Peking University, Beijing, China
[2]IBM China Research Lab, Beijing, China
{chenliwei,fengyansong,mojinghui,zhaodongyan}@pku.edu.cn
huangsf@cn.ibm.com

Abstract

Populating Knowledge Base (KB) with new knowledge facts from reliable text resources usually consists of linking name mentions to KB entities and identifying relationship between entity pairs. However, the task often suffers from errors propagating from upstream entity linkers to downstream relation extractors. In this paper, we propose a novel joint inference framework to allow interactions between the two subtasks and find an optimal assignment by addressing the coherence among preliminary local predictions: whether the types of entities meet the expectations of relations explicitly or implicitly, and whether the local predictions are globally compatible. We further measure the confidence of the extracted triples by looking at the details of the complete extraction process. Experiments show that the proposed framework can significantly reduce the error propagations thus obtain more reliable facts, and outperforms competitive baselines with state-of-the-art relation extraction models.

1 Introduction

Recent advances in natural language processing have made it possible to construct structured KBs from online encyclopedia resources, at an unprecedented scale and much more efficiently than traditional manual edit. However, in those KBs, entities which are popular to the community usually contain more knowledge facts, e.g., the basketball player *LeBron James*, the actor Nicholas Cage, etc., while most other entities often have fewer facts. On the other hand, knowledge facts should be updated as the development of entities, such as changes in the cabinet, a marriage event, or an acquisition between two companies, etc.

In order to address the above issues, we could consult populating existing KBs from reliable text resources, e.g., newswire, which usually involves enriching KBs with new entities and populating KBs with new knowledge facts, in the form of *<Entity, Relation, Entity>* triple. In this paper, we will focus on the latter, identifying relationship between two existing KB entities. This task can be intuitively considered in a pipeline paradigm, that is, name mentions in the texts are first linked to entities in the KB (entity linking, EL), and then the relationship between them are identified (relation extraction, RE). It is worth mentioning that the first task EL is different from the task of named entity recognition (NER) in traditional information extraction (IE) tasks, where NER recognizes and classifies the entity mentions (to several predefined types) in the texts, but EL focuses on linking the mentions to their corresponding entities in the KB. Such pipeline systems often suffer from errors propagating from upstream to downstream, since only the local best results are selected to the next step. One idea to solve the problem is to allow interactions among the local predictions of both subtasks and jointly select an optimal assignment to eliminate possible errors in the pipeline.

Let us first look at an example. Suppose we are extracting knowledge facts from two sentences in Figure 1: in sentence [1], if we are more confident to extract the relation *fb:org.headquarters*[1], we will be then prompted to select *Bryant University*, which indeed favors the RE prediction that requires an organization to be its subject. On the other side, if we are sure to link to *Kobe Bryant* in sentence [2], we will probably select *fb:pro_athlete.teams*, whose subject position expects an athlete, e.g., an NBA player. It is not difficult to see that the argument type expectations of relations can encourage the two subtasks interact with each other and select coherent predictions for

[1]The prefix *fb* means the relations are defined in Freebase.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1912–1923,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

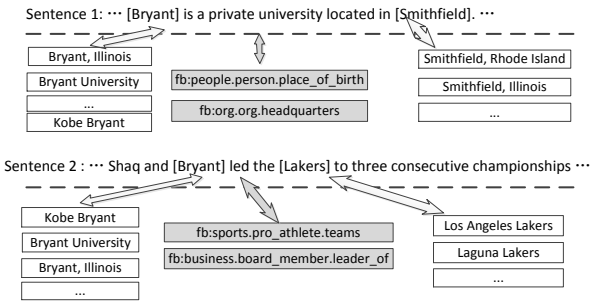


Figure 1: Two example sentences from which we can harvest knowledge facts.

both of them. In KBs with well-defined schemas, such as Freebase, type requirements can be collected and utilized explicitly (Yao et al., 2010). However, in other KBs with less reliable or even no schemas, it is more appropriate to implicitly capture the type expectations for a given relation (Riedel et al., 2013).

Furthermore, previous RE approaches usually process each triple individually, which ignores whether those local predictions are compatible with each other. For example, suppose the local predictions of the two sentences above are *<Kobe Bryant, fb:org.headquarters, Smithfield, Rhode Island>* and *<Kobe Bryant, fb:pro_athlete.teams, Los Angeles Lakers>*, respectively, which, in fact, disagree with each other with respect to the KB, since, in most cases, these two relations cannot share subjects. Now we can see that either the relation predictions or the EL results for "*Bryant*" are incorrect. Those disagreements provide us an effective way to remove the possible incorrect predictions that cause the incompatibilities.

On the other hand, the automatically extracted knowledge facts inevitably contain errors, especially for those triples collected from open domain. Extractions with confidence scores will be more than useful for users to make proper decisions according to their requirements, such as trading recall for precision, or supporting approximate queries.

In this paper, we propose a joint framework to populate an existing KB with new knowledge facts extracted from reliable text resources. The joint framework is designed to address the error propagation issue in a pipeline system, where subtasks are optimized in isolation and locally. We find an optimal configuration from top k results of both subtasks, which maximizes the scores of each step, fulfills the argument type expectations of relations, which can be captured explicitly or implicitly, in the KB, and avoids globally incoherent predictions. We formulate this optimization problem in an Integer Linear Program (ILP) framework, and further adopt a logistic regression model to measure the reliability of the whole process, and assign confidences to all extracted triples to facilitate further applications. The experiments on a real-world case study show that our framework can eliminate error propagations in the pipeline systems by taking relations' argument type expectations and global compatibilities into account, thus outperforms the pipeline approaches based on state-of-the-art relation extractors by a large margin. Furthermore, we investigate both explicit and implicit type clues for relations, and provide suggestions about which to choose according to the characteristics of existing KBs. Additionally, our proposed confidence estimations can help to achieve a precision of over 85% for a considerable amount of high quality extractions.

In the rest of the paper, we first review related work and then define the knowledge base population task that we will address in this paper. Next we detail the proposed framework and present our experiments and results. Finally, we conclude this paper with future directions.

2 Related Work

Knowledge base population (KBP), the task of extending existing KBs with entities and relations, has been studied in the TAC-KBP evaluations (Ji et al., 2011), containing three tasks. The entity linking task links entity mentions to existing KB nodes and creates new nodes for the entities absent in the current KBs, which can be considered as a kind of entity population (Dredze et al., 2010; Tamang et al., 2012; Cassidy et al., 2011). The slot-filling task populates new relations to the KB (Tamang et al., 2012; Roth et al., 2012; Liu and Zhao, 2012), but the relations are limited to a predefined sets of attributes according to the types of entities. In contrast, our RE models only require minimal supervision and do not need well-annotated training data. Our framework is therefore easy to adapt to new scenarios and suits real-world applications. The cold-start task aims at constructing a KB from scratch in a slot-filling style (Sun et al., 2012; Monahan and Carpenter, 2012).

Entity linking is a crucial part in many KB re-

lated tasks. Many EL models explore local contexts of entity mentions to measure the similarity between mentions and candidate entities (Han et al., 2011; Han and Sun, 2011; Ratinov et al., 2011; Cheng and Roth, 2013). Some methods further exploit global coherence among candidate entities in the same document by assuming that these entities should be closely related (Han et al., 2011; Ratinov et al., 2011; Sen, 2012; Cheng and Roth, 2013). There are also some approaches regarding entity linking as a ranking task (Zhou et al., 2010; Chen and Ji, 2011). Lin et al. (2012) propose an approach to detect and type entities that are currently not in the KB.

Note that the EL task in KBP is different from the name entity mention extraction task, mainly in the ACE task style, which mainly identifies the boundaries and types of entity mentions and does not explicitly link entity mentions into a KB (ACE, 2004; Florian et al., 2006; Florian et al., 2010; Li and Ji, 2014), thus are different from our work.

Meanwhile, relation extraction has also been studied extensively in recent years, ranging from supervised learning methods (ACE, 2004; Zhao and Grishman, 2005; Li and Ji, 2014) to unsupervised open extractions (Fader et al., 2011; Carlson et al., 2010). There are also models, with distant supervision (DS), utilizing reliable texts resources and existing KBs to predict relations for a large amount of texts (Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012). These distantly supervised models can extract relations from texts in open domain, and do not need much human involvement. Hence, DS is more suitable for our task compared to other traditional RE approaches.

Joint inference over multiple local models has been applied to many NLP tasks. Our task is different from the traditional joint IE works based in the ACE framework (Singh et al., 2013; Li and Ji, 2014; Kate and Mooney, 2010), which jointly extract and/or classify named entity mentions to several predefined types in a sentence and identify in a sentence level which relation this specific sentence describes (between a pair of entity mentions in this sentence). Li and Ji (2014) follow the ACE task definitions and present a neat incremental joint framework to simultaneously extract entity mentions and relations by structure perceptron. In contrast, we link entity mentions from a text corpus to their corresponding entities in an existing KB and identify the relations between pairs of entities based on that text corpus. Choi et al. (2006) jointly extracts the expressions and sources of opinion as well as the linking relations (i.e., a source entity expresses an opinion expression) between them, while we focus on jointly modeling EL and RE in open domain, which is a different and challenging task.

Since the automatically extracted knowledge facts inevitably contain errors, many approaches manage to assign confidences for those extracted facts (Fader et al., 2011; Wick et al., 2013). Wick et al. (2013) also point out that confidence estimation should be a crucial part in the automated KB constructions and will play a key role for the wide applications of automatically built KBs. We thus propose to model the reliability of the complete extraction process and take the argument type expectations of the relation, coherence with other predictions and the triples in the existing KB into account for each populated triple.

3 Task definition

We formalize our task as follows. Given a set of entities sampled from an existing KB, $E = \{e_1, e_2, ..., e_{|E|}\}$, a set of canonicalized relations from the same KB, $R = \{r_1, r_2, ..., r_{|R|}\}$, a set of sentences extracted from news corpus, $SN = \{sn_1, sn_2, ..., sn_{|SN|}\}$, each contains two mentions m_1 and m_2 whose candidate entities belong to E, a set of text fragments $\mathcal{T} = \{t_1, t_2, ..., t_{|\mathcal{T}|}\}$, where t_i contains its corresponding target sentence sn_i and acts as its context. Our task is to link those mentions to entities in the given KB, identify the relationship between entity pairs and populate new knowledge facts into the KB.

4 The Framework

We propose to perform joint inference over subtasks involved. For each sentence with two entity mentions, we first employ a preliminary EL model and RE model to obtain entity candidates and possible relation candidates between the two mentions, respectively. Our joint inference framework will then find an optimal assignment by taking the preliminary prediction scores, the argument type expectations of relations and the global compatibilities among the predictions into account. In the task of KBP, an entity pair may appear in multiple sentences as different relation instances, and the crucial point is whether we can identify all the cor-

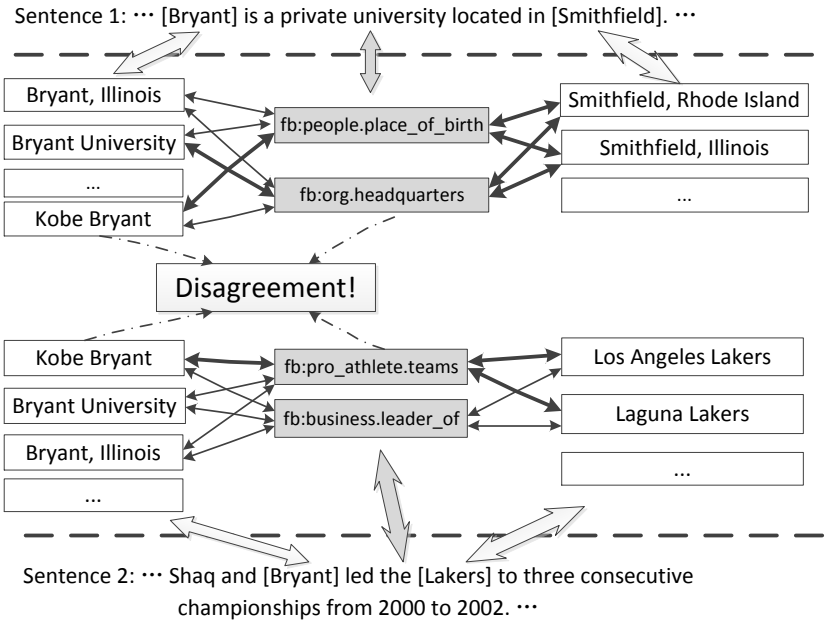


Figure 2: An example of our joint inference framework. The top and bottom are two example sentences with entity mentions in the square brackets, candidate entities in the white boxes, candidate relations in the grey boxes, and the solid lines with arrows between relations and entities represent their preference scores, with thickness indicating the preferences' value.

rect relations for an entity pair. Thus, after finding an optimal sentence-level assignment, we aggregate those local predictions by ORing them into the entity pair level. Finally, we employ a regression model to capture the reliability of the complete extraction process.

4.1 Preliminary Models

Entity Linking The preliminary EL model can be any approach which outputs a score for each entity candidate. Note that a recall-oriented model will be more than welcome, since we expect to introduce more potentially correct local predictions into the inference step. In this paper, we adopt an unsupervised approach in (Han et al., 2011) to avoid preparing training data. Note the challenging NIL problem, i.e., identifying which entity mentions do not have corresponding entities in the KB (labeled as NIL) and clustering those mentions, will be our future work. For each mention we retain the entities with top p scores for the succeeding inference step.

Relation Extraction The choice of RE model is also broad. Any sentence level extractor whose results are easy to be aggregated to entity pair level can be utilized here (again, a recall-oriented version will be welcome), such as Mintz++ mentioned in (Surdeanu et al., 2012), which we adapt into a Maximum Entropy version. We also include a special label, *NA*, to represent the case where there is no predefined relationship between an entity pair. For each sentence, we retain the relations with top q scores for the inference step, and we also call that this sentence **supports** those candidate relations. As for the features of RE models, we use the same features (lexical features and syntactic features) with the previous works (Chen et al., 2014; Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011).

4.2 Relations' Expectations for Argument Types

In most KBs' schemas, canonicalized relations are designed to expect specific types of entities to be their arguments. For example, in Figure 2, it is more likely that an entity *Kobe Bryant* takes the subject position of a relation *fb:pro_athlete.teams*, but it is unlikely for this entity to take the subject position of a relation *fb:org.headquarters*. Making use of these type requirements can encourage the framework to select relation and entity candidates which are coherent with each other, and discard incoherent choices.

In order to obtain the preference scores between

the entities in E and the relations in R, we generate two matrices with $|E|$ rows and $|R|$ columns, whose elements sp_{ij} indicates the preference score of entity i and relation j. The matrix S_{subj} is for relations and their subjects, and the matrix S_{obj} is for relations and their objects. We initialize the two matrices using the KB as follows: for entity i and relation j, if relation j takes entity i as its subject/object in the KB, the element at the position (i, j) of the corresponding matrix will be 1, otherwise it will be 0. Note that in our experiments, we do not count the triples that are evaluated in the testing data, to build the matrices. Now the problem is how we can obtain the unknown elements in the matrices.

Explicit Type Information Intuitively, we should examine whether the explicit types of the entities fulfill the expectations of relations in the KB. For each unknown element $S_{subj}(i, j)$, we first obtain the type of entity i, which is collected from the lowest level of the KB's type hierarchy, and examine whether there is another entity with the same type taking the subject position of relation j in the initial matrix. If such an entity exists, $S_{subj}(i, j) = 1$, otherwise 0. For example, for the subject *Jay Fletcher Vincent* and the relation *fb:pro_athlete.teams*, we first obtain the subject's type *basketball_player*, and then we go through the initial matrix and find another entity *Kobe Bryant* with the same type taking the subject position of *fb:pro_athlete.teams*, indicating that *Jay Fletcher Vincent* may take the relation *fb:pro_athlete.teams*. The matrix S_{obj} is processed in the same way.

Implicit Type Expectations In practice, few KBs have well-defined schemas. In order to make our framework more flexible, we need to come up with an approach to implicitly capture the relations' type expectations, which will also be represented as preference scores.

Inspired by Riedel et al. (2013) who use a matrix factorization approach to capture the association between textual patterns, relations and entities based on large text corpora, we adopt a collaborative filtering (CF) method to compute the preference scores between entities and relations based on the statistics obtained from an existing KB.

In CF, the preferences between customers and items are calculated via matrix factorization over the initial customer-item matrix. In our framework, we compute the preference scores between entities and relations via the same approach over the two initialized matrices S_{subj} and S_{obj}, resulting in two entity-relation matrices with estimated preference values. We use ALS-WR (Zhou et al., 2008) to process the matrices and compute the preference of a relation taking an entity as its subject and object, respectively. We normalize the preference scores of each entity using their means μ and standard deviations σ.

4.3 Compatibilities among Predicted Triples

The second aspect we investigate is whether the extracted triples are compatible with respect to all other knowledge facts. For example, according to the KB, the two relations *fb:org.headquarters* and *fb:pro_athlete.teams* in Figure 2 cannot share the same entity as their subjects. So if such sharing happens, that will indicate either the predictions of the relations or the entities are incorrect. The clues can be roughly grouped into three categories, namely whether two relations can share the same subjects, whether two relations can share the same objects, and whether one relation's subject can be the other relation's object.

Global compatibilities among local predictions have been investigated by several joint models (Li et al., 2011; Li and Ji, 2014; Chen et al., 2014) to eliminate the errors propagating in a pipeline system. Specifically, Chen et al. (2014) utilized the clues with respect to the compatibilities of relations in the task of relation extraction. Following (Li et al., 2011; Chen et al., 2014), we extend the idea of global compatibilities to the entity and relation predictions during knowledge base population. We examine the pointwise mutual information (PMI) between the argument sets of two relations to collect such clues. For example, if we want to learn whether two relations can share the same subject, we first collect the subject sets of both relations from the KB, and then compute the PMI value between them. If the value is lower than a certain threshold (set to -3 in this paper), the clue that *the two relations cannot share the same subject* is added. These clues can be easily integrated into an optimization framework in the form of constraints.

4.4 Integer Linear Program Formulation

Now we describe how we aggregate the above components, and formulate the joint inference problem into an ILP framework. For each candi-

date entity e of mention m in text fragment t, we define a boolean decision variable $d_t^{m,e}$, which denotes whether this entity is selected into the final configuration or not. Similarly, for each candidate relation r of fragment t, we define a boolean decision variable d_t^r. In order to introduce the preference scores into the model, we also need a decision variable $d_t^{r,m,e}$, which denotes whether both relation r and candidate entity e of mention m are selected in t.

We use $s_{el}^{t,m,e}$ to represent the score of mention m in t disambiguated to entity e, which is output by the EL model, $s_{re}^{t,r}$ representing the score of relation r assigned to t, which is output by the RE model, $s_p^{r,e}$ the explicit/implicit preference score between relation r and entity e.

Our goal is to find the best assignment to the variables d_t^r and $d_t^{m,e}$, such that it maximizes the overall scores of the two subtasks and the coherence among the preliminary predictions, while satisfying the constraints between the predicted triples as well. Our objective function can be written as:

$$\max el \times conf^{ent} + re \times conf^{rel} + sp \times coh^{e-r} \tag{1}$$

where el, re and sp are three weighting parameters tuned on development set. $conf^{ent}$ is the overall score of entity linking:

$$conf^{ent} = \sum_t \sum_{m \in M(t)} \sum_{e \in C_e(m)} s_{el}^{t,m,e} d_t^{m,e} \tag{2}$$

where $M(t)$ is the set of mentions in t, $C_e(m)$ is the candidate entity set of the mention m. $conf^{rel}$ represents the overall score of relation extraction:

$$conf^{rel} = \sum_t \sum_{r \in C_r(t)} s_{re}^{t,r} d_t^r \tag{3}$$

where $C_r(t)$ is the set of candidate relations in t. coh^{e-r} is the coherence between the candidate relations and entities in the framework:

$$coh^{e-r} = \sum_t \sum_{r \in C_r(t)} \sum_{m \in M(t)} \sum_{e \in C_e(m)} s_p^{r,e} d_t^{r,m,e} \tag{4}$$

Now we describe the constraints used in our ILP problem. The first kind of constraints is introduced to ensure that each mention should be disambiguated to only one entity:

$$\forall t, \forall m \in M(t), \quad \sum_{e \in C_e(m)} d_t^{m,e} \leq 1 \tag{5}$$

The second type of constraints ensure that each entity mention pair in one sentence can only take one relation label:

$$\forall t, \quad \sum_{r \in C_r(t)} d_r^t \leq 1 \tag{6}$$

The third is introduced to ensure the decision variable $d_t^{r,m,e}$ equals 1 if and only if both the corresponding variables d_t^r and $d_t^{m,e}$ equal 1.

$$\forall t, \forall r \in C_r(t), \forall m \in M(t), \forall e \in C_e(m)$$

$$d_t^{r,m,e} \leq d_t^r \tag{7}$$

$$d_t^{r,m,e} \leq d_t^{m,e} \tag{8}$$

$$d_t^r + d_t^{m,e} \leq d_t^{r,m,e} + 1 \tag{9}$$

As for the compatibility constraints, we need to introduce another type of boolean decision variables. If a mention m_1 in t_1 and another mention m_2 in t_2 share an entity candidate e, we add a variable y for this mention pair, which equals 1 if and only if both $d_{t_1}^{m_1,e}$ and $d_{t_2}^{m_2,e}$ equal 1. So we add the following constraints for each mention pair m_1 and m_2 satisfies the previous condition:

$$y \leq d_{t_1}^{m_1,e} \tag{10}$$

$$y \leq d_{t_2}^{m_2,e} \tag{11}$$

$$d_{t_1}^{m_1,e} + d_{t_2}^{m_2,e} \leq y + 1 \tag{12}$$

Then we further add the following constraints for each mention pair to avoid incompatible predictions:

$$\forall r_1 \in C_r(t_1), r_2 \in C_r(t_2)$$

$$\text{If } (r_1, r_2) \in \mathcal{C}^{sr}, p(m_1) = subj, p(m_2) = subj$$

$$d_{t_1}^{r_1} + d_{t_2}^{r_2} + y \leq 2 \tag{13}$$

$$\text{If } (r_1, r_2) \in \mathcal{C}^{ro}, p(m_1) = obj, p(m_2) = obj$$

$$d_{t_1}^{r_1} + d_{t_2}^{r_2} + y \leq 2 \tag{14}$$

$$\text{If } (r_1, r_2) \in \mathcal{C}^{sro}, p(m_1) = obj, p(m_2) = subj$$

$$d_{t_1}^{r_1} + d_{t_2}^{r_2} + y \leq 2 \tag{15}$$

where $p(m)$ returns the position of mention m, either $subj$ (subject) or obj (object). $\mathcal{C}^{sr}$ is the pairs of relations which cannot share the same subject, $\mathcal{C}^{ro}$ is the pairs of relations which cannot share the same object, $\mathcal{C}^{sro}$ is the pairs of relations in which one relation's subject cannot be the other one's object.

We use IBM ILOG Cplex[2] to solve the above ILP problem.

[2]http://www.cplex.com

Table 1: The features used to calculate the confidence scores.

Type	Feature
Real	The RE score of the relation.
Real	The EL score of the subject.
Real	The EL score of the object.
Real	The preference score between the relation and the subject.
Real	The preference score between the relation and the object.
Real	The ratio of the highest and the second highest relation score in this entity pair.
Real	The ratio of the current relation score and the maximum relation score in this entity pair.
Real	The ratio of the number of sentences supporting the current relation and the total number of sentences in this entity pair.
Real	Whether the extracted triple is coherent with the KB according to the constraints in Section 4.3.

4.5 Confidence Estimation for Extracted Triples

The automatically extracted triples inevitably contain errors and are often considered as with high recall but low precision. Since our aim is to populate the extracted triples into an existing KB, which requires highly reliable knowledge facts, we need a measure of confidence for those extracted triples, so that others can properly utilize them.

Here, we use a logistic regression model to measure the reliability of the process, how the entities are disambiguated, how the relationships are identified, and whether those predictions are compatible. The features we used are listed in Table 1, which are all efficiently computable and independent from specific relations or entities. We manually annotate 1000 triples as correct or incorrect to prepare the training data.

5 Experiments

We evaluate the proposed framework in a real-world scenario: given a set of news texts with entity mentions and a KB, a model should find more and accurate new knowledge facts between pairs of those entities.

5.1 Dataset

We use New York Times dataset from 2005 to 2007 as the text corpus, and Freebase as the KB. We divide the corpus into two equal parts, one for creating training data for the RE models using the distant supervision strategy (we do not need training data for EL), and the other as the testing data. For the convenience of experimentation, we randomly sample a subset of entities for testing. We first collect all sentences containing two mentions which may refer to the sampled entities, and prune them according to: (1)there should be no more than 10 words between the two mentions; (2)the prior probability of the mention referring to the target entity is higher than a threshold (set to 0.1 in this paper), which is set to filter the impossible mappings; (3)the mention pairs should not belong to different clauses. The resulting test set is split into 10 parts and a development set, each with 3500 entity pairs roughly, which leads to averagely 200,000 variables and 900,000 constraints per split and may take 1 hour for Cplex to solve. Note that we do not count the triples that will be evaluated in the testing data when we learn the preferences and the clues from the KB.

5.2 Experimental Setup

We compare our framework with three baselines. The first one, *ME-pl*, is the pipeline system constructed by the entity linker in (Han et al., 2011) and the MaxEnt version of Mintz++ extractor mentioned in (Surdeanu et al., 2012). The second and third baselines are the pipeline systems constructed by the same linker and two state-of-the-art DS approaches, MultiR (Hoffmann et al., 2011) and MIML-RE (Surdeanu et al., 2012), respectively. They are referred to as *MultiR-pl* and *MIML-pl* in the rest of this paper.

We also implement several variants of our framework to investigate the following two components in our framework: whether to use explicit (E) or implicit (I) argument type expectations, whether to take global (G) compatibilities into account, resulting in four variants: *ME-JE*, *ME-JI*, *ME-JEG*, *ME-JIG*.

We tune the parameters in the objective function on the development set to be $re = 1$, $el = 4$, $sp = 1$. The numbers of preliminary results retained to the inference step are set to $p = 2$, $q = 3$. Three metrics used in our experiments include: (1)the precision of extracted triples, which is the ratio of the number of correct triples and the number of total extracted triples; (2)the number of correct triples (NoC); (3)the number of correct triples in the results ranked in top n. The third metric is crucial for KBP, since most users are only interested in the knowledge facts with high confidences. We compare the extracted triples against

Table 2: The results of our joint frameworks and the three baselines.

Approach	Precision	NoC	Top 50	Top 100
ME-pl	28.7 ± 0.8	725 ± 12	38 ± 2	75 ± 4
MultiR-pl	31.0 ± 0.8	647 ± 15	39 ± 2	71 ± 3
MIML-pl	33.2 ± 0.6	608 ± 16	40 ± 3	74 ± 5
ME-JE	32.8 ± 0.7	768 ± 10	46 ± 2	90 ± 3
ME-JEG	34.2 ± 0.5	757 ± 8	46 ± 2	90 ± 3
ME-JI	34.5 ± 1.0	784 ± 9	43 ± 3	88 ± 3
ME-JIG	35.7 ± 1.0	772 ± 8	43 ± 3	88 ± 4

Freebase to compute the precision, which may underestimate the performance since Freebase is incomplete. Since we do not have exact annotations for the EL, it is difficult to calculate the exact recall. We therefore use NoC instead. We evaluate our framework on the 10 subsets of the testing dataset and compute their means and standard deviations.

5.3 Overall Performance

We are interested to find out: **(a)**whether the task benefits from the joint inference i.e., can we collect more and correct facts? Or with a higher precision? **(b)** whether the argument type expectations (explicit and implicit) and global compatibility do their jobs as we expected? And, how do we choose from these components ? **(c)**whether the framework can work with other RE models? **(d)**whether we can find a suitable approach to measure the confidence or uncertainty during the extraction so that users or other applications can better utilize the extracted KB facts?

Let us first look at the performance of the baselines and our framework in Table 2 for an overview. Comparing the three pipeline systems, we can discover that using the same entity linker, *MIML-pl* performs the best in precision with slightly fewer correct triples, while *ME-pl* performs the worst. It is not surprising, *ME-pl*, as a strong and high-recall baseline, outputs the most correct triples. As for the results with high confidences, *MultiR-pl* outputs more correct triples in the top 50 results than *ME-pl*, and *MIML-pl* performs better or comparable than *ME-pl* in top n results.

After performing the joint inference, *ME-JE* improves *ME-pl* with 4.1% in precision and 43 more correct triples averagely, and results in better performance in top n results. By taking global compatibilities into consideration, *ME-JEG* further improve the precision to 34.2% in average with slightly fewer correct triples, indicating that

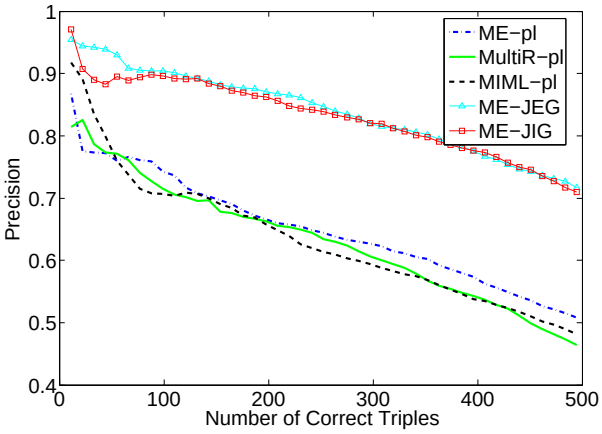


Figure 3: The numbers of correct triples v.s. the precisions for different approaches.

both argument type expectations and global compatibilities are useful in improving the performance: argument type information can help to select the correct and coherent predictions from the candidates EL and RE outputs, while global compatibilities can further prune incorrect triples that cause disagreements, although a few correct ones may be incorrectly eliminated. We can also observe that *ME-JIG* performs even higher than *ME-JEG* in overall precision, but *ME-JEG* collects more correct triples than *ME-JIG* in the top n predictions, showing that explicit type expectations with more accurate type information may perform better in high confidence results.

Furthermore, even though *MultiR-pl* and *MIML-pl* are based on state-of-the-art RE approaches, our model (for example, *ME-JIG*) can still outperform them in terms of all metrics, with 4.7% higher in precision than *MultiR-pl*, 2.5% higher than *MIML-pl*. Our model can extract 125 more correct triples than *MultiR-pl*, 164 more than *MIML-pl*, and perform better in top n results as well.

In previous RE tasks, Precision-Recall curves are mostly used to evaluate the systems' performances. In our task, since it is difficult to calculate the recall exactly, we use the number of correct triples instead, and plot curves of Precision-NoC to show the performance of the competitors and our approaches in more detail. For each value of NoC, the precision is the average of the ten splits of the testing dataset.

As shown in Figure 3, our approaches (*ME-JEG* and *ME-JIG*) obtain higher precisions on each NoC value, and the curves are much smoother than

Table 3: The results of our joint frameworks with MultiR sentence extractor.

Approach	Precision	NoC	Top 50	Top 100
MultiR-pl	31.0 ± 0.8	647 ± 15	39 ± 2	71 ± 3
MultiR-JEG	36.9 ± 0.8	687 ± 15	46 ± 2	88 ± 3
MultiR-JIG	38.5 ± 0.9	700 ± 15	45 ± 2	88 ± 3

the pipeline systems, indicating that our framework is more suitable for harvesting high quality knowledge facts. Comparing the two kinds of type clues, we can see that explicit ones perform better when the confidence control is high and the number of correct triples is small, and then the two are comparable. Since the precision of the triples with high confidences is crucial for the task of KBP, we still suggest choosing the explicit ones when there is a well-defined schema available in the KB, although implicit type expectations can result in higher overall precision.

5.4 Adapting MultiR Sentence Extractor into the Framework

The preliminary relation extractor of our framework is not limited to the MaxEnt[3] extractor. It can be any sentence level recall-oriented relation extractors. To further investigate the generalization of our joint inference framework, we also try to fit other sentence level relation extractors into the framework. Considering that MIML-RE does not output sentence-level results, we only adapt MultiR, with both global compatibilities and explicit/implicit type expectations, named as *MultiR-JEG* and *MultiR-JIG*, respectively. Since the scores output by the original MultiR are unnormalized, which are difficult to directly apply to our framework, we normalize their scores and retune the framework's parameters accordingly. The parameters are set to $re = 1$, $el = 32$, $sp = 16$.

As seen in Table 3, *MultiR-JEG* helps MultiR obtain about 40 more correct triples in average, and achieves 5.9% higher in precision, as well as significant improvements in top n correct predictions. As for *MultiR-JIG*, the improvements are 7.5% in precision and 53 in number of correct triples. In terms of top n results, the explicit and implicit type expectations perform comparable. We also observe that our framework improves MultiR as much as it does to MaxEnt, indicating our joint framework can generalize well in different RE models.

[3] http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html

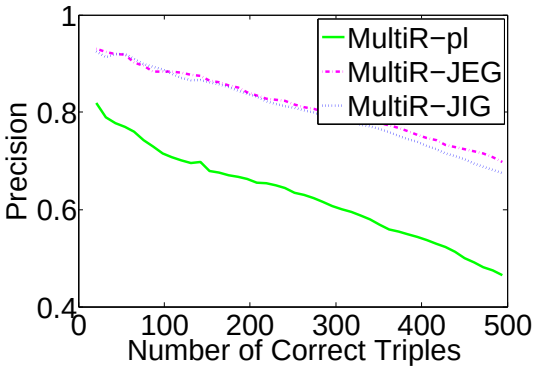


Figure 4: The numbers of correct triples v.s. the precisions for approaches with MultiR extractor.

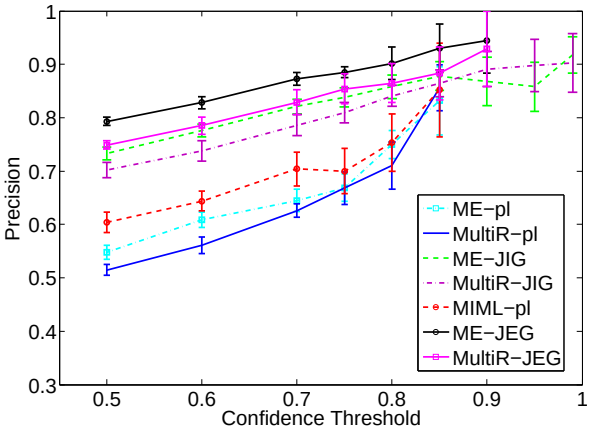


Figure 5: The precisions of different models under different confidence thresholds. The error bars represents the standard deviations of the results.

We further plot Precision-NoC curves for *MultiR-JEG* and *MultiR-JIG* in Figure 4, showing that our framework can result in better performance and smoother curves with MultiR extractor. It is interesting to see that with MultiR extractor, the two kinds of expectations perform comparably.

5.5 Results with Confidence Estimations

Now, we will investigate the results from another perspective with the help of confidence estimations. We calculate the precisions of the competitors and our approaches on different confidence thresholds from 0.5 to 1. The results are summarized in Figure 5. Note that the results across different approaches are not directly comparable, we put them in the same figure only to save space.

In Figure 5, intuitively, as the confidence threshold goes up, the extraction precisions should increase, indicating triples with higher confidences are more likely to be correct. However,

lower thresholds tend to result in estimations with smaller standard derivations due to those precisions are estimated over much more triples than those with higher thresholds, which means the randomness will be smaller.

On the other hand, our joint frameworks provide more evidences that can be used to well capture the reliability of an extraction. For example, the precisions of *Multir-JIG* and *ME-JIG* both stay around 85% when the confidence is higher than 0.85, with about 120 correct triples, indicating that by setting a proper threshold, we can obtain considerable amount of high quality knowledge facts at an acceptable precision, which is crucial for KBP. However, we cannot harvest such amount of high quality knowledge facts from the other three pipeline systems.

6 Conclusions

In this paper, we propose a joint framework for the task of populating KBs with new knowledge facts, which performs joint inference on two subtasks, maximizes their preliminary scores, fulfills the type expectations of relations and avoids global incompatibilities with respect to all local predictions to find an optimal assignment. Experimental results show that our framework can significantly eliminate the error propagations in pipeline systems and outperforms competitive pipeline systems with state-of-the-art RE models. Regarding the explicit argument type expectations and the implicit ones, the latter can result in a higher overall precision, while the former performs better in acquiring high quality knowledge facts with higher confidence control, indicating that if the KB has a well-defined schema we can use explicit type requirements for the KBP task, and if not, our model can still perform well by mining the implicit ones. Our framework can also generalize well with other preliminary RE models. Furthermore, we assign extraction confidences to all extracted facts to facilitate further applications. By setting a suitable threshold, our framework can populate high quality reliable knowledge facts to existing KBs.

For future work, we will address the NIL issue of EL where we currently assume all entities should be linked to a KB. It would be also interesting to jointly model the two subtasks through structured learning, instead of joint inference only. Currently we only use the coherence of extracted triples and the KB to estimate confidences, which would be nice to directly model the issue in a joint model.

Acknowledgments

We would like to thank Heng Ji, Kun Xu, Dong Wang and Junyang Rao for their helpful discussions and the anonymous reviewers for their insightful comments that improved the work considerably. This work was supported by the National High Technology R&D Program of China (Grant No. 2012AA011101, 2014AA015102), National Natural Science Foundation of China (Grant No. 61272344, 61202233, 61370055) and the joint project with IBM Research. Any correspondence please refer to Yansong Feng.

References

ACE. 2004. The automatic content extraction projects. http://projects.ldc.upenn.edu/ace.

Andrew Carlson, Justin Betteridge, Byran Kisiel, Burr Settles, Estevam Hruschka Jr., and Tom Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 1306–1313. AAAI Press.

Taylor Cassidy, Zheng Chen, Javier Artiles, Heng Ji, Hongbo Deng, Lev-Arie Ratinov, Jing Zheng, Jiawei Han, and Dan Roth. 2011. Entity linking system description. In *TAC2011*.

Zheng Chen and Heng Ji. 2011. Collaborative ranking: A case study on entity linking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 771–781, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liwei Chen, Yansong Feng, Songfang Huang, Yong Qin, and Dongyan Zhao. 2014. Encoding relation requirements for relation extraction via joint inference. In *Proceedings of the 52nd Annual Meeting on Association for Computational Linguistics*, ACL 2014, pages 818–827, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *EMNLP*.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 431–439, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity disambiguation for knowledge base population. In *Coling2010*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Radu Florian, Hongyan Jing, Nanda Kambhatla, and Imed Zitouni. 2006. Factorizing complex models: A case study in mention detection. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 473–480. Association for Computational Linguistics.

Radu Florian, John F Pitrelli, Salim Roukos, and Imed Zitouni. 2010. Improving mention detection robustness to noisy input. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 335–345. Association for Computational Linguistics.

Xianpei Han and Le Sun. 2011. A generative entity-mention model for linking entities with knowledge base. In *Proceedings of ACL*, HLT '11, pages 945–954, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: a graph-based method. In *SIGIR*, SIGIR '11, pages 765–774, New York, NY, USA. ACM.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th ACL-HLT - Volume 1*, HLT '11, pages 541–550, Stroudsburg, PA, USA. ACL.

Heng Ji, Ralph Grishman, and Hoa Dang. 2011. Overview of the tac2011 knowledge base population track. In *Proceedings of TAC*.

Rohit J. Kate and Raymond J. Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL '10, pages 203–212, Stroudsburg, PA, USA. Association for Computational Linguistics.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting on Association for Computational Linguistics*, ACL 2014, pages 402–412, Stroudsburg, PA, USA. Association for Computational Linguistics.

Qi Li, Sam Anzaroot, Wen-Pin Lin, Xiang Li, and Heng Ji. 2011. Joint inference for cross-document information extraction. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 2225–2228, New York, NY, USA. ACM.

Thomas Lin, Mausam, and Oren Etzioni. 2012. No noun phrase left behind: Detecting and typing unlinkable entities. In *Proceedings of the 2012 EMNLP-CoNLL*, EMNLP-CoNLL '12, pages 893–903, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fang Liu and Jun Zhao. 2012. Sweat2012: Pattern based english slot filling system for knowledge base population at tac 2012. In *TAC2012*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011.

Sean Monahan and Dean Carpenter. 2012. Lorify: A knowledge base from scratch. In *TAC2012*.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *ACL*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, volume 6323 of *Lecture Notes in Computer Science*, pages 148–163. Springer Berlin / Heidelberg.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. 2013. Relation extraction with matrix factorization and universal schemas. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL '13)*, June.

Benjamin Roth, Grzegorz Chrupala, Michael Wiegand, Mittul Singh, and Dietrich Klakow. 2012. Generalizing from freebase and patterns using cluster-based distant supervision for tac kbp slotfilling 2012. In *TAC2012*.

Prithviraj Sen. 2012. Collective context-aware topic models for entity disambiguation. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 729–738, New York, NY, USA. ACM.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, AKBC '13, pages 1–6, New York, NY, USA. ACM.

Ang Sun, Xin Wang, Sen Xu, Yigit Kiran, Shakthi Poornima, Andrew Borthwick, , and Ralph Grishman. 2012. Intelius-nyu tac-kbp2012 cold start system. In *TAC2012*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP-CoNLL*, pages 455–465. ACL.

Suzanne Tamang, Zheng Chen, and Heng Ji. 2012. Entity linking system and slot filling validation system. In *TAC2012*.

Michael Wick, Sameer Singh, Ari Kobren, and Andrew McCallum. 2013. Assessing confidence of knowledge base content with an experimental study in entity resolution. In *AKBC2013*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extraction without labelled data. In *Proceedings of EMNLP*, EMNLP '10, pages 1013–1023, Stroudsburg, PA, USA. ACL.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 419–426, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. 2008. Large-scale parallel collaborative filtering for the netflix prize. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management*, AAIM '08, pages 337–348, Berlin, Heidelberg. Springer-Verlag.

Yiping Zhou, Lan Nie, Omid Rouhani-Kalleh, Flavian Vasile, and Scott Gaffney. 2010. Resolving surface forms to wikipedia topics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1335–1343, Stroudsburg, PA, USA. Association for Computational Linguistics.

Combining Visual and Textual Features for Information Extraction from Online Flyers

Emilia Apostolova
BrokerSavant Inc
2506 N. Clark St.
Chicago, IL 60614
emilia@brokersavant.com

Noriko Tomuro
DePaul University
243 S. Wabash Ave.
Chicago, IL 60604
tomuro@cs.depaul.edu

Abstract

Information in visually rich formats such as PDF and HTML is often conveyed by a combination of textual and visual features. In particular, genres such as marketing flyers and info-graphics often augment textual information by its color, size, positioning, etc. As a result, traditional text-based approaches to information extraction (IE) could underperform. In this study, we present a supervised machine learning approach to IE from online commercial real estate flyers. We evaluated the performance of SVM classifiers on the task of identifying 12 types of named entities using a combination of textual and visual features. Results show that the addition of visual features such as color, size, and positioning significantly increased classifier performance.

1 Introduction

Since the Message Understanding Conferences in the 1990s (Grishman and Sundheim, 1996; Chinchor and Robinson, 1997), Information Extraction (IE) and Named Entity Recognition (NER) approaches have been applied and evaluated on a variety of domains and textual genres. The majority of the work, however, focuses on the journalistic, scientific, and informal genres (newswires, scientific publications, blogs, tweets, and other social media texts) (Nadeau and Sekine, 2007) and deals with purely textual corpora. As a result, the feature space of NER systems involves purely textual features, typically word attributes and characteristics (orthography, morphology, dictionary lookup, etc.), their contexts and document features (surrounding word window, local syntax, document/corpus word frequencies, etc.) (Nadeau and Sekine, 2007).

At the same time, textual information is often presented in visually rich formats, e.g. HTML and PDF. In addition to text, these formats use a variety of visually salient characteristics, (e.g. color, font size, positioning) to either highlight or augment textual information. In some genres and domains, a textual representation of the data, excluding visual features is often not enough to accurately identify named entities of interest or extract relevant information. Marketing materials, such as online flyers or HTML emails, often contain a plethora of visual features and text-based NER approaches lead to poor results. In this paper, we present a supervised approach that uses a combination of textual and visual features to recognize named entities in online marketing materials.

2 Motivation and Problem Definition

A number of broker-based industries (e.g. commercial real estate, heavy equipment machinery, etc.) lack a centralized searchable database with industry offerings. In particular, the commercial real estate industry (unlike residential real estate) does not have a centralized database or an established source of information. Commercial real estate brokers often need to rely on networking, chance, and waste time with a variety of commercial real estate databases that often present outdated information. While brokers do not often update third party inventory databases, they do create marketing materials (usually PDF flyers) that contain all relevant listing information. Virtually all commercial real estate offerings come with publicly available marketing material that contains all relevant listing information. Our goal is to harness this source of information (the marketing flyer) and use it to extract structured listing information.

Figure 1 shows an example of a commercial real estate flyer. The commercial real estate flyers are often distributed as PDF documents, links to HTML pages, or visually rich HTML-based

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1924–1929,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Figure 1: An example of a commercial real estate flyer © Kudan Group Real Estate.

emails. They typically contain all relevant listing information such as the address and neighborhood of the offering, the names and contact information of the brokers, the type of space offered (building, land, unit(s) within a building), etc. Similar to other info-graphics, relevant information could be easily pinpointed by visual clues. For example, the listing street address in Figure 1 (*1629 N. Halsted St.*, upper left corner) can be quickly identified and distinguished from the brokerage firm street address (*156 N. Jefferson St.*, upper right corner) due to its visual prominence (font color, size, positioning).

In this study we explored a supervised machine learning approach to the task of identifying listing information from commercial real estate flyers. In particular, we focused on the recognition of 12 types of named entities as described in Table 1 below.

3 Related Work

Nadeau and Satoshi (2007) present a survey of NER and describe the feature space of NER research. While they mention multi-media NER in the context of video/text processing, all described features/approaches focus only on textual representation.

Broker Name Broker Email Broker Phone Company Phone	The contact information of all listing brokers, including full name, email address, phone number. The brokerage company phone number.
Street City Neighborhood State Zip	The address information of the listing address including street or intersection, city, neighborhood, state, and zip code.
Space Size Space Type	Size and attributes of relevant spaces (e.g. *27,042 SF building, 4.44 acres site*, etc.); Mentions of space type descriptors, e.g. building, land/lot, floor, unit. This excludes space type and size information of non-essential listing attributes (e.g. basement size or parking lot size).
Confidential	Any mentions of confidentiality.

Table 1: Types and descriptions of named entities relevant to extracting listing information from commercial real estate flyers.

The literature on Information Extraction from HTML resources is dominated by various approaches based on wrapper induction (Kushmerick, 1997; Kushmerick, 2000). Wrapper inductions rely on common HTML structure (based on the HTML DOM) and formatting features to extract structured information from similarly formatted HTML pages. This approach, however, is not applicable to the genres of marketing materials (PDF and HTML) since they typically do not share any common structure that can be used to identify relevant named entities. Laender et al. (2002) present a survey of data extraction techniques and tools from structured or semi-structured web resources.

Cai et al. (2003) present a vision-based segmentation algorithm of web pages that uses HTML layout features and attempts to partition the page at the semantic level. In (Burget and Rudolfova, 2009) authors propose web-page block classification based on visual features. Yang and Zhang (2001) build a content tree of HTML documents based on *visual consistency* inferred semantics. Burget (2007) proposes a layout based information extraction from HTML documents and states that this visual approach is more robust than traditional DOM-based methods.

Changuel et al.(2009a) describe a system for automatically extracting author information from web-pages. They use spatial information based on the depth of the text node in the HTML DOM tree. In (Changuel et al., 2009b) and (Hu et al., 2006),

the authors proposed a machine learning method for title extraction and utilize format information such as font size, position, and font weight. In (Zhu et al., 2007) authors use layout information based on font size and weight for NER for automated expense reimbursement.

While the idea of utilizing visual features based on HTML style has been previously suggested, this study tackles a non-trivial visually rich dataset that prevents the use of previously suggested simplistic approaches to computing HTML features (such as relying on the HTML DOM tree or simplistic HTML style rendering). In addition, we introduce the use of RGB color as a feature and normalize it approximating human perception.

4 Dataset and Method

The dataset consists of 800 randomly selected commercial real estate flyers spanning 315 US locations, 75 companies, and 730 brokers. The flyers were collected from various online sources and were originally generated using a variety of HTML and PDF creator tools. The collection represents numerous flyer formats and layouts, commercial real estate property types (*industrial, retail, office, land*, etc.), and transactions (*investment, sale, lease*).

All flyers were converted to a common format (HTML)[1]. The HTML versions of all documents were then annotated by 2 annotators. Figure 2 shows an example of an annotated flyer. Annotation guidelines were developed and the 2 annotators were able to achieve an inter-annotator agreement of 91%[2]. The named entities with lowest inter-annotator agreement were entities describing Space Size and Type because of the somewhat complex rules for determining essential listing space information. For example, one of the space size/type rules reads as follows: *If the listing refers to a building and mentions the lot size, include both the land size, the building size, and corresponding space types. Do not include individual parts of the building (e.g. office/basement) as separate spaces. If the listing refers to a UNIT within the building, not the whole building, then DO NOT include the land site as a separate space.*

A supervised machine learning approach was

[1]PDFs were converted to HTML using the PDFTOHTML conversion program `http://pdftohtml.sourceforge.net/`.

[2]The inter-annotator agreement was measured as F1-score using one of the annotator's named entities as the gold standard set and the other as a comparison set.

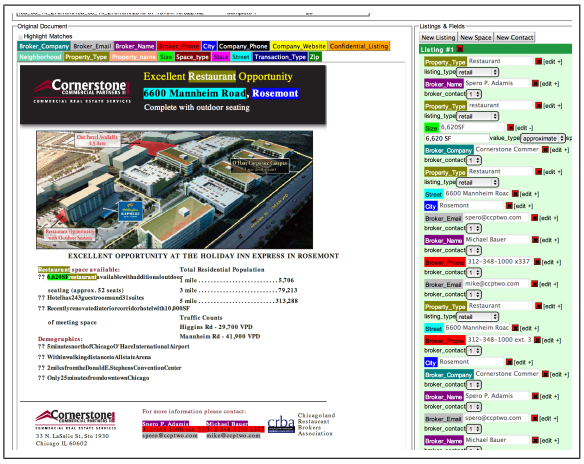


Figure 2: The HTML versions of the flyers were annotated by 2 annotators using a custom web-based annotation tool.

then applied to the task of identifying the 12 named entities shown in Table 1. Flyers were converted to text using an HTML parser while preserving some of the white space formatting. The text was tokenized and the task was then modeled as a **BIO** classification task, classifiers identify the **B**eginning, the **I**nside, and **O**utside of the text segments. We first used a traditional set of text-based features for the classification task. Table 2 lists the various text-based features used. In all cases, a sliding window including the 5 preceding and 5 following tokens was used as features.

Feature Name	Description
Token	A normalized string representation of the token. All tokens were converted to lower case and all digits were converted to a common format.
Token Orth	The token orthography. Possible values are lowercase (all token characters are lower case), all capitals (all token characters are upper case), upper initial (the first token character is upper case, the rest are lower case), mixed (any mixture of upper and lower case letters not included in the previous categories).
Token Kind	Possible values are word, number, symbol, punctuation.
Regex type	Regex-based rules were used to mark chunks as one of 3 regex types: email, phone number, zip code.
Gazetteer	Text chunks were marked as possible US cities or states based on US Census Bureau city and state data. www.census.gov/geo/maps-data/data/gazetteer2013.html.

Table 2: List of text-based features used for the NER task. A sliding window of the 5 preceding and 5 following tokens was used for all features.

As noted previously, human annotators were able to quickly spot named entities of interest solely because of their visual characteristics. For example, a text-only version of the flyer shown in Figure 1, stripped of all rich formatting, will make it quite difficult to distinguish the listing address (shown in prominent size, position, and color) from the brokerage company address, which is rarely prominent as it is not considered important information in the context of the flyer. Similarly, the essential size information for the listing shown on Figure 2 appears prominently on the first page (square footage of the offered restaurant), while non-essential size information, such as the size of the adjacent parking lot or basement, tend to appear in smaller font on subsequent flyer pages.

To account for such visual characteristics we attempted to also include visual features associated with text chunks. We used the computed HTML style attributes for each DOM element containing text. Table 3 lists the computed visual features.

Feature Name	Description
Font Size	The computed *font-size* attribute of the surrounding HTML DOM element, normalized to 7 basic sizes (*xx-small, x-small, small, medium, large, x-large, xx-large*).
Color	The computed *color* attribute of the surrounding HTML DOM element. The RGB values were normalized to a set of 100 basic colors. We converted the RGB values to the YUV color space, and then used Euclidian distance to find the most similar basic color approximating human perception.
Y Coordinate	The computed *top* attribute of the surrounding HTML DOM element, i.e. the y-coordinate in pixels. The pixel locations was normalized to 150 pixel increments (roughly 1/5th of the visible screen for the most common screen resolution.)

Table 3: List of visual features used for the NER task. A sliding window of 5 preceding and 5 following DOM elements were used for all features.

Computing the HTML style attributes is a complex task since they are typically defined by a combination of CSS files, in-lined HTML style attributes, and browser defaults. The complexities of style definition, inheritance, and overwriting are handled by browsers[3]. We used the Chrome browser to compute dynamically the style of each DOM element and output it as inline style attributes. To achieve this we programmatically inserted a javascript snippet that inlines the computed style and saves the new version of the HTML on the local file system utilizing the HTML5 *saveAs* interface[4]. Details on how we normalized the style attribute values for font size, RGB color, and Y coordinate are shown in Table 3.

We then applied Support Vector Machines (SVM) (Vapnik, 2000) on the NER task using the LibSVM library (Chang and Lin, 2011). We chose SVMs as they have been shown to perform well on a variety of NER tasks, for example (Isozaki and Kazawa, 2002; Takeuchi and Collier, 2002; Mayfield et al., 2003; Ekbal and Bandyopadhyay, 2008). We used a linear kernel model with the default parameters. The multi-class problem was converted to binary problems using the one-vs-others scheme. 80% of the documents were used for training, and the remaining 20% for testing.

5 Results

Results are shown in Table 4. We compared classifier performance using only textual features (first 3 columns), versus performance using both textual and visual features (next 3 columns). Results were averaged over 2 runs of randomly selected training/test documents with 80%/20% ratio. We used an exact measure which considers an answer to be correct only if both the entity boundaries and entity type are accurately predicted.

The addition of visual features significantly[5] increased the overall F1-score from 83 to 87%. As expected, performance gains are more significant for named entities that are typically visually salient and are otherwise difficult (or impossible) to identify in a text-only version of the flyers. Named Entities referring to listing address information showed the most significant improvements. In particular, the F1-score for mentions of *Neighborhoods* (typically prominently shown on the first page of the flyers) improved by 19%; F1-score for mentions of the listing *State* improved by 9%; and *Street, City, Zip* by roughly 4% each, all

[3]We attempted to use an HTML renderer from the Cobra java toolkit `http://lobobrowser.org/cobra.jsp` to compute HTML style attributes. However, this renderer produced poor results on our dataset and failed to accurately compute the pixel location of text elements.

[4]`https://github.com/eligrey/FileSaver.js`

[5]The difference is statistically significant with p value $<$ 0.0001% using Z-test on two proportions.

Named Entity	Pt	Rt	Ft	Pv+t	Rv+t	Fv+t	S
Broker Name	82.7	91.7	87.0	95.0	91.6	93.2	Y
Broker Email	92.3	92.8	92.6	97.2	90.2	93.6	N
Broker Phone	90.2	86.1	88.1	94.7	85.2	89.7	N
Company Ph.	95.2	67.4	78.9	89.8	65.4	75.7	N
Street	87.4	70.5	78.1	87.3	77.3	82.0	Y
City	92.5	88.5	90.5	94.9	92.8	93.8	Y
Neighborhood	68.2	52.8	59.5	85.3	72.9	78.6	Y
State	77.4	97.5	86.3	95.8	95.0	95.4	Y
Zip	89.7	94.5	92.1	96.1	97.1	96.6	Y
Space Size	80.2	65.0	71.8	87.0	70.6	77.9	Y
Space Type	76.0	74.7	75.3	78.6	72.2	75.3	N
Confidential	100	60.0	75.0	75.0	85.7	79.9	N
OVERALL	84.8	81.3	83.0	91.2	83.2	87.0	Y

Table 4: Results from applying SVM using the textual features described in Table 2, as well as both the textual and visual features described in Tables 2 and 3. t=textual features only, v+t=visual + textual features, P=Precision, R=Recall, F=F1-score, S=Significant Difference

statistically significant. Visual clues are also typically used when identifying relevant size information and, as expected, performance improved significantly by roughly 6%. The difference in performance for mentions used to describe confidential information is not statistically significant[6] because such mentions rarely occurred in the dataset. Similarly, performance differences for Company Phone, Broker Phone, Broker Email, and Space Type are not statistically significant. In all of these cases, visual features did not influence performance and text-based features proved adequate predictors.

6 Conclusion

We have shown that information extraction in certain genres and domains spans different media - textual and visual. Ubiquitous online and digital formats such as PDF and HTML often exploit the interaction of textual and visual elements. Information is often augmented or conveyed by non-textual features such as positioning, font size, color, and images. However, traditionally, NER approaches rely exclusively on textual features and as a result could perform poorly in visually rich genres such as online marketing flyers or infographics. We have evaluated the performance gain on the task of NER from commercial real estate flyers by adding visual features to a set of traditional text-based features. We used SVM classifiers for the task of identifying 12 types of named entities. Results show that overall visual features improved performance significantly.

[6] p value = 0.7323% using Z-test on two proportions.

References

Radek Burget and Ivana Rudolfova. 2009. Web page element classification based on visual features. In *Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on*, pages 67–72. IEEE.

Radek Burget. 2007. Layout based information extraction from html documents. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 624–628. IEEE.

Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. 2003. Extracting content structure for web pages based on visual representation. In *Web Technologies and Applications*, pages 406–417. Springer.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Sahar Changuel, Nicolas Labroche, and Bernadette Bouchon-Meunier. 2009a. Automatic web pages author extraction. In *Flexible Query Answering Systems*, pages 300–311. Springer.

Sahar Changuel, Nicolas Labroche, and Bernadette Bouchon-Meunier. 2009b. A general learning method for automatic title extraction from html pages. In *Machine Learning and Data Mining in Pattern Recognition*, pages 704–718. Springer.

Nancy Chinchor and Patricia Robinson. 1997. Muc-7 named entity task definition. In *Proceedings of the 7th Conference on Message Understanding*.

Asif Ekbal and Sivaji Bandyopadhyay. 2008. Named entity recognition using support vector machine: A language independent approach. *International Journal of Computer Systems Science & Engineering*, 4(2).

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference-6: A brief history. In *COLING*, volume 96, pages 466–471.

Yunhua Hu, Hang Li, Yunbo Cao, Li Teng, Dmitriy Meyerzon, and Qinghua Zheng. 2006. Automatic extraction of titles from general documents using machine learning. *Information processing & management*, 42(5):1276–1293.

Hideki Isozaki and Hideto Kazawa. 2002. Efficient support vector classifiers for named entity recognition. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Nicholas Kushmerick. 1997. *Wrapper induction for information extraction*. Ph.D. thesis, University of Washington.

Nicholas Kushmerick. 2000. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1):15–68.

Alberto HF Laender, Berthier A Ribeiro-Neto, Altigran S da Silva, and Juliana S Teixeira. 2002. A brief survey of web data extraction tools. *ACM Sigmod Record*, 31(2):84–93.

James Mayfield, Paul McNamee, and Christine Piatko. 2003. Named entity recognition using hundreds of thousands of features. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 184–187. Association for Computational Linguistics.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Koichi Takeuchi and Nigel Collier. 2002. Use of support vector machines in extended named entity recognition. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–7. Association for Computational Linguistics.

Vladimir Vapnik. 2000. *The nature of statistical learning theory*. springer.

Yudong Yang and HongJiang Zhang. 2001. Html page analysis based on visual cues. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 859–864. IEEE.

Guangyu Zhu, Timothy J Bethea, and Vikas Krishna. 2007. Extracting relevant named entities for automated expense reimbursement. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1004–1012. ACM.

CTPs: Contextual Temporal Profiles for Time Scoping Facts using State Change Detection

Derry Tanti Wijaya
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213
dwijaya@cs.cmu.edu

Ndapandula Nakashole
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213
ndapa@cs.cmu.edu

Tom M. Mitchell
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213
tom.mitchell@cs.cmu.edu

Abstract

Temporal scope adds a time dimension to facts in Knowledge Bases (KBs). These time scopes specify the time periods when a given fact was valid in real life. Without temporal scope, many facts are underspecified, reducing the usefulness of the data for upper level applications such as Question Answering. Existing methods for temporal scope *inference* and *extraction* still suffer from low accuracy. In this paper, we present a new method that leverages temporal profiles augmented with context— Contextual Temporal Profiles (CTPs) of entities. Through change patterns in an entity's CTP, we model the entity's state change brought about by real world events that happen to the entity (e.g, hired, fired, divorced, etc.). This leads to a new formulation of the temporal scoping problem as a *state change detection problem*. Our experiments show that this formulation of the problem, and the resulting solution are highly effective for inferring temporal scope of facts.

1 Introduction

Recent years have seen the emergence of large Knowledge Bases (KBs) of facts (Carlson 2010; Auer 2007; Bollacker 2008; Suchanek 2007). While the wealth of accumulated facts is huge, most KBs are still sparsely populated in terms of temporal scope. Time information is an important dimension in KBs because knowledge is not static, it changes over time: people get divorced; countries elect new leaders; and athletes change teams. This means that facts are not always indefinitely true. Therefore, temporal scope has crucial implications for KB accuracy.

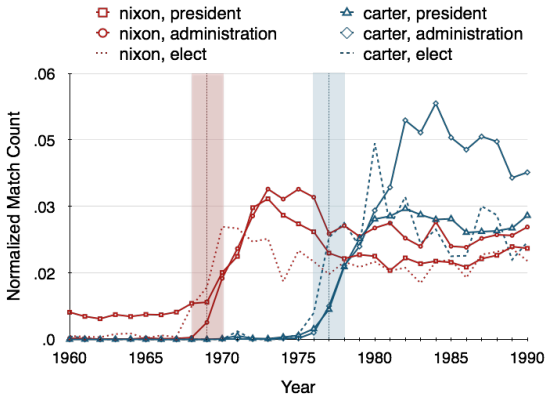


Figure 1: Behavior patterns of context uni-grams for the US presidency state change as seen in the Google Books N-grams corpus: the rise of 'elect', immediately followed by the rise of 'administration' and 'president'.

Towards bridging the time gap in KBs, we propose a new method for temporal scope inference. Our method is based on leveraging aggregate statistics from a time-stamped corpus. First we generate Contextual Temporal Profiles (CTPs) of entities from contexts surrounding mentions of these entities in the corpus. We then detect change patterns in the CTPs. We then use these changes to determine when a given entity undergoes a specific state change caused by real world events. Our main insight is as follows: events that happen to an entity change the entity's state and therefore its facts. Thus by learning *when* a given entity undergoes a specific state change, we can directly infer the time scopes of its facts. For example, in the *divorce event*, the person's state changes from 'married' to 'divorced' hence the *hasSpouse* relation no longer applies to it, signaling the *end* time of its current *hasSpouse* value. In a *country election event*, the country's state changes and it obtains a new value for its *hasPresident* relation.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1930–1936,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Our method involves learning context units (uni-grams and bi-grams surrounding mentions of an entity) that are relevant to a given state change. For this we use a few seed examples of entities that have gone through the state change. For example, for the US presidency state change denoting the beginning of a US presidency, given seed examples such as *(Richard Nixon, 1969)* and *(Jimmy Carter, 1977)*, relevant context units include unigrams such as 'administration' and 'elect', which are common to both CTPs in 1969 and 1977 respectively. Secondly, we learn the mention behavior of these context units for an entity undergoing a given state change (section 3 has more details). Figure 1 shows a motivating example, we see the behavior patterns of context uni-grams for the US presidency state change: the rise of 'elect' at the beginning of presidencies, immediately followed by the rise of 'administration' and 'president' in the context of the entities, *Nixon* and *Carter*.

2 Related work

Prior work mainly falls into two categories: i) methods that extract temporal scope from text, at the time of fact extraction; ii) methods that infer temporal scope from aggregate statistics in large Web corpora. Early methods mostly fall under category i); Timely YAGO (Wang 2010), TIE (Ling 2010), and PRAVDA (Wang 2011) are three such methods. Timely YAGO applies regular expressions to Wikipedia infoboxes to extract time scopes. It is therefore not applicable to any other corpora but Wikipedia. The TIE (Ling 2010) system produces a maximal set of events and their temporal relations based on the text of a given sentence. PRAVDA uses textual patterns along with a graph-based re-ranking method. Methods falling under category i) have the downside that it is unclear how they can be applied to facts that are already in the knowledge base. Only one other approach learned time scopes from aggregate corpus statistics, a recent system called CoTS (Talukdar 2012b). CoTS uses temporal profiles of facts and how the mentions of such facts rise and fall over time. However, CoTS is based on frequency counts of fact mentions and does not take into account state change inducing context. For example, to find the time scope of Nixon presidency, CoTS uses the rise and fall of the mention 'nixon' and 'president' over time. To improve accuracy, CoTS combined this frequency signal with manually supplied constraints such as the functionality of the US presidency relation to scope the beginning and end of Nixon presidency. In contrast, the proposed system does not require constraints as input.

There have also been tools and competitions developed to facilitate temporal scope extraction. TARSQI (Verhagen 2005) is a tool for automatically annotating time expressions in text. The TempEval (Verhagen 2007) challenge has led to a number of works on temporal relation extraction (Puscasu 2007; Yoshikawa 2009; Bethard 2007).

3 Method

Given an entity and its Contextual Temporal Profile (CTP), we can learn *when* such an entity undergoes a specific state change. We can then directly infer the *begin* or *end* time of the fact associated with the state change.

The CTP of an entity at a given time point t contains the context within which the entity is mentioned at that time. Our method is based on two related insights: i) the context of the entity at time t reflects the events happening to the entity and the state of the entity at time t. ii) the *difference in context* before, at time $t-1$, and after, at time t, reflect the associated state change at time t. However an entity can undergo a multiplicity of changes at the same time. Thus both the contexts and the differences in contexts can contain information pertaining to several state changes. We therefore need a way of determining which part of the context is relevant to a given state change sc_i. To this end, we generate what we refer to as an $aggregate$ state vector, $Vs(\overline{e}, sc_i)$ for a hypothetical average entity $\overline{e}$ undergoing state change sc_i. We generate $Vs(\overline{e}, sc_i)$ from the CTPs of a seed set of entities at the time they undergo state change sc_i.

3.1 Learning State and State Change Vectors

To build CTPs for entities, we use two time-stamped corpora: the Google Books Ngram corpus (Michel 2011); and the English Gigaword (Graff 2003) corpus. The Google Books Ngram corpus contains n-grams for $n = 1-5$; along with occurrence statistics from over about 5 million digitized books. The English Gigaword (Graff

2003) corpus contains newswire text from 1994-2008. From these corpora, we use the time granularity of a year as it is the finest granularity common to both corpora.

Definition 1 (Contextual Temporal Profile) *The Contextual Temporal Profile (CTP) of an entity e at time t, $C_e(t)$, consists of the context within which e is mentioned. Specifically $C_e(t)$ consists of uni-grams and bi-grams generated from the 5-grams(Google Books Ngram) or sentences (Gigaword) that mention e at time t.*

Notice that the CTPs can contain context units (bi-grams or uni-grams) that are simply noise. To filter the noise, we compute *tf-idf* statistics for each contextual unit and only retain the top k ranking units in $C_e(t)$. In our experiments, we used $k = 100$. We compute *tf-idf* by treating each time unit t as a document containing words that occur in the context of e (Wijaya 2011).

Furthermore, CTPs may contain context units attributed to several state changes. We therefore tease apart the CTPs to isolate contexts specific to a given state change. For this, our method takes as input a small set of seed entities, $\mathcal{S}(sc_i)$, for each type of state change. Thus for the US presidency state change that denotes the beginning of a US presidency, we would have seeds as follows: *(Richard Nixon, 1969)*, *(Jimmy Carter, 1977)*. From the CTPs of the seeds for state change sc_i, we generate an aggregate state vector, $Vs(\overline{e}, sc_i)$. To obtain the few dozen seeds required by our method, one can leverage semi-structured sources such as Wikipedia *infoboxes*, where relations e.g., *spouse* often have time information.

Definition 2 (Aggregate State Vector for $\overline{e}$) *The aggregate state vector of a mean entity $\overline{e}$ for state change sc_i, $Vs(\overline{e}, sc_i)$, is made up of the contextual units from the CTPs of entities in the seed set $\mathcal{S}(sc_i)$ that undergo state change sc_i. Thus, we have: $Vs(\overline{e}, sc_i) = \frac{1}{|\mathcal{S}(sc_i)|} \sum_{e,t:(e,t)\in\mathcal{S}(sc_i)} C_e(t)$.*

Thus, the state vector $Vs(\overline{e}, sc_i)$ reflects events happening to $\overline{e}$ and the state of $\overline{e}$ at the time it undergoes the state change sc_i. Additionally, we compute another type of aggregate vector, *aggregate change vector* $\triangle Vs(\overline{e}, sc_i)$ to capture the change patterns in the context units of $\overline{e}$. Recall that context units rise or fall due to state change, as seen earlier in Figure 1.

Definition 3 (Aggregate Change Vector for $\overline{e}$) *The aggregate change vector of a mean entity $\overline{e}$ for state change sc_i, $\triangle Vs(\overline{e}, sc_i)$, is made up of the change in the contextual units of the CTPs of entities in the seed set $\mathcal{S}(sc_i)$ that undergo state change sc_i. Thus, we have: $\triangle Vs(\overline{e}, sc_i) = \frac{1}{|\mathcal{S}(sc_i)|} \sum_{e,t:(e,t)\in\mathcal{S}(sc_i)} C_e(t) - C_e(t-1)$.*

The aggregate state vector $Vs(\overline{e}, sc_i)$ and the aggregate change vector $\triangle Vs(\overline{e}, sc_i)$ are then used to detect state changes.

3.2 Detecting State Changes

To detect state changes in a previously unseen entity e_{new}, we generate its state vector, $C_{e_{new}}(t)$, and its change vector, $\triangle C_{e_{new}}(t)$ = $C_{e_{new}}(t)$ - $C_{e_{new}}(t-1)$, for every time point t. We consider every time point t in the CTP of the new entity to be a candidate for a given state change sc_i, which we seek to determine whether e_{new} goes through and at which time point. We then compare the state vector and change vector of every candidate time point t to the aggregate state and aggregate change vector of state change sc_i. We use cosine similarity to measure similarities between the state vector and the aggregate state vector and between the change vector and the aggregate change vector. To combine these two vector similarities, we sum the state vector and change vector similarities. In future we can explore cross validation and a separate development set to define a weighted sum for combining these two similarities.

The highest ranking candidate time point (most similar to the aggregate state and aggregate change vector) is then considered to be the start of state change sc_i for the new entity e_{new}.

4 Experiments

We carried out experiments to answer the following questions: *Is treating temporal scoping as state change detection in Contextual Temporal Profiles(CTPs) effective? Do CTPs help improve temporal scope extraction over context-unaware temporal profiles?*

4.1 Methods under Comparison

We answer these questions by comparing to the following methods.

1. **CoTS** a state-of-the-art temporal scoping system (Talukdar 2012b)

2. **MaxEnt** a baseline to which CoTS was compared. It is a Maximum Entropy classifier trained separately for each relation using normalized counts and gradients of facts as features. An Integer Linear Program (ILP) is used to predict which facts are active at which times. This is done based on the output of the MAXENT classifier together with temporal intra-relation constraints that regulate the temporal scoping of one or more facsts from a *single* relation (e.g., FUNCTIONAL constraints on US President relation that regulate that at most one fact from the relation can be true at any given time i.e., there is only one US President at any given time).

3. **MaxEnt + Intra Relation Constraints** MaxEnt with cross relation constraints added: constraints that couple facts from *multiple* relations e.g., a constraint that *Al Gore*'s vice presidency is aligned exactly with *Bill Clinton*'s presidency.

We evaluate on the same set of facts as CoTS and its baselines: facts from the US Administration domain (US President, US Vice President, and US Secretary of State); and facts from the Academy Awards domain (Best Director and Best Picture). The number of facts per relation are as follows: US President, 9; US Vice President, 12; US Secretary of State, 13; Best Director, 14; and Best Picture, 14. Our method however is not specific to these relations from these two domains. Since our method does not depend on temporal constraints, the method can work a very different domain, for example one where many facts can exist for any time span without being superseded by another, as long as the entities involved experience a change of state. Thus, it can be applied to relations like *spouse*, even though many people are married in a year as these people change state from *single* or *engaged* to *married*.

Similar to CoTS, the datasets from which the CTPs were generated are as follows: The Google Books Ngram (1960-2008) dataset (Michel 2011) for the US Administration domain and the English Gigaword (1994-2008) dataset (Graff 2003) for Academy Award domain.

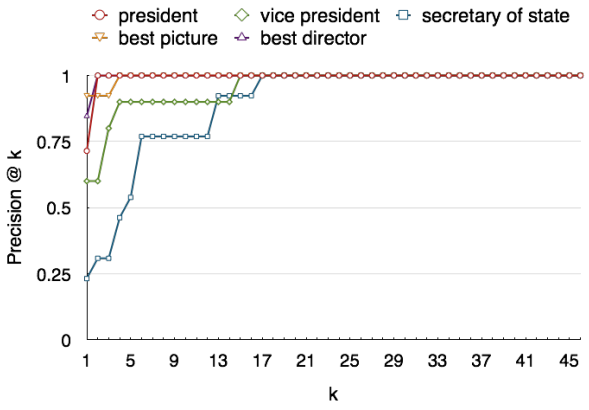


Figure 2: Precision @ k using Contextual Temporal Profiles.

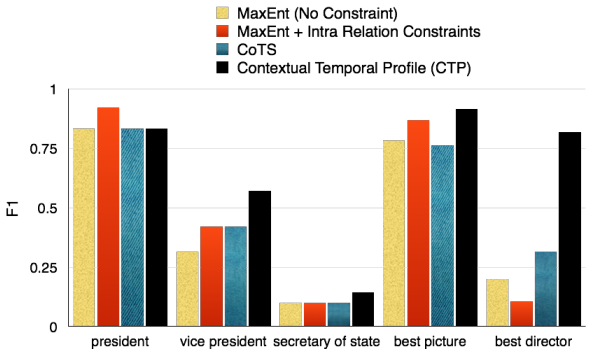


Figure 3: Comparison of F1 scores with CoTS and other baselines.

4.2 CTPs Begin time precision

To compute precision we used cross validation, in particular, leave-one-out cross validation due to the small number of facts per relation.We predict the *begin* time of each fact, the time the fact starts to be valid. True begin times were determined by a human annotator. This human annotated data formed the gold-standard which we used to determine Precision (P), Recall (R), and the F1 measure. All evaluations were performed at the year level, the finest granularity common to the two time-stamped datasets.

For our first experiment, we report the average precision@k, where k=1 to n, where n=47 is the number of years between 1960 to 2008 to select from. As can be seen in Figure 2, precision quickly reaches 1 for most relations. The true begin time is usually found within top k=5 results.

4.3 Comparison to baselines

For our second experiment, we compared to the F1 scores of CoTS and other baselines in (Talukdar 2012b). As can be seen in Figure 3, our CTPs approach gives comparable or better F1 (@k=1) than systems that use only plain temporal profiles, even when these systems are supplemented with many carefully crafted, hand-specified constraints.

We note that the performance on the *US Secretary of State* relation is low in both CoTS (Talukdar 2012b) and in our approach. We found that this was due to few documents mentioning the "secretary of state" in Google Books Ngram dataset. This leads to weak signals for predicting the temporal scope of secretary of state appointments.

We also observe that the uni-grams and bigrams in the train CTPs and change vectors reflect meaningful events and state changes happening to the entities (Table 1). For example, after 'becoming president' and 'taking office', US presidents often see a drop in mentions of their previous (job title state) such as 'senator', 'governor' or 'vice president' as they gain the'president' state.

Relation	CTP State Context	Unigrams and Bigrams in CTP Change Vectors
US President	was elected, took office, became president	vice president (-), by president (+), administration (+), senator (-), governor (-), candidate (-)
Best Picture	nominated for, to win, won the, was nominated	best picture (+), hour minute (-), academy award (+), oscar (+), nominated (+), won (+), star (-), best actress (+), best actor (+), best supporting (+)

Table 1: Example behavior of various contextual units (unigrams and bigrams) automatically learned in the train CTPs and change vector. The *(+)* and *(-)* signs indicate rise and fall in mention frequency, respectively.

4.4 Discussion

Overall, our results show that our method is promising for detecting *begin* time of facts. In its current state, our method performs poorly on inferring *end* times as contexts relevant to a fact often still mentioned with the entity even after the fact ceases to be valid. For example, the entity *Al Gore* is still mentioned a lot with the bi-gram 'vice president' even after he is no longer a vice president. Prior work, CoTS, inferred *end* times by leveraging manually specified constraints, e.g., that there can only be one vice president at a time: the beginning of one signals the end of another (Talukdar 2012b). However such methods do not scale due to the amount of constraints that must be hand-specified. In future, we would like to investigate how to better detect the *end* times of facts.

5 Conclusion

This paper presented a new approach for inferring temporal scopes of facts. Our approach is to reformulate temporal scoping as a state change detection problem. To this end, we introduced Contextual Temporal Profiles (CTPs) which are entity temporal profiles enriched with relevant context. From the CTPs, we learned change vectors that reflect change patterns in context units of CTPs. Our experiments showed that the change patterns are highly relevant for detecting state change, which is an effective way of identifying *begin* times of facts. For future work, we would like to investigate how our method can be improved to dp better at detecting fact *end* times. We also would like to investigate time-stamped corpora of finer-grained granularity such as day. This information can be obtained by subscribing to daily newsfeeds of specific entities.

Acknowledgments

We thank members of the NELL team at CMU for their helpful comments. This research was supported by DARPA under contract number FA8750-13-2-0005 and in part by Fulbright and Google Anita Borg Memorial Scholarship.

References

A. Angel, N. Koudas, N. Sarkas, D. Srivastava: Dense Subgraph Maintenance under Streaming Edge Weight Updates for Real-time Story Identification. In *Proceedings of the VLDB Endowment*, PVLDB 5(10):574–585, 2012.

S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z.G. Ives: DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, pages 722–735, Busan, Korea, 2007.

M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, O. Etzioni: Open Information Extraction from

the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2670–2676, Hyderabad, India, 2007.

S. Bethard and J.H. Martin. Cu-tmp: Temporal relation classification using syntactic and semantic features. In In SemEval-2007, 2007.

K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor: Freebase: a Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages, 1247-1250, Vancouver, BC, Canada, 2008.

A. Carlson, J. Betteridge, R.C. Wang, E.R. Hruschka, T.M. Mitchell: Coupled Semi-supervised Learning for Information Extraction. In *Proceedings of the Third International Conference on Web Search and Web Data Mining (WSDM)*, pages 101–110, New York, NY, USA, 2010.

A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr., T. M. Mitchell: Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)* 2010.

L. Del Corro, R. Gemulla: ClausIE: clause-based open information extraction. In *Proceedings of the 22nd International Conference on World Wide Web (WWW)*, pages 355-366. 2013.

A. Das Sarma, A. Jain, C. Yu: Dynamic Relationship and Event Discovery. In *Proceedings of the Forth International Conference on Web Search and Web Data Mining (WSDM)*, pages 207–216, Hong Kong, China, 2011.

A. Fader, S. Soderland, O. Etzioni: Identifying Relations for Open Information Extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1535–1545, Edinburgh, UK, 2011.

D. Graff, J. Kong, K. Chen, and K. Maeda. English gigaword. Linguistic Data Consortium, Philadelphia, 2003.

C. Havasi, R. Speer, J. Alonso. ConceptNet 3: a Flexible, Multilingual Semantic Network for Common Sense Knowledge. In *Proceedings of the Recent Advances in Natural Language Processing (RANLP)*, Borovets, Bulgaria, 2007.

J. Hoffart, F. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, G. Weikum: YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, pages 229–232, Hyderabad, India. 2011.

X. Ling and D.S. Weld. Temporal information extraction. In Proceedings of AAAI, 2010.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Holberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, Erez Lieberman Aiden: Quantitative Analysis of Culture Using Millions of Digitized Books. *Science*, 331(6014):176182.

N. Nakashole, M. Theobald, G. Weikum: Scalable Knowledge Harvesting with High Precision and High Recall. In *Proceedings of the 4th International Conference on Web Search and Web Data Mining (WSDM)*, pages 227–326, Hong Kong, China, 2011.

N. Nakashole, T. Tylenda, G. Weikum: Fine-grained Semantic Typing of Emerging Entities. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1488-1497, 2013.

N.Nakahsole, T. M. Mitchell: Language-Aware Truth Assessment of Fact Candidates In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1009-1019, 2014.

G. Puscasu. Wvali: Temporal relation identification by syntactico-semantic analysis. In Proceedings of the 4th International Workshop on SemEval, 2007.

J. Pustejovsky, J. Castano, R. Ingria, R. Sauri, R Gaizauskas, A. Setzer, G. Katz, and D. Radev. Timeml: Robust specification of event and temporal expressions in text. In Fifth International Workshop on Computational Semantics, 2003.

P. P. Talukdar, D. T. Wijaya, Tom M. Mitchell: Acquiring temporal constraints between relations. In *Proceeding of the 21st ACM International Conference on Information and Knowledge Management*, pages 992-1001, CIKM 2012.

P. P. Talukdar, D. T. Wijaya, T. Mitchell: Coupled temporal scoping of relational facts. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 2012.

M. Verhagen, I. Mani, R. Sauri, R. Knippen, S.B. Jang, J. Littman, A. Rumshisky, J. Phillips, and J. Pustejovsky. Automating temporal annotation with tarsqi. In Proceedings of the ACL Session on Interactive poster and demonstration sessions, 2005.

M. Verhagen, R. Gaizauskas, F. Schilder, M. Hepple, G. Katz, and J. Pustejovsky. Semeval-2007 task 15: Tempeval temporal relation identi

cation. In Proceedings of the 4th International Workshop on Semantic Evaluations, 2007.

D. T. Wijaya, and R. Yeniterzi: Understanding semantic change of words over centuries. In *Proceedings of the 2011 international workshop on DETecting and Exploiting Cultural diversiTy on the social web*. ACM, 2011.

F. M. Suchanek, G. Kasneci, G. Weikum: Yago: a Core of Semantic Knowledge. In *Proceedings of the 16th International Conference on World Wide Web (WWW)* pages, 697-706, Banff, Alberta, Canada, 2007.

Y. Wang, M. Zhu, L. Qu, M. Spaniol, and G. Weikum: Timely yago: harvesting, querying, and visualizing temporal knowledge from wikipedia. In *Proceedings of the 13th International Conference on Extending-Database Technology*, 2010.

W. Wu, H. Li, H. Wang, K. Zhu: Probase: A Probabilistic Taxonomy for Text Understanding. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 481–492, Scottsdale, AZ, USA, 2012.

Y. Wang, B. Yang, L. Qu, M. Spaniol, and G. Weikum: Harvesting facts from textual web sources by constrained label propagation. In *Proceedings of CIKM*, 2011.

K. Yoshikawa, S. Riedel, M. Asahara, and Y. Matsumoto. Jointly identifying temporal relations with markov logic. In Proceedings of ACL, 2009.

Noisy Or-based model for Relation Extraction using Distant Supervision

Ajay Nagesh[1,2,3] **Gholamreza Haffari** **Ganesh Ramakrishnan**
[1]IITB-Monash Research Academy [2]Faculty of IT, Monash University [3]Dept. of CSE, IIT Bombay
ajaynagesh@cse.iitb.ac.in gholamreza.haffari@monash.edu ganesh@cse.iitb.ac.in

Abstract

Distant supervision, a paradigm of relation extraction where training data is created by aligning facts in a database with a large unannotated corpus, is an attractive approach for training relation extractors. Various models are proposed in recent literature to align the facts in the database to their mentions in the corpus. In this paper, we discuss and critically analyse a popular alignment strategy called the *"at least one"* heuristic. We provide a simple, yet effective relaxation to this strategy. We formulate the inference procedures in training as integer linear programming (*ILP*) problems and implement the relaxation to the *"at least one "* heuristic *via* a soft constraint in this formulation. Empirically, we demonstrate that this simple strategy leads to a better performance under certain settings over the existing approaches.

1 Introduction

Although supervised approaches to relation extraction (GuoDong et al., 2005; Surdeanu and Ciaramita, 2007) achieve very high accuracies, they do not scale as they are data intensive and the cost of creating annotated data is quite high. To alleviate this problem, Mintz et al. (2009) proposed relation extraction in the paradigm of *distant supervision*. In this approach, given a database of facts (e.g. Freebase[1]) and an unannotated document collection, the goal is to heuristically align the facts in the database to the sentences in the corpus which contain the entities mentioned in the fact. This is done to create weakly labeled training data to train a classifier for relation extraction. The underlying assumption is that all mentions of an entity pair[2] (i.e. sentences containing the entity pair) in the corpus express the same relation as stated in the database.

The above assumption is a weak one and is often violated in natural language text. For instance, the entity pair, (`Barack Obama`, `United States`) participate in more than one relation: `citizenOf`, `presidentOf`, `bornIn` and every mention expresses either one of these fixed set of relations or none of them.

Consequently, a number of models have been proposed in literature to provide better heuristics for the mapping between the entity pair in the database and its mentions in the sentences of the corpus. Riedel et al. (2010) tightens the assumption of distant supervision in the following manner: "Given a pair of entities and their mentions in sentences from a corpus, *at least one* of the mentions express the relation given in the database". In other words, it models the problem as that of multi-instance (mentions) single-label (relation) learning. Following this, Hoffmann et al. (2011) and Surdeanu et al. (2012) propose models that consider the mapping as that of multi-instance multi-label learning. The instances are the mentions of the entity pair in the sentences of the corpus and the entity pair can participate in more than one relation.

Although, these models work very well in practice, they have a number of shortcomings. One of them is the possibility that during the alignment, a fact in the database might not have an instantiation in the corpus. For instance, if our corpus only contains documents from the years 2000 to 2005, the fact `presidentOf`(`Barack Obama`, `United States`) will not be present in the corpus. In such cases, the distant supervision assumption fails to provide a mapping for the fact in the corpus.

In this paper, we address this situation with a

[1]`www.freebase.com`

[2]In this paper we restrict ourselves to binary relations

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1937–1941,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

noisy-or model (Srinivas, 2013) in training the relation extractor by relaxing the *"at least one"* assumption discussed above. Our contributions in this paper are as follows: (i) We formulate the inference procedures in the training algorithm as integer linear programming (*ILP*) problems, (ii) We introduce a soft-constraint in the ILP objective to model noisy-or in training, and (iii) Empirically, our algorithm performs better than Hoffmann et al. (2011) procedure under certain settings on two benchmark datasets.

Our paper is organized as follows. In Section 2, we discuss our methodology. We review the approach of Hoffmann et al. (2011) and explain our modifications to it. In Section 3, we discuss related work. In Section 4, we discuss the experimental setup and our preliminary results. We conclude in Section 4.

2 Methodology

Our work extends the work of Hoffmann et al. (2011). So, we recapitulate Hoffmann's model in the following subsection. Following which our additions to this model is explained in detail.

Hoffmann's model

Hoffmann et al. (2011) present a multi-instance multi-label model for relation extraction through distant supervision. In this model, a pair of entities have multiple mentions (sentence containing the entity pair) in the corpus. An entity pair can have one or more relation labels (obtained from the database).

Objective function

Consider an entity pair (e_1, e_2) denoted by the index i. The set of sentences containing the entity pair is denoted $\mathbf{x_i}$ and the set of relation labels for the entity pair from the database is denoted by $\mathbf{y_i}$. The mention-level labels are denoted by the latent variable $\mathbf{z}$ (there is one variable z_j for each sentence j).

To learn the parameters θ, the training objective to maximize is the likelihood of the facts observed in the database conditioned on the sentences in the text corpus.

$$\theta^* = \arg\max_\theta \prod_i Pr(\mathbf{y_i}|\mathbf{x_i};\theta)$$
$$= \arg\max_\theta \prod_i \sum_z Pr(\mathbf{y_i}, \mathbf{z}|\mathbf{x_i};\theta)$$

The expression $Pr(\mathbf{y_i}, \mathbf{z}|\mathbf{x_i})$ for a given entity pair is defined by two types of factors in the factor graph. They are *extract factors* for each mention and *mention factors* between a relation label and all the mentions.

The *extract factors* capture the local signal for each mention and consists of a bunch of lexical and syntactic features like POS tags, dependency path between the entities and so on (Mintz et al., 2009).

The *mention factors* capture the dependency between relation label and its mentions. Here, the *at least one* assumption that was discussed in Section 1 is modeled. It is implemented as a simple deterministic OR operator as given below:

$$f_{mention}(y_r, \mathbf{z}) = \begin{cases} 1 & \text{if } y_r \text{ is true} \wedge \exists i : z_i = r \\ 0 & \text{otherwise} \end{cases}$$

Training algorithm

The learning algorithm is a perceptron-style parameter update scheme with 2 modifications: i) online learning ii) Viterbi approximation. The inference is shown to reduce to the well-known weighted edge-cover problem which can be solved exactly, although Hoffmann et al. (2011) provide an approximate solution.

Algorithm 1: Hoffmann et al. (2011) : Training

Input : *i*) Σ: set of sentences, *ii*) E: set of entities mentioned in the sentences, *iii*) R: set of relation labels, *iv*) Δ: database of facts
Output: Extraction model : Θ
begin
  for $t \leftarrow 1$ **to** T ; /* training iterations */
  do
    for $i \leftarrow 1$ **to** N ; /* No. of entity pairs */
    do
      $\widehat{\mathbf{y}}, \widehat{\mathbf{z}}, = \arg\max_{\mathbf{y},\mathbf{z}} Pr(\mathbf{y}, \mathbf{z}|\mathbf{x_i};\Theta)$
      if $\widehat{\mathbf{y}}! = \mathbf{y_i}$ **then**
        $\mathbf{z}^* = \arg\max_{\mathbf{z}} Pr(\mathbf{z}|\mathbf{y_i}, \mathbf{x_i};\Theta)$
        $\Theta^{new} = \Theta^{old} + \Phi(\mathbf{x_i}, \mathbf{z}^*) - \Phi(\mathbf{x_i}, \widehat{\mathbf{z}})$
end

Our additions to Hoffmann's model

In the training algorithm described above, there are two MAP inference procedures. Our contributions in this space is two-fold. Firstly, we

have formulated these as ILP problems. As a result of this, the approximate inference therein is replaced by an exact inference procedure. Secondly, we replace the *deterministic-or* by a *noisy-or* which provides a soft-constraint instead of the hard-constraint of Hoffmann. (*"at least one"* assumption)

ILP formulations

Some notations:

- z_{ji} : The mention variable z_j (or jth sentence) taking the relation value i
- s_{ji} : Score for z_j taking the value of i. Scores are computed from the *extract* factors
- y_i : relation label being i
- m : number of mentions (sentences) for the given entity pair
- R: total number of relation labels (excluding the *nil* label)

Deterministic OR

The following is the ILP formulation for the exact inference $\arg\max Pr(\mathbf{y}, \mathbf{z}|\mathbf{x_i})$ in the model based on the *deterministic-or*:

$$\max_{Z,Y} \left\{ \sum_{j=1}^{m} \sum_{i \in \{R,nil\}} \left[z_{ji} s_{ji} \right] \right\}$$

$$\textbf{s.t} \quad 1. \quad \sum_{i \in \{R,nil\}} z_{ji} = 1 \quad \forall j$$

$$2. \quad z_{ji} \leq y_i \quad \forall j, \forall i$$

$$3. \quad y_i \leq \sum_{j=1}^{m} z_{ji} \quad \forall i$$

$$\text{where} \quad z_{ji} \in \{0,1\}, \quad y_i \in \{0,1\}$$

The first constraint restricts a mention to have only one label. The second and third constraints impose the *at least one* assumption. This is the same formulation as Hoffmann but expressed as an ILP problem. However, posing the inference as an ILP allows us to easily add more constraints to it.

Noisy OR

As a case-study, we add the *noisy-or* soft-constraint in the above objective function. The idea is to model the situation where a fact is present in the database but it is not instantiated in the text. This is a common scenario, as the facts populated in the database and the text of the corpus can come from different domains and there might not be a very good match.

$$\max_{Z,Y,\epsilon} \left\{ \left(\sum_{j=1}^{m} \sum_{i \in \{R,nil\}} \left[z_{ji} s_{ji} \right] \right) - \left(\sum_{i \in R} \epsilon_i \right) \right\}$$

$$\textbf{s.t} \quad 1. \quad \sum_{i \in \{R,nil\}} z_{ji} = 1 \quad \forall j$$

$$2. \quad z_{ji} \leq y_i \quad \forall j, \forall i$$

$$3. \quad y_i \leq \sum_{j=1}^{m} z_{ji} + \epsilon_i \quad \forall i$$

$$\text{where} \quad z_{ji} \in \{0,1\}, \quad y_i \in \{0,1\}, \quad \epsilon_i \in \{0,1\}$$

In the above formulation, the objective function is augmented with a soft penalty. Also the third constraint is modified with this penalty term. We call this new term ϵ_i and it is a binary variable to model noise. Through this term we encourage *at least one* type of configuration but will not disallow a configuration that does not conform to this. Essentially, the consequence of this is to allow the case where a fact is present in the database but is not instantiated in the text.

3 Related Work

Relation Extraction in the paradigm of distant supervision was introduced by Craven and Kumlien (1999). They used a biological database as the source of distant supervision to discover relations between biological entities. The progression of models for information extraction using distant supervision was presented in Section 1.

Surdeanu et al. (2012) discuss a noisy-or method for combining the scores of various sentence level models to rank a relation during evaluation. In our approach, we introduce the noisy-or mechanism in the training phase of the algorithm.

Our work is inspired from previous works like Roth and tau Yih (2004). The use of ILP for this problem facilitates easy incorporation of different constraints and to the best of our knowledge, has not been investigated by the community.

4 Experiments

The experimental runs were carried out using the publicly available Stanford's distantly supervised slot-filling system[3] (Surdeanu et al., 2011) and Hoffmann et al. (2011) code-base[4].

[3] http://nlp.stanford.edu/software/mimlre.shtml

[4] http://www.cs.washington.edu/ai/raphaelh/mr/

Datasets and Evaluation

We report results on two standard datasets used as benchmarks by the community namely KBP and Riedel datasets. A complete description of these datasets is provided in Surdeanu et al. (2012).

The evaluation setup and module is the same as that described in Surdeanu et al. (2012). We also use the same set of features used by the various systems in the package to ensure that the approaches are comparable. As in previous work, we report precision/recall (P/R) graphs to evaluate the various techniques.

We used the publicly available *lp_solve* package[5] to solve our inference problems.

Performance of ILP

Use of ILP raises concerns about performance as it is NP-hard. In our problem we solve a separate ILP for every entity pair. The number of variables is limited by the number of mentions for the given entity pair. Empirically, on the KBP dataset (larger of the two datasets), Hoffmann takes around 1hr to run. Our ILP formulation takes around 8.5hrs. However, MIMLRE algorithm (EM-based) takes around 23hrs to converge.

Results

We would primarily like to highlight two settings on which we report the P/R curves and contrast it with Hoffmann et al. (2011). Firstly, we replace the approximate inference in that work with our ILP-based exact inference; we call this setting the *hoffmann-ilp*. Secondly, we replace the deterministic-or in the model with a noisy-or, and call this setting the *noisy-or*. We further compare our approach with Surdeanu et al. (2012). The P/R curves for the various techniques on the two datasets are shown in Figures 1 and 2.

We further report the highest F1 point in the P/R curve for both the datasets in Tables 1 and 2.

Table 1 : Highest F1 point in P/R curve : KBP Dataset

	Precision	Recall	F1
Hoffmann	0.306451619	0.197916672	0.2405063349
MIMLRE	0.28061223	0.286458343	0.2835051518
Noisy-OR	0.297002733	0.189236104	0.2311770916
Hoffmann-ilp	0.293010741	0.189236104	0.2299577976

Discussion

We would like to discuss the results in the above two scenarios.

[5] `http://lpsolve.sourceforge.net/5.5/`

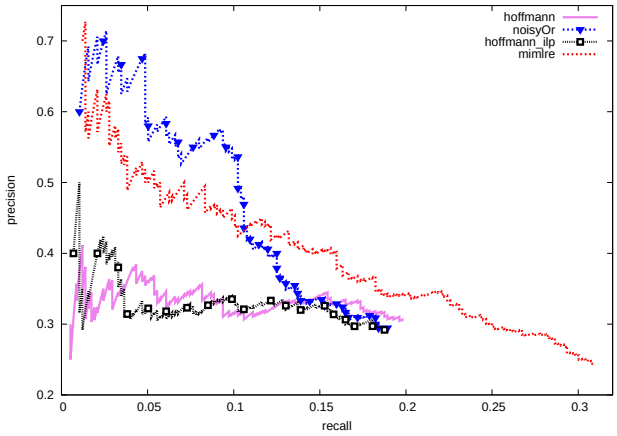


Figure 1: Results : KBP dataset

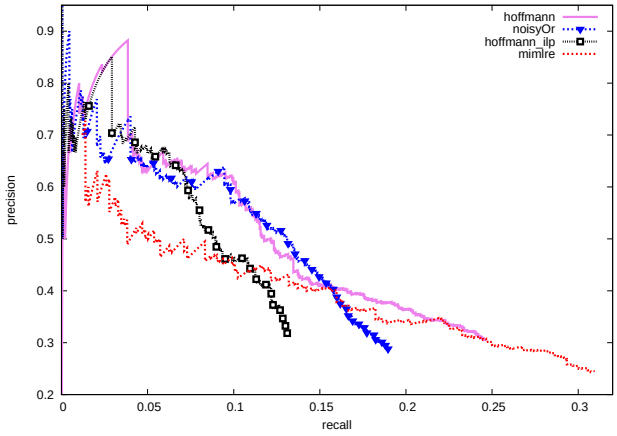


Figure 2: Results : Riedel dataset

1. Performance of *hoffmann-ilp*

   On the KBP dataset, we observe that *hoffmann-ilp* has higher precision in the range of 0.05 to 0.1 at lower recall (0 to 0.04). In other parts of the curve it is very close to the baseline (although hoffmann's algorithm is slightly better). In Table 1, we notice that recall of *hoffmann-ilp* is lower in comparison with hoffmann's algorithm.

   On the Riedel dataset, we observe that *hoffmann-ilp* has better precision (0.15 to 0.2) than MIMLRE within recall of 0.1. At recall > 0.1, precision drops drastically. This is because, *hoffmann-ilp* predicts significantly more nil labels. However, nil labels are not part of the label-set in the P/R curves reported in the community. In Table 2, we see that *hoffmann-ilp* has higher precision (0.04) compared to Hoffmann's algorithm.

2. Performance of *noisy-or*

Table 2 : Highest F1 point in P/R curve : Riedel Dataset			
	Precision	**Recall**	**F1**
Hoffmann	0.32054795	0.24049332	0.27480916
MIMLRE	0.28061223	0.28645834	0.28350515
Noisy-OR	0.317	0.18139774	0.23075178
Hoffmann-ilp	0.36701337	0.12692702	0.18862161

In Figure 1 we see that there is a big jump in precision (around 0.4) of *noisy-or* compared to Hoffmann's model in most parts of the curve on the KBP dataset. However, in Figure 2 (Riedel dataset), we do not see such a trend. Although, we do perform better than MIMLRE (Surdeanu et al., 2012) (precision > 0.15 for recall < 0.15).

On both datasets, *noisy-or* has higher precision than MIMLRE, as seen from Tables 1 and 2. However, the recall reduces. More investigation in this direction is part of future work.

5 Conclusion

In this paper we described an important addition to Hoffmann's model by the use of the *noisy-or* soft constraint to further relax the *at least one* assumption. Since we posed the inference procedures in Hoffmann using ILP, we could easily add this constraint during the training and inference.

Empirically, we showed that the resulting P/R curves have a significant performance boost over Hoffmann's algorithm as a result of this newly added constraint. Although our system has a lower recall when compared to MIMLRE (Surdeanu et al., 2012), it performs competitively w.r.t the precision at low recall.

As part of immediate future work, we would like to improve the system recall. Our ILP formulation provides a good framework to add new type of constraints to the problem. In the future, we would like to experiment with other constraints like modeling the selectional preferences of entity types.

References

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 77–86. AAAI Press.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 427–434, Stroudsburg, PA, USA. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 541–550, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III*, ECML PKDD'10, pages 148–163, Berlin, Heidelberg. Springer-Verlag.

Dan Roth and Wen tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *In Proceedings of CoNLL-2004*, pages 1–8.

Sampath Srinivas. 2013. A generalization of the noisy-or model. *CoRR*, abs/1303.1479.

Mihai Surdeanu and Massimiliano Ciaramita. 2007. Robust information extraction with perceptrons. In *Proceedings of the NIST 2007 Automatic Content Extraction Workshop (ACE07)*, March.

Mihai Surdeanu, Sonal Gupta, John Bauer, David McClosky, Angel X. Chang, Valentin I. Spitkovsky, and Christopher D. Manning. 2011. Stanford's distantly-supervised slot-filling system. In *Proceedings of the Fourth Text Analysis Conference (TAC 2011)*, Gaithersburg, Maryland, USA, November.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 455–465, Stroudsburg, PA, USA. Association for Computational Linguistics.

Search-Aware Tuning for Machine Translation

Lemao Liu
Queens College
City University of New York
lemaoliu@gmail.com

Liang Huang
Queens College and Graduate Center
City University of New York
liang.huang.sh@gmail.com

Abstract

Parameter tuning is an important problem in statistical machine translation, but surprisingly, most existing methods such as MERT, MIRA and PRO are *agnostic* about search, while search errors could severely degrade translation quality. We propose a search-aware framework to promote *promising* partial translations, preventing them from being pruned. To do so we develop two metrics to evaluate partial derivations. Our technique can be applied to all of the three above-mentioned tuning methods, and extensive experiments on Chinese-to-English and English-to-Chinese translation show up to +2.6 BLEU gains over search-agnostic baselines.

1 Introduction

Parameter tuning has been a key problem for machine translation since the statistical revolution. However, most existing tuning algorithms treat the decoder as a black box (Och, 2003; Hopkins and May, 2011; Chiang, 2012), ignoring the fact that many potentially promising partial translations are pruned by the decoder due to the prohibitively large search space. For example, the popular beam-search decoding algorithm for phrase-based MT (Koehn, 2004) only explores $O(nb)$ items for a sentence of n words (with a beam width of b), while the full search space is $O(2^n n^2)$ or worse (Knight, 1999).

As one of the very few exceptions to the "search-agnostic" majority, Yu et al. (2013) and Zhao et al. (2014) propose a variant of the perceptron algorithm that learns to keep the reference translations in the beam or chart. However, there are several obstacles that prevent their method from becoming popular: First of all, they rely on "forced decoding" to track gold derivations that lead to the reference translation, but in practice only a small portion of (mostly very short) sentence pairs have at least one such derivation. Secondly, they learn the model on the training set, and while this does enable a sparse feature set, it is orders of magnitude slower compared to MERT and PRO.

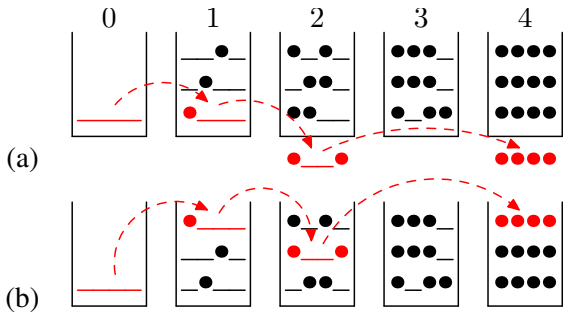


Figure 1: (a) Some potentially promising partial translations (in red) fall out of the beam (bin 2); (b) We identify such partial translations and assign them higher model scores so that they are more likely to survive the search.

We instead propose a very simple framework, **search-aware tuning**, which does not depend on forced decoding, and thus can be trained on all sentence pairs of any dataset. The key idea is that, besides caring about the rankings of the complete translations, we also promote potentially promising partial translations so that they are more likely to survive throughout the search, see **Figure 1** for illustration. We make the following contributions:

- Our idea of search-aware tuning can be applied (as a patch) to all of the three most popular tuning methods (MERT, PRO, and MIRA) by defining a modified objective function (Section 4).
- To measure the "promise" or "potential" of a partial translation, we define a new concept "**potential BLEU**" inspired by future cost in MT decoding (Koehn, 2004) and heuristics in A* search (Hart et al., 1968) (Section 3.2). This work is the first study of evaluating metrics for partial translations.
- Our method obtains substantial and consistent

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1942–1952,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

improvements on both the large-scale NIST Chinese-to-English and English-to-Chinese translation tasks on top of MERT, MIRA, and PRO baselines. This is the first time that consistent improvements can be achieved with a new learning algorithm under dense feature settings (Section 5).

For simplicity reasons, in this paper we use phrase-based translation, but our work has the potential to be applied to other translation paradigms.

2 Review: Beam Search for PBMT Decoding

We review beam search for phrase-based decoding in our notations which will facilitate the discussion of search-aware tuning in Section 4. Following Yu et al. (2013), let $\langle x, y \rangle$ be a Chinese-English sentence pair in the tuning set D, and

$$d = r_1 \circ r_2 \circ \ldots \circ r_{|d|}$$

be a (partial) derivation, where each $r_i = \langle c(r_i), e(r_i) \rangle$ is a rule, i.e., a phrase-pair. Let $|c(r)|$ be the number of Chinese words in rule r, and $e(d) \stackrel{\Delta}{=} e(r_1) \circ e(r_2) \ldots \circ e(r_{|d|})$ be the English prefix (i.e., partial translation) generated so far.

In beam search, each bin $B_i(x)$ contains the best k derivations covering exactly i Chinese words, based on items in previous bins (see Figures 1 and 2):

$$B_0(x) = \{\epsilon\}$$
$$B_i(x) = \mathbf{top}^k_{\mathbf{w}_0}\big(\bigcup_{j=1..i} \{d \circ r \mid d \in B_{i-j}(x), |c(r)| = j\}\big)$$

where r is a rule covering j Chinese words, and $\mathbf{top}^k_{\mathbf{w}_0}(\cdot)$ returns the top k derivations according to the current model $\mathbf{w}_0$. As a special case, note that $\mathbf{top}^1_{\mathbf{w}_0}(S) = \mathbf{argmax}_{d \in S}\, \mathbf{w}_0 \cdot \mathbf{\Phi}(d)$, so $\mathbf{top}^1_{\mathbf{w}_0}(B_{|x|}(x))$ is the final 1-best result.[1] See Figure 2 for an illustration.

3 Challenge: Evaluating Partial Derivations

As mentioned in Section 1, the current mainstream tuning methods such as MERT, MIRA, and PRO are all **search-agnostic**: they only care about the *complete* translations from the last bin, $B_{|x|}(x)$, ignoring all *partial* ones, i.e., $B_i(x)$ for all $i < |x|$. As a result, many potentially promising partial derivations never reach the final bin (See Figure 1).

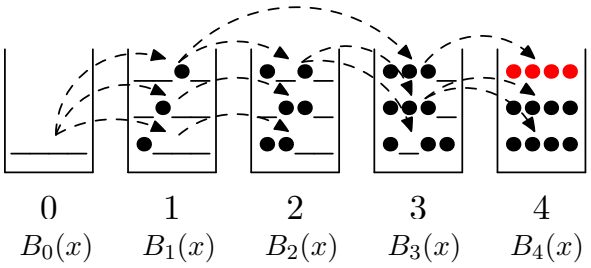


Figure 2: Beam search for phrase-based decoding. The item in red is $\mathbf{top}^1_{\mathbf{w}_0}(B_4(x))$, i.e., the 1-best result. Traditional tuning only uses the final bin $B_4(x)$ while search-aware tuning considers all bins $B_i(x)$ ($i = 1..4$).

To address this problem, our new "search-aware tuning" aims to promote not only the accurate translations in the final bin, but more importantly those potentially promising *partial derivations* in non-final bins. The key challenge, however, is how to evaluate the "promise" or "potential" of a partial derivation. In this Section, we develop two such measures, a simple "partial BLEU" (Section 3.1) and a more principled "potential BLEU" (Section 3.2). In Section 4, we will then adapt traditional tuning methods to their search-aware versions using these partial evaluation metrics.

3.1 Solution 1: Simple and Naive Partial BLEU

Inspired by a trick in (Li and Khudanpur, 2009) and (Chiang, 2012) for oracle or hope extraction, we use a very simple metric to evaluate partial translations for tuning. For a given derivation d, the basic idea is to evaluate the (short) partial translation $e(d)$ against the (full) reference y, but using a "prorated" reference length proportional to $c(d)$ which is the number of Chinese words covered so far in d:

$$|y| \cdot |c(d)|/|x|$$

For example, if d has covered 2 words on a 8-word Chinese sentence with a 12-word English reference, then the "effective reference length" is $12 \times 2/8 = 3$. We call this method "partial BLEU" since it does not complete the translation, and denote it by

$$\bar{\delta}^{|x|}_y(d) = -\delta\big(y, e(d); \text{reflen} = |y| \cdot |c(d)|/|x|\big). \quad (1)$$

[1] Actually $B_{|x|}(x)$ is an approximation to the k-best list since some derivations are merged by dynamic programming; to recover those we can use Alg. 3 of Huang and Chiang (2005).

$$\delta(y, y') = -\text{Bleu}^{+1}(y, y') \quad \text{string distance metric}$$
$$\delta_y(d) = \delta(y, e(d)) \quad \text{full derivations eval}$$
$$\delta_y^x(d) = \begin{cases} \bar{\delta}_y^{|x|}(d) & \text{partial bleu (Sec. 3.1)} \\ \delta(y, \bar{e}_x(d)) & \text{potential bleu (Sec. 3.2)} \end{cases}$$

Table 1: Notations for evaluating full and partial derivations. Functions $\bar{\delta}_y^{|x|}(\cdot)$ and $\bar{e}_x(\cdot)$ are defined by Equations 1 and 3, respectively.

where reflen is the effective length of reference translations, see (Papineni et al., 2002) for details.

3.1.1 Problem with Partial BLEU

Simple as it is, this method does not work well in practice because comparison of partial derivations might be unfair for different derivations covering different set of Chinese words, as it will naturally favor those covering "easier" portions of the input sentence (which we do observe empirically). For instance, consider the following Chinese-to-English example which involves a reordering of the Chinese PP:

(2) *wǒ cóng Shànghǎi fēi dào Běijīng*
I from Shanghai fly to Beijing
"I flew from Shanghai to Beijing"

Partial BLEU will prefer subtranslation "I from" to "I fly" in bin 2 (covering 2 Chinese words) because the former has 2 unigram mathces while the latter only 1, even though the latter is almost identical to the reference and will eventually lead to a complete translation with substantially higher Bleu^{+1} score (matching a 4-gram "from Shanghai to Beijing"). Similarly, it will prefer "I from Shanghai" to "I fly from" in bin 3, without knowing that the former will eventually pay the price of word-order difference. This example suggests that we need a more "global" or less greedy metric (see below).

3.2 Solution 2: Potential BLEU via Extension

Inspired by future cost computation in MT decoding (Koehn, 2004), we define a very simple future string by simply concatenating the best model-score translation (with no reorderings) in each uncovered span. Let $\text{best}_{\mathbf{w}}(x_{[i:j]})$ denote the best monotonic derivation for span $[i:j]$, then

$$future(d, x) = \circ_{[i:j] \in uncov(d,x)} \; e(\text{best}_{\mathbf{w}}(x_{[i:j]}))$$

where $\circ$ is the concatenation operator and $uncov(d, x)$ returns an ordered list of uncovered spans of x. See Figure 3 for an example. This future string resembles (inadmissible) heuristic function (Hart et al., 1968). Now the "extended translation" is simply a concatenation of the existing partial translation $e(d)$ and the future string $future(d, x)$:

$$\bar{e}_x(d) = e(d) \circ future(d, x). \quad (3)$$

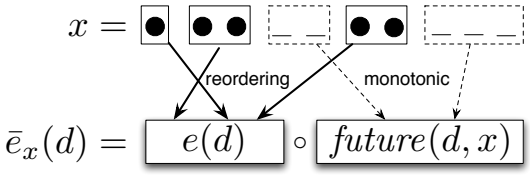


Figure 3: Example of the extension function $\bar{e}_x(\cdot)$ (and future string) on an incomplete derivation d.

Instead of calculating $\text{best}_{\mathbf{w}}(x_{[i:j]})$ on-the-fly for each derivation d, we can precompute it for each span $[i : j]$ during future-cost computation, since the score of $\text{best}_{\mathbf{w}}(x_{[i:j]})$ is context-free (Koehn, 2004). Algorithm 1 shows the pseudo-code of computing $\text{best}_{\mathbf{w}}(x_{[i:j]})$. In practice, since future-cost precomputation already solves the best (monotonic) model-score for each span, is the only extra work for potential BLEU is to record (for each span) the subtranslation that achieves that best score. Therefore, the extra time for potential BLEU is negligible (the time complexity is $O(n^2)$, but just as in future cost, the constant is much smaller than real decoding). The implementation should require minimal hacking on a phrase-based decoder (such as Moses).

To summarize the notation, we use $\delta_y^x(d)$ to denote a generic evaluation function for partial derivation d, which could be instantiated in two ways, partial bleu ($\bar{\delta}_y^{|x|}(d)$) or potential bleu ($\delta(y, \bar{e}_x(d))$). See Table 1 for details. The next Section will only use the generic notation $\delta_y^x(d)$.

Finally, it is important to note that although both partial and potential metrics are not BLEU-specific, the latter is much easier to adapt to other metrics such as TER since it does not change the original Bleu^{+1} definition. By contrast, it is not clear to us at all how to generalize partial BLEU to partial TER.

4 Search-Aware MERT, MIRA, and PRO

Parameter tuning aims to optimize the weight vector $\mathbf{w}$ so that the rankings based on model score defined by $\mathbf{w}$ is positively correlated with those based

Algorithm 1 Computation of best Translations for Potential BLEU.

Input: Source sentence x, a rule set $\mathfrak{R}$ for x, and $\mathbf{w}$.
Output: Best translations $e(\text{best}_{\mathbf{w}}(x[i:j]))$ for all spans $[i:j]$.

for l **in** $(0..|x|)$ **do**
  for i **in** $(0..|x|-l)$ **do**
    $j = i + l + 1$
    best_score $= -\infty$
    if $\mathfrak{R}[i:j] \neq \emptyset$ **then** ▷ $\mathfrak{R}[i:j]$ is a subset of rules $\mathfrak{R}$ for span $[i:j]$.
      $\text{best}_{\mathbf{w}}(x[i:j]) = \mathbf{argmax}_{r \in \mathfrak{R}[i:j]}\, \mathbf{w} \cdot \mathbf{\Phi}(\{r\})$ ▷ $\{r\}$ is a derivation consisting of one rule r.
      best_score $= \mathbf{w} \cdot \mathbf{\Phi}(\text{best}_{\mathbf{w}}(x[i:j]))$
    for k **in** $(i+1\ ..\ i+p)$ **do** ▷ p is the phrase length limit
      if best_score $< \mathbf{w} \cdot \mathbf{\Phi}\Big(\text{best}_{\mathbf{w}}(x[i:k]) \circ \text{best}_{\mathbf{w}}(x[k:j])\Big)$ **then**
        $\text{best}_{\mathbf{w}}(x[i:j]) = \text{best}_{\mathbf{w}}(x[i:k]) \circ \text{best}_{\mathbf{w}}(x[k:j])$
        best_score $= \mathbf{w} \cdot \mathbf{\Phi}(\text{best}_{\mathbf{w}}(x[i:j]))$

on some translation metric (such as BLEU (Papineni et al., 2002)). In other words, for a training sentence pair $\langle x, y \rangle$, if a pair of its translations $y_1 = e(d_1)$ and $y_2 = e(d_2)$ satisfies $\text{BLEU}(y, y_1) > \text{BLEU}(y, y_2)$, then we expect $\mathbf{w} \cdot \mathbf{\Phi}(d_1) > \mathbf{w} \cdot \mathbf{\Phi}(d_2)$ to hold after tuning.

4.1 From MERT to Search-Aware MERT

Suppose D is a tuning set of $\langle x, y \rangle$ pairs. Traditional MERT learns the weight by iteratively reranking the complete translations towards those with higher BLEU in the final bin $B_{|x|}(x)$ for each x in D. Formally, it tries to minimize the document-level error of 1-best translations:

$$\ell_{\text{MERT}}(D, \mathbf{w}) = \bigoplus_{\langle x,y \rangle \in D} \delta_y\Big(\mathbf{top}^1_{\mathbf{w}}(B_{|x|}(x))\Big), \tag{4}$$

where $\mathbf{top}^1_{\mathbf{w}}(S)$ is the best derivation in S under model $\mathbf{w}$, and $\delta.(\cdot)$ is the full derivation metric as defined in Table 1; in this paper we use $\delta_y(y') = -\text{BLEU}(y, y')$. Here we follow Och (2003) and Lopez (2008) to simplify the notations, where the $\oplus$ operator (similar to $\sum$) is an over-simplification for BLEU which, as a document-level metric, is actually not factorizable across sentences.

Besides reranking the complete translations as traditional MERT, our search-aware MERT (SA-MERT) also reranks the partial translations such that potential translations may survive in the middle bins during search. Formally, its objective function is defined as follows:

$$\ell_{\text{SA-MERT}}(D, \mathbf{w}) = \bigoplus_{\langle x,y \rangle \in D} \boxed{\bigoplus_{i=1..|x|}} \boxed{\delta^x_y}\Big(\mathbf{top}^1_{\mathbf{w}}(\boxed{B_i}(x))\Big) \tag{5}$$

where $\mathbf{top}^1_{\mathbf{w}}(\cdot)$ is defined in Eq. (4), and $\delta^x_y(d)$, defined in Table 1, is the generic metric for evaluating a partial derivation d which has two implementations (partial bleu or potential bleu). In order words we can obtain two implementations of search-aware MERT methods, SA-MERT^{par} and SA-MERT^{pot}.

Notice that the traditional MERT is a special case of SA-MERT where i is fixed to $|x|$.

4.2 From MIRA to Search-Aware MIRA

MIRA is another popular tuning method for SMT. It firstly introduced in (Watanabe et al., 2007), and then was improved in (Chiang et al., 2008; Chiang, 2012; Cherry and Foster, 2012). Its main idea is to optimize a weight such that the model score difference of a pair of derivations is greater than their loss difference.

In this paper, we follow the objective function in (Chiang, 2012; Cherry and Foster, 2012), where only the violation between hope and fear derivations is concerned. Formally, we define $d^+(x, y)$ and $d^-(x, y)$ as the hope and fear derivations in the final bin (i.e., complete derivations):

$$d^+(x, y) = \mathop{\mathbf{argmax}}_{d \in B_{|x|}(x)}\ \mathbf{w}_0 \cdot \mathbf{\Phi}(d) - \delta_y(d) \tag{10}$$

$$d^-(x, y) = \mathop{\mathbf{argmax}}_{d \in B_{|x|}(x)}\ \mathbf{w}_0 \cdot \mathbf{\Phi}(d) + \delta_y(d) \tag{11}$$

where $\mathbf{w}_0$ is the current model. The loss function of MIRA is in Figure 4. The update will be between $d^+(x, y)$ and $d^-(x, y)$.

To adapt MIRA to search-aware MIRA (SA-MIRA), we need to extend the definitions of hope

$$\ell_{\text{MIRA}}(D,\mathbf{w}) = \frac{1}{2C}\|\mathbf{w}-\mathbf{w}_0\|^2 + \sum_{\langle x,y\rangle\in D}\left[\Delta\delta_y\big(d^+(x,y),d^-(x,y)\big) - \mathbf{w}\cdot\Delta\boldsymbol{\Phi}\big(d^+(x,y),d^-(x,y)\big)\right]_+ \tag{6}$$

$$\ell_{\text{SA-MIRA}}(D,\mathbf{w}) = \frac{1}{2C}\|\mathbf{w}-\mathbf{w}_0\|^2 + \sum_{\langle x,y\rangle\in D}\sum_{i=1}^{|x|}\left[\Delta\delta_y^x\big(d_i^+(x,y),d_i^-(x,y)\big) - \mathbf{w}\cdot\Delta\boldsymbol{\Phi}\big(d_i^+(x,y),d_i^-(x,y)\big)\right]_+ \tag{7}$$

$$\ell_{\text{PRO}}(D,\mathbf{w}) = \sum_{\langle x,y\rangle\in D}\ \sum_{d_1,d_2\in B_{|x|}(x),\ \Delta\delta_y(d_1,d_2)>0} \log\Big(1+\exp(-\mathbf{w}\cdot\Delta\boldsymbol{\Phi}(d_1,d_2))\Big) \tag{8}$$

$$\ell_{\text{SA-PRO}}(D,\mathbf{w}) = \sum_{\langle x,y\rangle\in D}\sum_{i=1}^{|x|}\ \sum_{d_1,d_2\in B_i(x),\ \Delta\delta_y^x(d_1,d_2)>0} \log\Big(1+\exp(-\mathbf{w}\cdot\Delta\boldsymbol{\Phi}(d_1,d_2))\Big) \tag{9}$$

Figure 4: Loss functions of MIRA, SA-MIRA, PRO, and SA-PRO. The differences between traditional and search-aware versions are highlighted in gray. The hope and fear derivations are defined in Equations 10–13, and we define $\Delta\delta_y(d_1,d_2) = \delta_y(d_1) - \delta_y(d_2)$, and $\Delta\delta_y^x(d_1,d_2) = \delta_y^x(d_1) - \delta_y^x(d_2)$. In addition, $[\theta]_+ = \max\{\theta, 0\}$.

and fear derivations from the final bin to all bins:

$$d_i^+(x,y) = \operatorname*{\mathbf{argmax}}_{d\in B_i(x)}\ \mathbf{w}_0\cdot\boldsymbol{\Phi}(d) - \delta_y(d) \tag{12}$$

$$d_i^-(x,y) = \operatorname*{\mathbf{argmax}}_{d\in B_i(x)}\ \mathbf{w}_0\cdot\boldsymbol{\Phi}(d) + \delta_y(d) \tag{13}$$

The new loss function for SA-MIRA is Eq. 7 in Figure 4. Now instead of one update per sentence, we will perform $|x|$ updates, each based on a pair $d_i^+(x,y)$ and $d_i^-(x,y)$.

4.3 From PRO to Search-Aware PRO

Finally, the PRO algorithm (Hopkins and May, 2011; Green et al., 2013) aims to correlate the ranking under model score and the ranking under BLEU score, among all complete derivations in the final bin. For each preference-pair $d_1, d_2 \in B_{|x|}(x)$ such that d_1 has a higher BLEU score than d_2 (i.e., $\delta_y(d_1) < \delta_y(d_2)$), we add one positive example $\boldsymbol{\Phi}(d_1) - \boldsymbol{\Phi}(d_2)$ and one negative example $\boldsymbol{\Phi}(d_2) - \boldsymbol{\Phi}(d_1)$.

Now to adapt it to search-aware PRO (SA-**PRO**), we will have many more examples to consider: besides the final bin, we will include all preference-pairs in the non-final bins as well. For each bin $B_i(x)$, for each preference-pairs $d_1, d_2 \in B_i(x)$ such that d_1 has a higher partial or potential BLEU score than d_2 (i.e., $\delta_y^x(d_1) < \delta_y^x(d_2)$), we add one positive example $\boldsymbol{\Phi}(d_1) - \boldsymbol{\Phi}(d_2)$ and one negative example $\boldsymbol{\Phi}(d_2) - \boldsymbol{\Phi}(d_1)$. In sum, search-aware PRO has $|x|$ times more examples than traditional PRO. The loss functions of PRO and search-aware PRO are defined in Figure 4.

5 Experiments

We evaluate our new tuning methods on two large scale NIST translation tasks: Chinese-to-English (CH-EN) and English-to-Chinese (EN-CH) tasks.

5.1 System Preparation and Data

We base our experiments on Cubit[2] (Huang and Chiang, 2007), a state-of-art phrase-based system in Python. We set phrase-limit to 7, beam size to 30 and distortion limit 6. We use the 11 dense features from Moses (Koehn et al., 2007), which can lead to good performance and are widely used in almost all SMT systems. The baseline tuning methods MERT (Och, 2003), MIRA (Cherry and Foster, 2012), and PRO (Hopkins and May, 2011) are from the Moses toolkit, which are batch tuning methods based on k-best translations. The search-aware tuning methods are called SA-MERT, SA-MIRA, and SA-PRO, respectively. Their partial BLEU versions are marked with superscript [1] and their potential BLEU versions are marked with superscript [2], as explained in Section 3. All these search-aware tuning methods are implemented on the basis of Moses toolkit. They employ the de-

[2] http://www.cis.upenn.edu/~lhuang3/cubit/

Methods	nist03	nist04	nist05	nist06	nist08	avg
MERT	33.6	35.1	33.4	31.6	27.9	–
SA-MERT^{par}	-0.2	+0.0	+0.1	-0.1	-0.1	–
SA-MERT^{pot}	**+0.8**	**+1.1**	**+0.9**	**+1.7**	**+1.5**	+1.2
MIRA	33.5	35.2	33.5	31.6	27.6	–
SA-MIRA^{par}	+0.3	+0.3	+0.4	+0.4	+0.6	–
SA-MIRA^{pot}	**+1.3**	**+1.6**	**+1.4**	**+2.2**	**+2.6**	+1.8
PRO	33.3	35.1	33.3	31.1	27.5	–
*SA-PRO^{par}	-2.0	-2.7	-2.2	-1.0	-1.7	–
*SA-PRO^{pot}	**+0.8**	**+0.5**	**+1.0**	**+1.6**	**+1.6**	+1.1

Table 2: CH-EN task: BLEU scores on test sets (nist03, nist04, nist05, nist06, and nist08). ^{par}: partial BLEU; ^{pot}: potential BLEU. *: SA-PRO tunes on only 109 short sentences (with less than 10 words) from nist02.

	Final bin	All bins
MERT	35.5	28.2
SA-MERT	-0.1	+3.1

Table 3: Evaluation on nist02 tuning set using two methods: BLEU is used to evaluate 1-best complete translations in the final bin; while potential BLEU is used to evaluate 1-best partial translations in all bins. The search-aware objective cares about (the potential of) all bins, not just the final bin, which can explain this result.

fault settings following Moses toolkit: for MERT and SA-MERT, the stop condition is defined by the weight difference threshold; for MIRA, SA-MIRA, PRO and SA-PRO, their stop condition is defined by max iteration set to 25; for all tuning methods, we use the final weight for testing.

The training data for both CH-EN and EN-CH tasks is the same, and it is collected from the NIST2008 Open Machine Translation Campaign. It consists of about $1.8M$ sentence pairs, including about $40M/48M$ words in Chinese/English sides. For CH-EN task, the tuning set is nist02 (878 sents), and test sets are nist03 (919 sents), nist04 (1788 sents), nist05 (1082 sents), nist06 (616 sents from news portion) and nist08 (691 from news portion). For EN-CH task, the tuning set is ssmt07 (995 sents)[3], and the test set is nist08 (1859 sents). For both tasks, all the tuning and test sets contain 4 references.

We use GIZA++ (Och and Ney, 2003) for word alignment, and SRILM (Stolcke, 2002) for 4-gram language models with the Kneser-Ney smoothing option. The LM for EN-CH is trained on its target side; and that for CH-EN is trained on the Xinhua portion of Gigaword. We use BLEU-4 (Papineni et al., 2002) with "average ref-len" to evaluate the translation performance for all experiments. In particular, the character-based BLEU-4 is employed for EN-CH task. Since all tuning methods involve randomness, all scores reported are average of three runs, as suggested by Clark et al. (2011) for fairer comparisons.

5.2 Main Results on CH-EN Task

Table 2 depicts the main results of our methods on CH-EN translation task. On all five test sets, our methods consistently achieve substantial improvements with two pruning options: SA-MERT^{pot} gains +1.2 BLEU points over MERT on average; and SA-MIRA^{pot} gains +1.8 BLEU points over MIRA on average as well. SA-PRO^{pot}, however, does not work out of the box when we use the entire nist02 as the tuning set, which might be attributed to the "Monster" behavior (Nakov et al., 2013). To alleviate this problem, we only use the 109 short sentences with less than 10 words from nist02 as our new tuning data. To our supurise, this trick works really well (despite using much less data), and also made SA-PRO^{pot} an order of magnitude faster. This further confirms that our search-aware tuning is consistent across all tuning methods and datasets.

As discussed in Section 3, evaluation metrics of partial derivations are crucial for search-aware tuning. Besides the principled "potential BLEU" version of search-aware tuning (i.e. SA-MERT^{pot}, SA-MIRA^{pot}, and SA-PRO^{pot}), we also run the simple "partial BLEU" version of search-aware tuning (i.e. SA-MERT^{par}, SA-MIRA^{par}, and SA-

[3] On EN-CH task, there is only one test set available for us, and thus we use ssmt07 as the tuning set, which is provided at the Third Symposium on Statistical Machine Translation (http://mitlab.hit.edu.cn/ssmt2007.html).

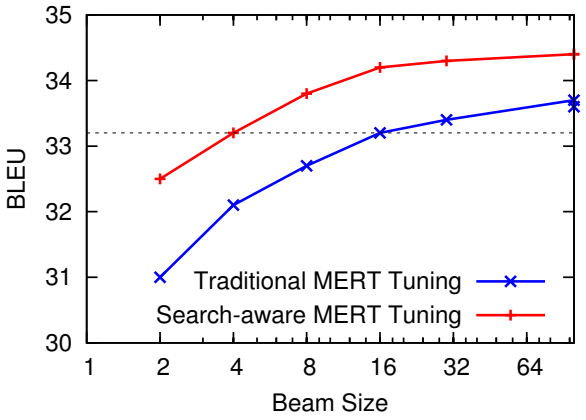


Figure 5: BLEU scores against beam size on nist05. Our search-aware tuning can achieve (almost) the same BLEU scores with much smaller beam size (beam of 4 vs. 16).

	methods	nist02	nist05
1-best	MERT	35.5	33.4
	SA-MERT	-0.1	+0.9
Oracle	MERT	44.3	41.1
	SA-MERT	+0.5	+1.6

Table 4: The k-best oracle BLEU comparison between MERT and SA-MERT.

PRO^{par}). In Table 2, we can see that they may achieve slight improvements over tradition tuning on some datasets, but SA-MERT^{pot}, SA-MIRA^{pot}, and SA-PRO^{pot} using potential BLEU consistently outperform them on all the datasets.

Even though our search-aware tuning gains substantially on all test sets, it does not gain significantly on nist02 tuning set. The main reason is that, search-aware tuning optimizes an objective (i.e. BLEU for all bins) which is different from the objective for evaluation (i.e. BLEU for the final bin), and thus it is not quite fair to evaluate the complete translations for search-aware tuning as the same done for traditional tuning on the tuning set. Actally, if we evaluate the potential BLEU for all partial translations, we find that search-aware tuning gains about 3.0 BLEU on nist02 tuning set, as shown in Table 3.

5.3 Analysis on CH-EN Task

Different beam size. Since our search-aware tuning considers the rankings of partial derivations in the middle bins besides complete ones in the last bin, ideally, if the weight learned by search-aware tuning can exactly evaluate partial derivations, then accurate partial derivations will rank higher according to model score. In this way, even with small beam size, these accurate partial derivations may still survive in the bins. Therefore, it is expected that search-aware tuning can achieve good performance with smaller beam size. To justify our conjecture, we run SA-MERT^{pot} with different beam size (2,4,8,16,30,100), its testing results on nist05 are depicted in Figure 5. our mehtods achieve better trade-off between performance and efficiey. Figure 5 shows that search-aware tuning is consistent with all beam sizes, and as a by-product, search-aware MERT with a beam of 4 can achieve almost identical BLEU scores to MERT with beam of 16.

Diversity	nist02	nist05
MERT	0.216	0.204
SA-MERT	0.227	0.213

Table 5: The diversity comparison based on the k-best list in the final bin on both tuning and nist05 test sets by tuning methods. The higher the metric is, the more diverse the k-best list is.

Oracle BLEU. In addition, we examine the BLEU ponits of oracle for MERT and SA-MERT. We use the weight tuned by MERT and SA-MERT for k-best decoding on nist05 test set, and calculate the oracle over these two k-best lists. The oracle BLEU comparison is shown in Table 4. On nist05 test set, for MERT the oracle BLEU is 41.1; while for SA-MERT its oracle BLEU is 42.7, i.e. with 1.6 BLEU improvements. Although search-aware tuning employs the objective different from the objective of evaluation on nist02 tuning set, it still gains 0.5 BLEU improvements.

Diversity. A k-best list with higher diversity can better represent the entire decoding space, and thus tuning on such a k-best list may lead to better tesing performance (Gimpel et al., 2013). Intuitively, tuning with all bins will encourage the diversity in prefix, infix and suffix of complete translations in the final bin. To testify this, we need a diversity metric.

Indeed, Gimpel et al. (2013) define a diversity metric based on the n-gram matches between two sentences y and y' as follows:

$$d(y, y') = -\sum_{i=1}^{|y|-q} \sum_{j=1}^{|y'|-q} [\![y_{i:i+q} = y'_{j:j+q}]\!]$$

Methods	tuning set				test sets (4-refs)				
	set	# refs	# sents	# words	nist03	nist04	nist05	nist06	nist08
MERT	nist02	4	878	23181	33.6	35.1	33.4	31.6	27.9
SA-MERT^{pot}	nist02	4	878	23181	**34.4**	**36.2**	**34.3**	**33.3**	**29.4**
MAXFORCE	nist02-px	1	434	6227	29.0	30.3	28.7	26.8	24.1
MAXFORCE	train-r-part	1	1225	22684	31.7	33.5	31.5	30.3	26.7
MERT	nist02-r	1	92	1173	31.6	32.7	31.3	29.3	25.9
SA-MERT^{pot}	nist02-r	1	92	1173	33.5	35.0	33.4	31.5	28.0

Table 6: Comparisons with MAXFORCE in terms of BLEU. nist02-px is the non-trivial reachable prefix-data from nist02 via forced decoding; nist02-r is a subset of nist02-px consisting of the fully reachable data; train-r is a subset of fully reachable data from training data that is comparable in size to nist02. All experiments use only dense features.

where $q = n - 1$, and $[\![x]\!]$ equals to 1 if x is true, 0 otherwise. This metric, however, has the following critical problems:

- it is not length-normalized: longer strings will look as if they are more different.
- it suffers from duplicates in n-grams. After normalization, $d(y, y)$ will exceed -1 for any y. In the extreme case, consider y_1 = "the the the the" and y_2 = "the ... the" with 10 the's then will be considered identical after normalization by length.

So we define a balanced metric based on their metric

$$d'(y, y') = 1 - \frac{2 \times d(y, y')}{d(y, y) + d(y', y')}$$

which satisfies the following nice properties:

- $d'(y, y) = 0$ for all y;
- $0 \leq d'(y, y') \leq 1$ for all y, y';
- $d'(y, y') = 1$ if y and y' is completely disjoint.
- it does not suffer from duplicates, and can differentiate y_1 and y_2 defined above.

With this new metric, we evaluate the diversity of k-best lists for both MERT and SA-MERT. As shown in Table 5, on both tuning and test sets the k-best list generated by SA-MERT is more diverse.

5.4 Comparison with Max-Violation Perceptron

Our method considers the rankings of partial derivations, which is simlar to MAXFORCE method (Yu et al., 2013), and thus we re-implement MAXFORCE method. Since the nist02 tuning set contains 4 references and forced decoding is performed for only one reference, we enlarge the nist02 set to a variant set following the transformation in Figure 6, and obtain a variant tuning set denoted as **nist02-px**, which consists of 4-times sentence-pairs. On nist02-px, the non-trivial reachable prefix-data only accounts for 12% sentences and 7% words. Both these sentence-level and the word-level percentages are much lower than those on the training data as shown in Table 3 from (Yu et al., 2013). This is because there are many OOV words on a tuning set. We run the MAXFORCE with dense feature setting on nist02-px and its testing results are shown in Table 6. We can see that on all the test sets, its testing performance is lower than that of SA-MERT^{pot} tuning on nist02 with about 5 BLEU points.

Bùshí yǔ Shālóng jǔxíng huìtán	Bush and Sharon held a meeting Bush held talks with Sharon
qiāngshǒu bèi jǐngfāng jībì	police killed the gunman the gunman was shot dead

⇓

Bùshí yǔ Shālóng jǔxíng huìtán	Bush and Sharon held a meeting
Bùshí yǔ Shālóng jǔxíng huìtán	Bush held talks with Sharon
qiāngshǒu bèi jǐngfāng jībì	police killed the gunman
qiāngshǒu bèi jǐngfāng jībì	the gunman was shot dead

Figure 6: Transformation of a tuning set in forced decoding for MAXFORCE: the original tuning set (on the top) contains 2 source sentences with 2 references for each; while the transformed set (on the bottom) contains 4 source sentences with one reference for each.

For more direct comparisons, we run MERT and SA-MERT^{pot} on a data set similar to nist02-px. We pick up the fully reachable sentences from nist02-px, remove the sentence pairs with the same source side, and get a new tuning set denoted as **nist02-r**. When tuning on nist02-r, we find that MERT is bet-

Methods	tuning-set	nist08
MERT	ssmt07	31.3
MAXFORCE	train-r-part	29.9
SA-MERT^{par}	ssmt07	31.3
SA-MERT^{pot}	ssmt07	**31.7**

Table 7: EN-CH task: BLEU scores on nist08 test set for MERT, SA-MERT, and MAXFORCE on different tuning sets. train-r-part is a part of fully reachable data from training data via forced decoding. All the tuning methods run with dense feature set.

Optimization time	MERT	MIRA	PRO
basline	3	2	2
search-aware	50	7	6

Table 8: Search-aware tuning slows down MERT significantly, and MIRA and PRO moderately. The time (in minutes) is for optimization only (excluding decoding) and measured at the last iteration during the entire tuning (search aware tuning does not increase the number of iterations in our experiments). The decoding time is 20 min. on a single CPU but can be parallelized.

ter than MAXFORCE,[4] and SA-MERT^{pot} are much better than MERT on all the test sets. In addition, we select about $1.2k$ fully reachable sentence pairs from training data, and run the forced decoding on this new tuning data (denoted as **train-r-part**), which is with similar size to nist02. [5] With more tuning data, the performance of max-violation is improved largely, but it is still underperformed by SA-MERT^{pot}.

5.5 Results on EN-CH Translation Task

We also run our search-aware tuning method on EN-CH task. We use SA-MERT as the representative of search-aware tuning methods, and compare its two versions with other tuning methods MERT, MAXFORCE. For MAXFORCE, we first run forced decoding on the training data and then select about $1.2k$ fully reachable sentence pairs as its tuning set (denoted as **train-r-part**). For MERT, SA-MERT^{pot}, and SA-MERT^{par}, their tuning set is ssmt07. Table 7 shows that SA-MERT^{pot} is much better than MAXFORCE, i.e. it achieves 0.4 BLEU improvements over MERT. Finally, comparison between SA-MERT^{pot} and SA-MERT^{par} shows that the potential BLEU is better for evaluation of partial derivations.

5.6 Discussions on Tuning Efficiency

As shown in Figure 2, search-aware tuning considers all partial translations in the middle bins beside all complete translations in the last bin, and thus its total number of training examples is much greater than that of the traditional tuning. In details, suppose the tuning data consists of two sentences with length 10 and 30, respectively. Then, for traditional tuning its number of training examples is 2; but for search-aware tuning, the total number is 40. More training examples makes our search-aware tuning slower than the traditional tuning.

Table 8 shows the training time comparisons between search-aware tuning and the traditional tuning. From this Table, one can see that both SA-MIRA and SA-PRO are with the same order of magtitude as MIRA and PRO; but SA-MERT is much slower than MERT. The main reason is that, as the training examples increase dramatically, the envelope calculation for exact line search (see (Och, 2003)) in MERT is less efficient than the update based on (sub-)gradient with inexact line search in MIRA and PRO.

One possible solution to speed up SA-MERT is the parallelization but we leave it for future work.

6 Related Work

Many tuning methods have been proposed for SMT so far. These methods differ by the objective function or training mode: their objective functions are based on either evaluation-directed loss (Och, 2003; Galley and Quirk, 2011; Galley et al., 2013) or surrogate loss (Hopkins and May, 2011; Gimpel and Smith, 2012; Eidelman et al., 2013); they are either batch (Och, 2003; Hopkins and May, 2011; Cherry and Foster, 2012) or online mode (Watanabe, 2012; Simianer et al., 2012; Flanigan et al., 2013; Green et al., 2013). These methods share a common characteristic: they learn a weight by iteratively reranking a set of *complete* translations represented by k-best (Och, 2003; Watanabe et al., 2007; Chiang et al., 2008) or lattice (hypergraph) (Tromble et al., 2008; Kumar et al., 2009), and they do not care about search errors that potential *partial* translations may be pruned during decoding, even if they agree with

[4] Under the dense feature setting, MAXFORCE is worse than standard MERT. This result is consistent with that in Figure 12 of (Yu et al., 2013).

[5] We run MAXFORCE on train-r-part, i.e. a part of reachable data instead of the entire reachable data, as we found that more tuning data does not necessarily lead to better testing performance under dense feature setting in our internal experiments.

that their decoders are built on the beam pruning based search.

On the other hand, it is well-known that search errors can undermine the standard training for many beam search based NLP systems (Huang et al., 2012). As a result, Collins and Roark (2004) and Huang et al. (2012) propose the early-update and max-violation update to deal with the search errors. Their idea is to update on prefix or partial hypotheses when the correct solution falls out of the beam. This idea has been successfully used in many NLP tasks and improves the performance over the state-of-art NLP systems (Huang and Sagae, 2010; Huang et al., 2012; Zhang et al., 2013).

Goldberg and Nivre (2012) propose the concept of "dynamic oracle" which is the absolute best potential of a partial derivation, and is more akin to a strictly admissible heuristic. This idea inspired and is closely related to our potential BLEU, except that in our case, computing an admissible heuristic is too costly, so our potential BLEU is more like an average potential.

Gesmundo and Henderson (2014) also consider the rankings between *partial* translation pairs as well. However, they evaluate a partial translation through extending it to a complete translation by *re-decoding*, and thus they need many passes of decoding for many partial translations, while ours only need one pass of decoding for all partial translations and thus is much more efficient. In summary, our tuning framework is more general and has potential to be employed over all the state-of-art tuning methods mentioned above, even though ours is only tested on three popular methods.

7 Conclusions and Future Work

We have presented a simple yet powerful approach of "search-aware tuning" by promoting promising partial derivations, and this idea can be applied to all three popular tuning methods. To solve the key challenge of evaluating partial derivations, we develop a concept of "potential BLEU" inspired by future cost in MT decoding. Extensive experiments confirmed substantial BLEU gains with only dense features. We believe our framework can be applied to sparse feature settings and other translation paradigms, and potentially to other structured prediction problems (such as incremental parsing) as well.

Acknowledgements

We thank the three anonymous reviewers for suggestions, and Kai Zhao and Feifei Zhai for discussions. In particular, we thank reviewer #3 and Chin-Yew Lin for pushing us to think about diversity. This project was supported by DARPA FA8750-13-2-0041 (DEFT), NSF IIS-1449278, a Google Faculty Research Award, and a PSC-CUNY Award.

References

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of NAACL-HLT*, pages 427–436, Montréal, Canada, June.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of EMNLP 2008*.

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *J. Machine Learning Research (JMLR)*, 13:1159–1187.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proc. of ACL 2011*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*.

Vladimir Eidelman, Yuval Marton, and Philip Resnik. 2013. Online relative margin maximization for statistical machine translation. In *Proceedings of ACL*, pages 1116–1126, Sofia, Bulgaria, August.

Jeffrey Flanigan, Chris Dyer, and Jaime Carbonell. 2013. Large-scale discriminative training for statistical machine translation using held-out line search. In *Proceedings of NAACL-HLT*, pages 248–258, Atlanta, Georgia, June.

Michel Galley and Chris Quirk. 2011. Optimal search for minimum error rate training. In *Proceedings of EMNLP*, pages 38–49, Edinburgh, Scotland, UK., July.

Michel Galley, Chris Quirk, Colin Cherry, and Kristina Toutanova. 2013. Regularized minimum error rate training. In *Proceedings of EMNLP*, pages 1948–1959, Seattle, Washington, USA, October.

Andrea Gesmundo and James Henderson. 2014. Undirected machine translation with discriminative reinforcement learning. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, April.

Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proceedings of NAACL-HLT*, pages 221–231, Montréal, Canada, June.

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A systematic exploration of diversity in machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, October.

Yoav Goldberg and Joakim Nivre. 2012. Training deterministic parsers with non-deterministic oracles. In *Proceedings of COLING 2012*.

Spence Green, Sida Wang, Daniel Cer, and Christopher Manning. 2013. Fast and adaptive online training of feature-rich translation models. In *Proc. of ACL 2013*.

P. E. Hart, N. J. Nilsson, and B. Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of EMNLP*.

Liang Huang and David Chiang. 2005. Better k-best Parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT-2005)*.

Liang Huang and David Chiang. 2007. Forest rescoring: Fast decoding with integrated language models. In *Proceedings of ACL*, Prague, Czech Rep., June.

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

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of NAACL*.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of ACL: Demonstrations*.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*, pages 115–124.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of ACL-IJCNLP*, Suntec, Singapore, August.

Zhifei Li and Sanjeev Khudanpur. 2009. Efficient extraction of oracle-best translations from hypergraphs. In *Proceedings of HLT-NAACL Short Papers*.

Adam Lopez. 2008. Statistical machine translation. *ACM Comput. Surv.*, 40(3).

Preslav Nakov, Francisco Guzmn, and Stephan Voge. 2013. A tale about pro and monsters. In *Proceedings of ACL Short Papers*.

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

Franz Joseph Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadephia, USA, July.

Patrick Simianer, Stefan Riezler, and Chris Dyer. 2012. Joint feature selection in distributed stochastic learning for large-scale discriminative training in smt. In *Proceedings of ACL*, pages 11–21, Jeju Island, Korea, July.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of ICSLP*, volume 30, pages 901–904.

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice Minimum Bayes-Risk decoding for statistical machine translation. In *Proceedings of EMNLP*, pages 620–629, Honolulu, Hawaii, October.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of EMNLP-CoNLL*.

Taro Watanabe. 2012. Optimized online rank learning for machine translation. In *Proceedings of NAACL-HLT*, pages 253–262, Montréal, Canada, June.

Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. 2013. Max-violation perceptron and forced decoding for scalable mt training. In *Proceedings of EMNLP 2013*.

Hao Zhang, Liang Huang, Kai Zhao, and Ryan McDonald. 2013. Online learning with inexact hypergraph search. In *Proceedings of EMNLP 2013*.

Kai Zhao, Liang Huang, Haitao Mi, and Abe Ittycheriah. 2014. Hierarchical mt training using max-violation perceptron. In *Proceedings of ACL*, Baltimore, Maryland, June.

Latent-Variable Synchronous CFGs for Hierarchical Translation

Avneesh Saluja and **Chris Dyer**
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
{avneesh,cdyer}@cs.cmu.edu

Shay B. Cohen
University of Edinburgh
Edinburgh EH8 9AB, UK
scohen@inf.ed.ac.uk

Abstract

Data-driven refinement of non-terminal categories has been demonstrated to be a reliable technique for improving monolingual parsing with PCFGs. In this paper, we extend these techniques to learn latent refinements of single-category synchronous grammars, so as to improve translation performance. We compare two estimators for this latent-variable model: one based on EM and the other is a spectral algorithm based on the method of moments. We evaluate their performance on a Chinese–English translation task. The results indicate that we can achieve significant gains over the baseline with both approaches, but in particular the moments-based estimator is both faster and performs better than EM.

1 Introduction

Translation models based on synchronous context-free grammars (SCFGs) treat the translation problem as a context-free parsing problem. A parser constructs trees over the input sentence by parsing with the source language projection of a synchronous CFG, and each derivation induces translations in the target language (Chiang, 2007). However, in contrast to syntactic parsing, where linguistic intuitions can help elucidate the "right" tree structure for a grammatical sentence, no such intuitions are available for synchronous derivations, and so learning the "right" grammars is a central challenge.

Of course, learning synchronous grammars from parallel data is a widely studied problem (Wu, 1997; Blunsom et al., 2008; Levenberg et al., 2012, *inter alia*). However, there has been less exploration of learning rich non-terminal categories, largely because previous efforts to learn such categories have been coupled with efforts to learn derivation structures—a computationally formidable challenge. One popular approach has been to derive categories from source and/or target monolingual grammars (Galley et al., 2004; Zollmann and Venugopal, 2006; Hanneman and Lavie, 2013). While often successful, accurate parsers are not available in many languages: a more appealing approach is therefore to learn the category structure from the data itself.

In this work, we take a different approach to previous work in synchronous grammar induction by assuming that reasonable tree structures for a parallel corpus can be chosen heuristically, and then, fixing the trees (thereby enabling us to sidestep the worst of the computational issues), we learn non-terminal categories as latent variables to explain the distribution of these synchronous trees. This technique has a long history in monolingual parsing (Petrov et al., 2006; Liang et al., 2007; Cohen et al., 2014), where it reliably yields state-of-the-art phrase structure parsers based on generative models, but we are the first to apply it to translation.

We first generalize the concept of latent PCFGs to latent-variable SCFGs (§2). We then follow by a presentation of the tensor-based formulation for our parameters, a representation that makes it convenient to marginalize over latent states. Subsequently, two methods for parameter estimation are presented (§4): a spectral approach based on the method of moments, and an EM-based likelihood maximization. Results on a Chinese–English evaluation set (§5) indicate significant gains over baselines and point to the promise of using latent-variable synchronous grammars in conjunction with a smaller, simpler set of rules instead of unwieldy and bloated grammars extracted via existing heuristics, where a large number of context-independent but un-generalizable rules are utilized. Hence, the hope is that this work pro-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1953–1964,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

motes the move towards translation models that directly model the conditional likelihood of translation rules via (potentially feature-rich) latent-variable models which leverage information contained in the synchronous tree structure, instead of relying on a heuristic set of features based on empirical relative frequencies (Koehn et al., 2003) from non-hierarchical phrase-based translation.

2 Latent-Variable SCFGs

Before discussing parameter learning, we introduce latent-variable synchronous context-free grammars (L-SCFGs) and discuss an inference algorithm for marginalizing over latent states.

We extend the definition of L-PCFGs (Matsuzaki et al., 2005; Petrov et al., 2006) to synchronous grammars as used in machine translation (Chiang, 2007). A latent-variable SCFG (L-SCFG) is a 6-tuple $(\mathcal{N}, m, n_s, n_t, \pi, t)$ where:

- $\mathcal{N}$ is a set of non-terminal (NT) symbols in the grammar. For hierarchical phrase-based translation (HPBT), the set consists of only two symbols, **X** and a goal symbol **S**.
- $[m]$ is the set of possible hidden states associated with NTs. Aligned pairs of NTs across the source and target languages share the same hidden state.
- $[n_s]$ is the set of source side words, i.e., the source-side vocabulary, with $[n_s] \cap \mathcal{N} = \emptyset$.
- $[n_t]$ is the set of target side words, i.e., the target-side vocabulary, with $[n_t] \cap \mathcal{N} = \emptyset$.
- The synchronous production rules compose a set $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1 \cup \mathcal{R}_2$:
  - Arity 2 (binary) rules ($\mathcal{R}_2$):

    $$a(h_1) \rightarrow \langle \alpha_1 b(h_2) \alpha_2 c(h_3) \alpha_3, \beta_1 b(h_2) \beta_2 c(h_3) \beta_3 \rangle$$

    or

    $$a(h_1) \rightarrow \langle \alpha_1 b(h_2) \alpha_2 c(h_3) \alpha_3, \beta_1 c(h_2) \beta_2 b(h_3) \beta_3 \rangle$$

    where $a, b, c \in \mathcal{N}$, $h_1, h_2, h_3 \in [m]$, $\alpha_1, \alpha_2, \alpha_3 \in [n_s]^*$ and $\beta_1, \beta_2, \beta_3 \in [n_t]^*$.
  - Arity 1 (unary) rules ($\mathcal{R}_1$):

    $$a(h_1) \rightarrow \langle \alpha_1 b(h_2) \alpha_2, \beta_1 b(h_2) \beta_2 \rangle$$

    where $a, b \in \mathcal{N}$, $h_1, h_2 \in [m]$, $\alpha_1, \alpha_2 \in [n_s]^*$ and $\beta, \beta_2 \in [n_t]^*$.
  - Pre-terminal rules ($\mathcal{R}_0$): $a(h_1) \rightarrow \langle \alpha, \beta \rangle$ where $a \in \mathcal{N}$, $\alpha \in [n_t]^*$ and $\beta \in [n_s]^*$.

  Each of these rules is associated with a probability $t(a(h_1) \rightarrow \gamma | a, h_1)$ where γ is the right-hand side (RHS) of the rule.
- For $a \in \mathcal{N}$, $h \in [m]$, $\pi(a, h)$ is a parameter specifying the root probability of $a(h)$.

A skeletal tree (s-tree) for a sentence is the set of rules in the synchronous derivation of that sentence, without any additional latent state information or decoration. A full tree consists of an s-tree $r_1, \ldots, r_N$ together with values $h_1, \ldots, h_N$ for every NT in the tree. An important point to keep in mind in comparison to L-PCFGs is that the right-hand side (RHS) non-terminals of synchronous rules are aligned pairs across the source and target languages.

In this work, we refine the one-category grammar introduced by Chiang (2007) for HPBT in order to learn additional latent NT categories. Thus, the following discussion is restricted to these kinds of grammars, although the method is equally applicable in other scenarios, e.g., the extended tree-to-string transducer (**xRs**) formalism (Huang et al., 2006; Graehl et al., 2008) commonly used in syntax-directed translation, and phrase-based MT (Koehn et al., 2003).

Marginal Inference with L-SCFGs. For a parameter t of rule r, the latent state h_1 attached to the left-hand side (LHS) NT of r is associated with the outside tree for the sub-tree rooted at the LHS, and the states attached to the RHS NTs are associated with the inside trees of that NT. Since we do not assume conditional independence of these states, we need to consider all possible interactions, which can be compactly represented as a 3rd-order tensor in the case of a binary rule, a matrix (i.e., a 2nd-order tensor) for unary rules, and a vector for pre-terminal (lexical) rules. Preferences for certain outside-inside tree combinations are reflected in the values contained in these tensor structures. In this manner, we intend to capture interactions between non-local context of a phrase, which can typically be represented via features defined over outside trees of the node spanning the phrase, and the interior context, correspondingly defined via features over the inside trees. We refer to these tensor structures collectively as C^r for rules $r \in \mathcal{R}$, which encompass the parameters t.

For $r \in \mathcal{R}_0 : C^r \in \mathbb{R}^{m \times 1}$; similarly for $r \in \mathcal{R}_1 : C^r \in \mathbb{R}^{m \times m}$ and $r \in \mathcal{R}_2 : C^r \in \mathbb{R}^{m \times m \times m}$. We also maintain a vector $C^{\mathbf{S}} \in \mathbb{R}^{1 \times m}$ corresponding to the parameters $\pi(\mathbf{S}, h)$ for the

Inputs: Sentence $f_1 \dots f_N$, L-SCFG $(\mathcal{N}, S, m, n)$, parameters $C^r \in \mathbb{R}^{(m \times m \times m)}$, $\in \mathbb{R}^{(m \times m)}$, or $\in \mathbb{R}^{(m \times 1)}$ for all $r \in \mathcal{R}$, $C^{\mathbf{S}} \in \mathbb{R}^{(1 \times m)}$, hypergraph $\mathcal{H}$.
Data structures:
For each node $q \in \mathcal{H}$:

- $\boldsymbol{\alpha}(q) \in \mathbb{R}^{m \times 1}$ is a column vector of inside terms.
- $\boldsymbol{\beta}(q) \in \mathbb{R}^{1 \times m}$ is a row vector of outside terms.
- For each incoming edge $e \in \mathbf{B}(q)$ to node q, $\mu(e)$ is a marginal probability for edge (rule) e.

Algorithm:
▷ *Inside Computation*
For nodes q in topological order in $\mathcal{H}$,
  $\boldsymbol{\alpha}(q) = \mathbf{0}$
  For each incoming edge $e \in \mathbf{B}(q)$,
    tail = **t**(e), rule = **r**(e)
    if |tail| $= 0$, then $\boldsymbol{\alpha}(q) = \boldsymbol{\alpha}(q) + C^{\text{rule}}$
    else if |tail| $= 1$, then $\boldsymbol{\alpha}(q) = \boldsymbol{\alpha}(q) + C^{\text{rule}} \times_1 \boldsymbol{\alpha}(\text{tail}_0)$
    else if |tail| $= 2$, then $\boldsymbol{\alpha}(q) = \boldsymbol{\alpha}(q) + C^{\text{rule}} \times_2 \boldsymbol{\alpha}(\text{tail}_1) \times_1 \boldsymbol{\alpha}(\text{tail}_0)$
▷ *Outside Computation*
For $q \in \mathcal{H}$,
  $\boldsymbol{\beta}(q) = \mathbf{0}$
$\boldsymbol{\beta}(\text{goal}) = C^{\mathbf{S}}$
For q in reverse topological order in $\mathcal{H}$,
  For each incoming edge $e \in \mathbf{B}(q)$,
    tail = **t**(e), rule = **r**(e)
    if |tail| $= 1$, then
      $\boldsymbol{\beta}(\text{tail}_0) = \boldsymbol{\beta}(\text{tail}_0) + \boldsymbol{\beta}(q) \times_0 C^{\text{rule}}$
    else if |tail| $= 2$, then
      $\boldsymbol{\beta}(\text{tail}_0) = \boldsymbol{\beta}(\text{tail}_0) + \boldsymbol{\beta}(q) \times_0 C^{\text{rule}} \times_2 \boldsymbol{\alpha}(\text{tail}_1)$
      $\boldsymbol{\beta}(\text{tail}_1) = \boldsymbol{\beta}(\text{tail}_1) + \boldsymbol{\beta}(q) \times_0 C^{\text{rule}} \times_1 \boldsymbol{\alpha}(\text{tail}_0)$
▷*Edge Marginals*
Sentence probability $g = \boldsymbol{\alpha}(\text{goal}) \times \boldsymbol{\beta}(\text{goal})$
For edge $e \in \mathcal{H}$,
  head = **h**(e), tail = **t**(e), rule = **r**(e)
  if |tail| $= 0$, then $\mu(e) = (\boldsymbol{\beta}(\text{head}) \times_0 C^{\text{rule}})/g$
  else if |tail| $= 1$, then $\mu(e) = (\boldsymbol{\beta}(\text{head}) \times_0 C^{\text{rule}} \times_1 \boldsymbol{\alpha}(\text{tail}_0))/g$
  else if |tail| $= 2$, then $\mu(e) = (\boldsymbol{\beta}(\text{head}) \times_0 C^{\text{rule}} \times_2 \boldsymbol{\alpha}(\text{tail}_1) \times_1 \boldsymbol{\alpha}(\text{tail}_0))/g$

Figure 1: The tensor form of the hypergraph inside-outside algorithm, for calculation of rule marginals $\mu(e)$. A slight simplification in the marginal computation yields NT marginals for spans $\mu(\mathbf{X}, i, j)$. **B**(q) returns the incoming hyperedges for node q, and **h**(e), **t**(e), **r**(e) return the head node, tail nodes, and rule for hyperedge e.

goal node (root). These parameters participate in tensor-vector operations: a 3rd-order tensor C^{r_2} can be multiplied along each of its three modes ($\times_0, \times_1, \times_2$), and if multiplied by an $m \times 1$ vector, will produce an $m \times m$ matrix.[1] Note that matrix multiplication can be represented by $\times_1$ when multiplying on the right and $\times_0$ when multiplying on the left of the matrix. The decoder computes marginal probabilities for each skeletal rule in the parse forest of a source sentence by marginalizing over the latent states, which in practice corresponds to simple tensor-vector products. This operation is not dependent on the manner in which the parameters were estimated.

Figure 1 presents the tensor version of the inside-outside algorithm for decoding L-SCFGs. The algorithm takes as input the parse forest of the source sentence represented as a hypergraph (Klein and Manning, 2001), which is computed using a bottom-up parser with Earley-style rules similar to the algorithm in Chiang (2007). Hypergraphs are a compact way to represent a forest of multiple parse trees. Each node in the hypergraph corresponds to an NT span, and can have multiple incoming and outgoing hyperedges. Hyperedges, which connect one or more tail nodes to a single head node, correspond exactly to rules, and tail or head nodes correspond to children (RHS NTs) or parent (LHS NT). The function **B**(q) returns all incoming hyperedges to a node q, i.e., all rules such that the LHS NT of the rule corresponds to the NT span of the node q. The algorithm computes inside and outside probabilities over the hypergraph using the tensor representations, and converts these probabilities to marginal rule probabilities. It is similar to the version presented in Cohen et al. (2014), but adapted to hypergraph parse forests.

The complexity of this decoding algorithm is $\mathcal{O}(n^3 m^3 |G|)$ where n is the length of the input sentence, m is the number of latent states, and $|G|$ is the number of production rules in the grammar *without* latent-variable annotations (i.e., $m = 1$).[2] The bulk of the computation is a series of tensor-vector products of relatively small size (each dimension is of length m), which can be computed very quickly and in parallel. The tensor computations can be significantly sped up using techniques described by Cohen and Collins (2012), so that they are linear in m and not cubic.

3 Derivation Trees for Parallel Sentences

To estimate the parameters t and π of an L-SCFG (discussed in detail in the next section), we assume the existence of a dataset composed of synchronous s-trees, which can be acquired from word alignments. Normally in phrase-based translation models, we consider all possible phrase

[1] This operation is sometimes called a contraction.

[2] In practice, the term $m^3|G|$ can be replaced with a smaller term, which separates the rules in G by the number of NTs on the RHS. This idea relates to the notion of "effective grammar size" which we discuss in §5.

pairs consistent with the word alignments and estimate features based on surface statistics associated with the phrase pairs or rules. The weights of these features are then learned using a discriminative training algorithm (Och, 2003; Chiang, 2012, *inter alia*). In contrast, in this work we restrict the number of possible synchronous derivations for each sentence pair to just one; thus, derivation forests do not have to be considered, making parameter estimation more tractable.[3]

To achieve this objective, for each sentence in the training data we extract the **minimal** set of synchronous rules consistent with the word alignments, as opposed to the **composed** set of rules (Galley et al., 2006). Composed rules are ones that can be formed from smaller rules in the grammar; with these rules, there are multiple synchronous trees consistent with the alignments for a given sentence pair, and thus the total number of applicable rules can be combinatorially larger than if we just consider the set of rules that cannot be formed from other rules, namely the minimal rules. The rule types across all sentence pairs are combined to form a minimal grammar.[4] To extract a set of minimal rules, we use the linear-time extraction algorithm of Zhang et al. (2008). We give a rough description of their method below, and refer the reader to the original paper for additional details.

The algorithm returns a complete minimal derivation tree for each word-aligned sentence pair, and generalizes an approach for finding all common intervals (pairs of phrases such that no word pair in the alignment links a word inside the phrase to a word outside the phrase) between two permutations (Uno and Yagiura, 2000) to sequences with many-to-many alignment links between the two sides, as in word alignment. The key idea is to encode all phrase pairs of a sentence alignment in a tree of size proportional to the source sentence length, which they call the normalized decomposition tree. Each node corresponds to a phrase pair, with larger phrase spans represented by higher nodes in the tree. Constructing the tree is analogous to finding common intervals in two permutations, a property that they leverage to propose a linear-time algorithm for tree extraction. Converting the tree to a set of minimal SCFG rules for the sentence pair is straightforward, by replacing nodes corresponding to spans with lexical items or NTs in a bottom-up manner.[5]

By using minimal rules as a starting point instead of the traditional heuristically-extracted rules (Chiang, 2007) or arbitrary compositions of minimal rules (Galley et al., 2006), we are also able to explore the transition from minimal rules to composed ones in a principled manner by encoding contextual information through the latent states. Thus, a beneficial side effect of our refinement process is the creation of more context-specific rules without increasing the overall size of the baseline grammar, instead holding this information in our parameters C^r.

4 Parameter Estimation for L-SCFGs

We explore two methods for estimating the parameters C^r of the model: a likelihood-maximization approach based on EM (Dempster et al., 1977), and a spectral approach based on the method of moments (Hsu et al., 2009; Cohen et al., 2014), where we identify a subspace using a singular value decomposition (SVD) of the cross-product feature space between inside and outside trees and estimate parameters in this subspace.

Figure 2 presents a side-by-side comparison of the two algorithms, which we discuss in this section. In the spectral approach, we base our parameter estimates on low-rank representations of moments of features, while EM explicitly maximizes a likelihood criterion. The parameter estimation algorithms are relatively similar, but in lieu of sparse feature functions in the spectral case, EM uses partial counts estimated with the current set of parameters. The nature of EM allows it to be susceptible to local optima, while the spectral approach comes with guarantees on obtaining the global optimum (Cohen et al., 2014). Lastly, computing the SVD and estimating parameters in the low-rank space is a one-shot operation, as opposed to the iterative procedure of EM, and therefore is much more computationally efficient.

4.1 Estimation with Spectral Method

We generalize the parameter estimation algorithm presented in Cohen et al. (2013) to the syn-

[3] For future work, we will consider efficient algorithms for parameter estimation over derivation forests, since there may be multiple valid ways to explain the sentence pair via a synchronous tree structure.

[4] Table 2 presents a comparison of grammar sizes for our experiments (§5.1).

[5] We filtered rules with arity 3 and above (i.e., containing more than 3 NTs on the RHS). While the L-SCFG formalism is perfectly capable of handling such cases, it would have resulted in higher order tensors for our parameter structures.

Inputs:
Training examples $(r^{(i)}, t^{(i,1)}, t^{(i,2)}, t^{(i,3)}, o^{(i)}, b^{(i)})$ for $i \in \{1 \dots M\}$, where $r^{(i)}$ is a context free rule; $t^{(i,1)}$, $t^{(i,2)}$, and $t^{(i,3)}$ are inside trees; $o^{(i)}$ is an outside tree; and $b^{(i)} = 1$ if the rule is at the root of tree, 0 otherwise. A function ϕ that maps inside trees t to feature-vectors $\phi(t) \in \mathbb{R}^d$. A function ψ that maps outside trees o to feature-vectors $\psi(o) \in \mathbb{R}^{d'}$.

Algorithm:

▷ *Step 0: Singular Value Decomposition*

- Compute the SVD of Eq. 1 to calculate matrices $\hat{U} \in \mathbb{R}^{(d \times m)}$ and $\hat{V} \in \mathbb{R}^{(d' \times m)}$.

▷ *Step 1: Projection*

$$Y(t) = U^\top \phi(t)$$
$$Z(o) = \Sigma^{-1} V^\top \psi(o)$$

▷ *Step 2: Calculate Correlations*

$$\hat{E}^r = \begin{cases} \frac{\sum_{o \in Q^r} Z(o)}{|Q^r|} & \text{if } r \in \mathcal{R}_0 \\ \frac{\sum_{(o,t) \in Q^r} Z(o) \otimes Y(t)}{|Q^r|} & \text{if } r \in \mathcal{R}_1 \\ \frac{\sum_{(o,t^2,t^3) \in Q^r} Z(o) \otimes Y(t^2) \otimes Y(t^3)}{|Q^r|} & \text{if } r \in \mathcal{R}_2 \end{cases}$$

Q^r is the set of outside-inside tree triples for binary rules, outside-inside tree pairs for unary rules, and outside trees for pre-terminals.

▷ *Step 3: Compute Final Parameters*

- For all $r \in \mathcal{R}$,
$$\hat{C}^r = \frac{\text{count}(r)}{M} \times \hat{E}^r$$
- For all $r^{(i)} \in \{1, \dots, M\}$ such that $b^{(i)}$ is 1,
$$\hat{C}^{\mathbf{S}} = \hat{C}^{\mathbf{S}} + \frac{Y(t^{(i,1)})}{|Q^{\mathbf{S}}|}$$

$Q^{\mathbf{S}}$ is the set of trees at the root.

(a) The spectral learning algorithm for estimating parameters of an L-SCFG.

Inputs:
Training examples $(r^{(i)}, t^{(i,1)}, t^{(i,2)}, t^{(i,3)}, o^{(i)}, b^{(i)})$ for $i \in \{1 \dots M\}$, where $r^{(i)}$ is a context free rule; $t^{(i,1)}$, $t^{(i,2)}$, and $t^{(i,3)}$ are inside trees; $o^{(i)}$ is an outside tree; $b^{(i)} = 1$ if the rule is at the root of tree, 0 otherwise; and MAX_ITERATIONS.

Algorithm:

▷ *Step 0: Parameter Initialization*

For rule $r \in \mathcal{R}$,

- if $r \in \mathcal{R}_0$: initialize $\hat{C}^r \in \mathbb{R}^{m \times 1}$
- if $r \in \mathcal{R}_1$: initialize $\hat{C}^r \mathbb{R}^{m \times m}$
- if $r \in \mathcal{R}_2$: initialize $\hat{C}^r \mathbb{R}^{m \times m \times m}$

Initialize $\hat{C}^{\mathbf{S}} \in \mathbb{R}^{m \times 1}$
$\hat{C}^r_0 = \hat{C}^r, \hat{C}^{\mathbf{S}}_0 = \hat{C}^{\mathbf{S}}$
For iteration $t = 1, \dots,$ MAX_ITERATIONS,

- Expectation Step:

  ▷ *Estimate Y and Z*

  Compute partial counts and total tree probabilities g for all t and o using Fig. 1 and parameters $\hat{C}^r_{t-1}, \hat{C}^{\mathbf{S}}_{t-1}$.

  ▷ *Calculate Correlations*

$$\hat{E}^r = \begin{cases} \sum_{o,g \in Q^r} \frac{Z(o)}{g} & \text{if } r \in \mathcal{R}_0 \\ \sum_{(o,t,g) \in Q^r} \frac{Z(o) \otimes Y(t)}{g} & \text{if } r \in \mathcal{R}_1 \\ \sum_{(o,t^2,t^3,g) \in Q^r} \frac{Z(o) \otimes Y(t^2) \otimes Y(t^3)}{g} & \text{if } r \in \mathcal{R}_2 \end{cases}$$

  ▷ *Update Parameters*

  For all $r \in \mathcal{R}$, $\hat{C}^r_t = \hat{C}^r_{t-1} \odot \hat{E}^r$

  For all $r^{(i)} \in \{1, \dots, M\}$ such that $b^{(i)}$ is 1, $\hat{C}^{\mathbf{S}}_t = \hat{C}^{\mathbf{S}}_t + (\hat{C}^{\mathbf{S}}_{t-1} \odot Y(r^{(i)}))/g$

  $Q^{\mathbf{S}}$ is the set of trees at the root.

- Maximization Step

  if $r \in \mathcal{R}_0$: $\forall h_1 : \hat{C}^r(h_1) = \frac{\hat{C}^r(h_1)}{\sum_{r'=r} \sum_{h_1} \hat{C}^{r'}(h_1)}$

  if $r \in \mathcal{R}_1$: $\forall h_1, h_2 : \hat{C}^r(h_1, h_2) = \frac{\hat{C}^r(h_1, h_2)}{\sum_{r'=r} \sum_{h_2} \hat{C}^{r'}(h_1, h_2)}$

  if $r \in \mathcal{R}_2$: $\forall h_1, h_2, h_3 : \hat{C}^r(h_1, h_2, h_3) = \frac{\hat{C}^r(h_1, h_2, h_3)}{\sum_{r'=r} \sum_{h_2, h_3} \hat{C}^{r'}(h_1, h_2, h_3)}$

  if LHS$(r) = \mathbf{S}$: $\forall h_1 : \hat{C}^r(h_1) = \frac{\hat{C}^r(h_1)}{\sum_{r'=r} \sum_{h_1} \hat{C}^{r'}(h_1)}$

(b) The EM-based algorithm for estimating parameters of an L-SCFG.

Figure 2: The two parameter estimation algorithms proposed for L-SCFGs; (**a**) method of moments; (**b**) expectation maximization. $\odot$ is the element-wise multiplication operator.

chronous or bilingual case. The central concept of the spectral parameter estimation algorithm is to learn an m-dimensional representation of inside and outside trees by defining these trees in terms of features, in combination with a projection step (SVD), with the hope being that the lower-dimensional space captures the syntactic and semantic regularities among rules from the sparse feature space. Every NT in an s-tree has an associated inside and outside tree; the inside tree contains the entire sub-tree at and below the NT, and the outside tree is everything else in the synchronous s-tree except the inside tree. The inside feature function ϕ maps the domain of inside tree

fragments to a d-dimensional Euclidean space, and the outside feature function ψ maps the domain of outside tree fragments to a d'-dimensional space. The specific features we used are discussed in §5.2.

Let $\mathcal{O}$ be the set of all tuples of inside-outside trees in our training corpus, whose size is equivalent to the number of rule tokens (occurrences in the corpus) M, and let $\phi(t) \in \mathbb{R}^{d\times 1}$, $\psi(o) \in \mathbb{R}^{d'\times 1}$ be the inside and outside feature functions for inside tree t and outside tree o. By computing the outer product $\otimes$ between the inside and outside feature vectors for each pair and aggregating, we obtain the empirical inside-outside feature covariance matrix:

$$\hat{\Omega} = \frac{1}{|\mathcal{O}|} \sum_{(o,t)\in\mathcal{O}} \phi(t) \left(\psi(o)\right)^{\top} \qquad (1)$$

If m is the desired latent space dimension, we compute an m-rank truncated SVD of the empirical covariance matrix $\hat{\Omega} \approx U\Sigma V^{\top}$, where $U \in \mathbb{R}^{d\times m}$ and $V \in \mathbb{R}^{d'\times m}$ are the matrices containing the left and right singular vectors, and $\Sigma \in \mathbb{R}^{m\times m}$ is a diagonal matrix containing the m-largest singular values along its diagonal.

Figure 2a provides the remaining steps in the algorithm. The M training examples are obtained by considering all nodes in all of the synchronous s-trees given as input. In step 1, for each inside and outside tree, we project its high-dimensional representation to the m-dimensional latent space. Using the m-dimensional representations for inside and outside trees, in step 2 for each rule type r we compute the covariance between the inside tree vectors and the outside tree vector using the *tensor product*, a generalized outer product to compute covariances between more than two random vectors. For binary rules, with two child inside vectors and one outside vector, the result $\hat{E}^r$ is a 3-mode tensor; for unary rules, a regular matrix, and for pre-terminal rules with no right-hand side non-terminals, a vector. The final parameter estimate is then the associated tensor/matrix/vector, scaled by the maximum likelihood estimate of the rule r, as in step 3.

The corresponding theoretical guarantees from Cohen et al. (2014) can also be generalized to the synchronous case. $\hat{\Omega}$ is an empirical estimate of the true covariance matrix Ω, and if Ω has rank m, then the marginals computed using the spectrally-estimated parameters will converge to the true marginals, with the sample complexity for convergence inversely proportional to a polynomial function of the m^{th} largest singular value of Ω.

4.2 Estimation with EM

A likelihood maximization approach can also be used to learn the parameters of an L-SCFG. Parameters are initialized by sampling each parameter value $\hat{C}^r(h_1, h_2, h_3)$ from the interval $[0, 1]$ uniformly at random.[6] We first decode the training corpus using an existing set of parameters to compute the inside and outside probability vectors associated with NTs for every rule in each s-tree, constrained to the tree structure of the training example. These probabilities can be computed using the decoding algorithm in Figure 1 (where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ correspond to the inside and outside probabilities respectively), except the parse forest consists of a single tree only. These vectors represent partial counts over latent states. We then define functions Y and Z (analogous to the spectral case) which map inside and outside tree instances to m-dimensional vectors containing these partial counts. In the spectral case, Y and Z are estimated just once, while in the case of EM they have to be re-estimated at each iteration.

The expectation step thus consists of computing the partial counts of inside and outside trees t and o, i.e., recovering the functions Y and Z, and updating parameters C^r by computing correlations, which involves summing over partial counts (across all occurrences of a rule in the corpus). Each partial count's contribution is divided by a normalization factor g, which is the total probability of the tree which t or o is part of. Note that unlike the spectral case, there is a specific normalization factor for each inside-outside tuple. Lastly, the correlations are scaled by the existing parameter estimates.

To obtain the next set of parameters, in the maximization step we normalize $\hat{C}^r$ for $r \in \mathcal{R}$ such that for every h_1, $\sum_{r'=r,h_2,h_3} \hat{C}^{r'}(h_1, h_2, h_3) = 1$ for $r \in \mathcal{R}_2$, $\sum_{r'=r,h_2} \hat{C}^{r'}(h_1, h_2) = 1$ for $r \in \mathcal{R}_1$, and $\sum_{r'=r,h_2} \hat{C}^{r'}(h_2) = 1$ for $r \in \mathcal{R}_0$. We also normalize the root rule parameters $\hat{C}^r$ where $\text{LHS}(r) = \mathbf{S}$. It is also possible to add sparse, overlapping features to an EM-based estimation

[6] In our experiments, we also tried the initialization scheme described in Matsuzaki et al. (2005), but found that it provided little benefit.

procedure (Berg-Kirkpatrick et al., 2010) and we leave this extension for future work.

5 Experiments

The goal of the experimental section is to evaluate the performance of the latent-variable SCFG in comparison to a baseline without any additional NT annotations (MIN-GRAMMAR), and to compare the performance of the two parameter estimation algorithms. We also compare L-SCFGs to a HIERO baseline (Chiang, 2007). The language pair of evaluation is Chinese–English (ZH-EN).

We score translations using BLEU (Papineni et al., 2002). The latent-variable model is integrated into the standard MT pipeline by computing marginal probabilities for each rule in the parse forest of a source sentence using the algorithm in Figure 1 with the parameters estimated through the algorithms in Figure 2, and is added as a feature for the rule during MERT (Och, 2003). These probabilities are conditioned on the LHS (**X**), and are thus joint probabilities for a source-target RHS pair. We also write out as features the conditional relative frequencies $\hat{P}(e|f)$ and $\hat{P}(f|e)$ as estimated by our latent-variable model, i.e., conditioned on the source and target RHS.

Overall, we find that both the spectral and the EM-based estimators improve upon a minimal grammar baseline with only a single category, but the spectral approach does better. In fact, it matches the performance of the standard HIERO baseline, despite learning on top of a minimal grammar.

5.1 Data and Baselines

The ZH-EN data is the BTEC parallel corpus (Paul, 2009); we combine the first and second development sets in one, and evaluate on the third development set. The development and test sets are evaluated with 16 references. Statistics for the data are shown in Table 1. We used the CDEC decoder (Dyer et al., 2010) to extract word alignments and the baseline hierarchical grammars, MERT tuning, and decoding. We used a 4-gram language model built from the target-side of the parallel training data. The Python-based implementation of the tensor-based decoder, as well as the parameter estimation algorithms is available at `github.com/asaluja/spectral-scfg/`.

The baseline HIERO system uses a grammar extracted by applying the commonly used heuristics (Chiang, 2007). Each rule is decorated with two lexical and phrasal features corresponding to the forward $(e|f)$ and backward $(f|e)$ conditional log frequencies, along with the log joint frequency (e, f), the log frequency of the source phrase (f), and whether the phrase pair or the source phrase is a singleton. Weights for the language model (and language model OOV), glue rule, and word penalty are also tuned. The MIN-GRAMMAR baseline[7] maintains the same set of weights.

	ZH-EN
TRAIN (SRC)	334K
TRAIN (TGT)	366K
DEV (SRC)	7K
DEV (TGT)	7.6K
TEST (SRC)	3.8K
TEST (TGT)	3.9K

Table 1: Corpus statistics (in words). For the target DEV and TEST statistics, we take the first reference.

Grammar	Number of Rules
HIERO	1.69M
MIN-GRAMMAR	59K
LV $m = 1$	27.56K
LV $m = 8$	3.18M
LV $m = 16$	22.22M

Table 2: Grammar sizes for the different systems; for the latent-variable models, effective grammar sizes are provided.

Grammar sizes are presented in Table 2. For the latent-variable models, we provide the effective grammar size, where the number of NTs on the RHS of a rule is taken into account when computing the grammar size, by assuming each possible latent variable configuration amongst the NTs generates a different rule. Furthermore, all singletons are mapped to the OOV rule, while we include singletons in MIN-GRAMMAR.[8] Hence, effective grammar size can be computed as $m(1 + |\mathcal{R}_0^{>1}|) + m^2|\mathcal{R}_1| + m^3|\mathcal{R}_2|$, where $\mathcal{R}_0^{>1}$ is the set of pre-terminal rules that occur more than once.

5.2 Spectral Features

We use the following set of sparse, binary features in the spectral learning process:

[7] Code to extract the minimal derivation trees is available at `www.cs.rochester.edu/u/gildea/mt/`.

[8] This OOV mapping is done so that the latent-variable model can handle unknown tokens.

- **Rule Indicator.** For the inside features, we consider the rule production containing the current non-terminal on the left-hand side, as well as the rules of the children (distinguishing between left and right children for binary rules). For the outside features, we consider the parent rule production along with the rule production of the sibling (if it exists).
- **Lexical.** for both the inside and outside features, any lexical items that appear in the rule productions are recorded. Furthermore, we consider the first and last words of spans (left and right child spans for inside features, distinguishing between the two if both exist, and sibling span for outside features). Source and target words are treated separately.
- **Length.** the span length of the tree and each of its children for inside features, and the span length of the parent and sibling for outside features.

In our experiments, we instantiated a total of 170,000 rule indicator features, 155,000 lexical features, and 80 length features.

5.3 Chinese–English Experiments

Table 3 presents a comprehensive evaluation of the ZH-EN experimental setup. The first section consists of the various baselines we consider. In addition to the aforementioned baselines, we evaluated a setup where the spectral parameters simply consist of the joint maximum likelihood estimates of the rules. This baseline should perform *en par* with MIN-GRAMMAR, which we see is the case on the development set. The performance on the test set is better though, primarily because we also include the reverse log relative frequency $(f|e)$ computed from the latent-variable model as an additional feature in MERT. Furthermore, in line with previous work (Galley et al., 2006) which compares minimal and composed rules, we find that minimal grammars take a hit of more than 2.5 BLEU points on the development set, compared to composed (HIERO) grammars. The $m = 1$ spectral baseline with only rule indicator features performs slightly better than the minimal grammar baseline, since it overtly takes into account inside-outside tree combination preferences in the parameters, but improvement is minimal with one latent state naturally and the performance on the test set is in line with the MLE baseline.

On top of the baselines, we looked at a number of feature combinations and latent states for the spectral and EM-estimated latent-variable models. For the spectral models, we tuned MERT parameters separately for each rank on a set of parameters estimated from rule indicator features only; subsequent variations within a given rank, e.g., the addition of lexical or length features or smoothing, were evaluated with the same set of rank-specific weights from MERT. For EM, we ran parameter estimation with 5 randomly initialized starting points for 50 iterations; we tuned the MERT parameters with EM parameters obtained after 25$^{\text{th}}$ iterations. Similar to the spectral experiments, we fixed the MERT weight values and evaluated BLEU performance with parameters after every 5 iterations and chose the iteration with the highest score on the development set. The results are averaged over the 5 initializations, with standard deviation in parentheses.

Firstly, we can see a clear dependence on rank, with peak performance for the spectral and EM models occurring at $m = 16$. In this instance, the spectral model roughly matches the performance of the HIERO baseline, but it only uses rules extracted from a minimal grammar, whose size is a fraction of the HIERO grammar. The gains seem to level off at this rank; additional ranks seem to add noise to the parameters. Feature-wise, additional lexical and length features add little, prob-

		BLEU	
	Setup	Dev	Test
Baselines	HIERO	46.08	55.31
	MIN-GRAMMAR	43.38	51.78
	MLE	43.24	52.80
Spectral	$m = 1$ RI	44.18	52.62
	$m = 8$ RI	44.60	53.63
	$m = 16$ RI	46.06	55.83
	m=16 RI+Lex+Sm	46.08	55.22
	m=16 RI+Lex+Len	45.70	55.29
	m=24 RI+Lex	43.00	51.28
	m=32 RI+Lex	43.06	52.16
EM	$m = 8$	40.53 (0.2)	49.78 (0.5)
	$m = 16$	42.85 (0.2)	52.93 (0.9)
	$m = 32$	41.07 (0.4)	49.95 (0.7)

Table 3: Results for the ZH-EN corpus, comparing across the baselines and the two parameter estimation techniques. RI, Lex, and Len correspond to the rule indicator, lexical, and length features respectively, and Sm denotes smoothing. For the EM experiments, we selected the best scoring iteration by tuning weights for parameters obtained after 25 iterations and evaluating other parameters with these weights. Results for EM are averaged over 5 starting points, with standard deviation given in parentheses. Spectral, EM, and MLE performances compared to the MIN-GRAMMAR baseline are statistically significant ($p < 0.01$).

ably because much of this information is encapsulated in the rule indicator features. For EM, $m = 16$ outperforms the minimal grammar baseline, but is not at the level of the spectral results. All EM, spectral, and MLE results are statistically significant ($p < 0.01$) with respect to the MIN-GRAMMAR baseline (Zhang et al., 2004), and the improvement over the HIERO baseline achieved by the $m = 16$ rule indicator configuration is also statistically significant.

The two estimation algorithms differ significantly in their estimation time. Given a feature covariance matrix, the spectral algorithm (SVD, which was done with Matlab, and correlation computation steps) for $m = 16$ took 7 minutes, while the EM algorithm took 5 minutes for *each* iteration with this rank.

5.4 Analysis

Figure 3 presents a comparison of the non-terminal span marginals for two sentences in the development set. We visualize these differences through a heat map of the CKY parse chart, where the starting word of the span is on the rows, and the span end index is on the columns. Each cell is shaded to represent the marginal of that particular non-terminal span, with higher likelihoods in blue and lower likelihoods in red.

For the most part, marginals at the leaves (i.e., pre-terminal marginals) tend to score relatively similarly across different setups. Higher up in the chart, the latent SCFG marginals look quite different than the MLE parameters. Most noticeably, spans starting at the beginning of the sentence are much more favored. It is these rules that allow the right translation to be preferred since the MLE chooses not to place the object of the sentence in the subject's span. However, the spectral parameters seem to discriminate between these higher-level rules better than EM, which scores spans starting with the first word uniformly highly. Another interesting point is that the range of likelihoods is much larger in the EM case compared to the MLE and spectral variants. For the second sentence (row), the 1-best hypothesis produced by all systems are the same, but the heat map accentuates the previous observation.

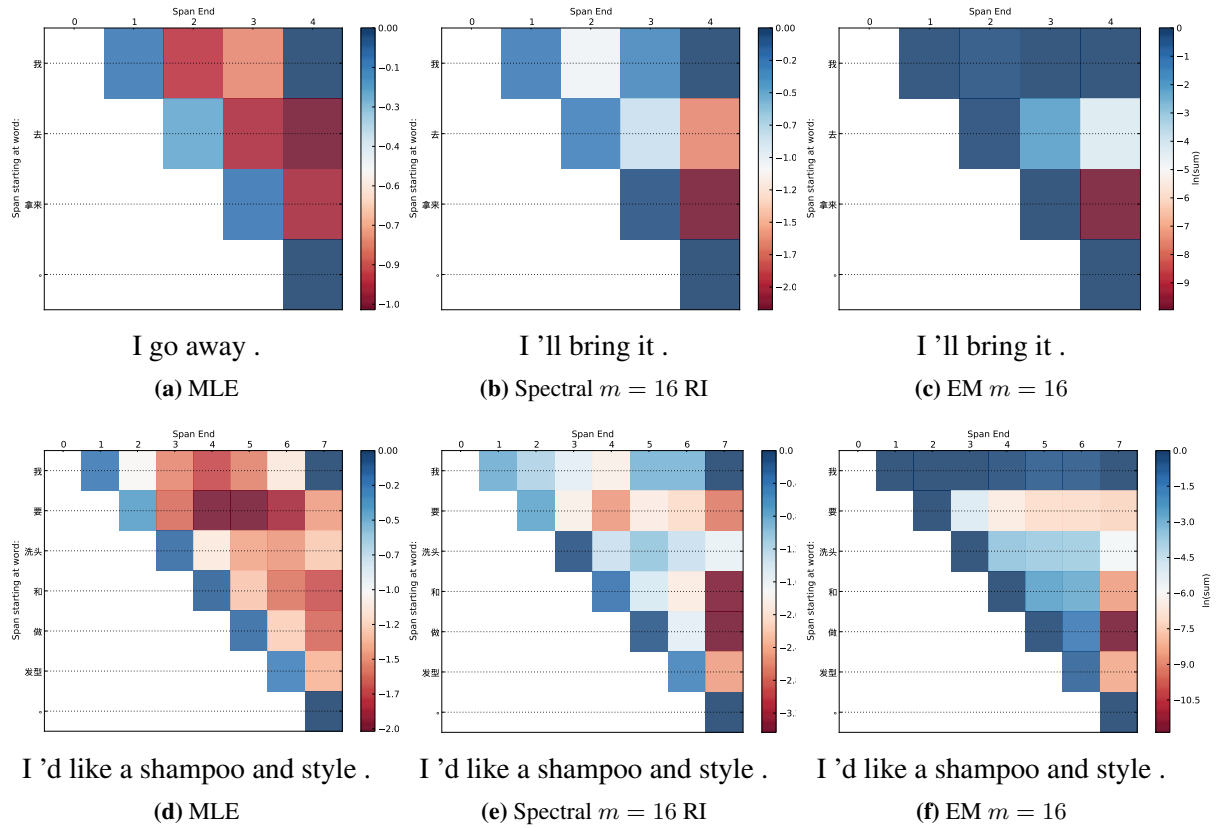
6 Related Work

The goal of refining single-category HPBT grammars or automatically learning the NT categories in a grammar, instead of relying on noisy parser outputs, has been explored from several different angles in the MT literature. Blunsom et al. (2008) present a Bayesian model for synchronous grammar induction, and place an appropriate nonparametric prior on the parameters. However, their starting point is to estimate a synchronous grammar with multiple categories from parallel data (using the word alignments as a prior), while we aim to refine a fixed grammar with additional latent states. Furthermore, their estimation procedure is extremely expensive and is restricted to learning up to five NT categories, via a series of mean-field approximations.

Another approach is to explicitly attach a real-valued vector to each NT: Huang et al. (2010) use an external source-language parser for this purpose and score rules based on the similarity between a source sentence parse and the information contained in this vector, which explicitly requires the integration of a good-quality source-language parser. The EM-based algorithm that we propose here is similar to what they propose, except that we need to handle tensor structures. Mylonakis and Sima'an (2011) select among linguistically motivated non-terminal labels with a cross-validated version of EM. Although they consider a restricted hypothesis space, they do marginalize over different derivations therefore their inside-outside algorithm is $\mathcal{O}(n^6)$. In the syntax-directed translation literature, there have been efforts to relax or coarsen the hard labels provided by a syntactic parser in an automatic manner to promote parameter sharing (Venugopal et al., 2009; Hanneman and Lavie, 2013), which is the complement of our aim in this paper.

The idea of automatically learned grammar refinements comes from the monolingual parsing literature, where phenomena like head lexicalization can be modeled through latent variables. Matsuzaki et al. (2005) look at a likelihood-based method to split the NT categories of a grammar into a fixed number of sub-categories, while Petrov et al. (2006) learn a variable number of sub-categories per NT. The latter's extension may be useful for finding the optimal number of latent states from the data in our case.

The question of whether we can incorporate additional contextual information in minimal rule grammars in MT via auxiliary models instead of using longer, composed rules has been investigated before as well. n-gram translation mod-



(a) MLE **(b)** Spectral $m = 16$ RI **(c)** EM $m = 16$

(d) MLE **(e)** Spectral $m = 16$ RI **(f)** EM $m = 16$

Figure 3: A comparison of the CKY charts containing marginal probabilities of non-terminal spans $\mu(\mathbf{X}, i, j)$ for the MLE, spectral $m = 16$ with rule indicator features, and EM $m = 16$, for the two Chinese sentences. Higher likelihoods are in blue, lower likelihoods in red. The hypotheses produced by each setup are below the heat maps.

els (Mariño et al., 2006; Durrani et al., 2011) seek to model long-distance dependencies and reorderings through n-grams. Similarly, Vaswani et al. (2011) use a Markov model in the context of tree-to-string translation, where the parameters are smoothed with absolute discounting (Ney et al., 1994), while in our instance we capture this smoothing effect through low rank or latent states. Feng and Cohn (2013) also utilize a Markov model for MT, but learn the parameters through a more sophisticated estimation technique that makes use of Pitman-Yor hierarchical priors.

Hsu et al. (2009) presented one of the initial efforts at spectral-based parameter estimation (using SVD) of observed moments for latent-variable models, in the case of Hidden Markov models. This idea was extended to L-PCFGs (Cohen et al., 2014), and our approach can be seen as a bilingual or synchronous generalization.

7 Conclusion

In this work, we presented an approach to refine synchronous grammars used in MT by inferring the latent categories for the single non-terminal in our grammar rules, and proposed two algorithms to estimate parameters for our latent-variable model. By fixing the synchronous derivations of each parallel sentence in the training data, it is possible to avoid many of the computational issues associated with synchronous grammar induction. Improvements over a minimal grammar baseline and equivalent performance to a hierarchical phrase-based baseline are achieved by the spectral approach. For future work, we will seek to relax this consideration and jointly reason about non-terminal categories and derivation structures.

Acknowledgements

The authors would like to thank Daniel Gildea for sharing his code to extract minimal derivation trees, Stefan Riezler for useful discussions, Brendan O'Connor for the CKY visualization advice, and the anonymous reviewers for their feedback. This work was supported by a grant from eBay Inc. (Saluja), the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533 (Dyer).

References

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of NAACL*.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. Bayesian Synchronous Grammar Induction. In *Proceedings of NIPS*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, June.

David Chiang. 2012. Hope and Fear for Discriminative Training of Statistical Translation Models. *Journal of Machine Learning Research*, pages 1159–1187.

Shay B. Cohen and Michael Collins. 2012. Tensor decomposition for fast parsing with latent-variable PCFGs. In *Proceedings of NIPS*.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL*.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2014. Spectral learning of latent-variable PCFGs: Algorithms and sample complexity. *Journal of Machine Learning Research*.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Nadir Durrani, Helmut Schmid, and Alexander Fraser. 2011. A joint sequence translation model with integrated reordering. In *Proceedings of ACL*.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of ACL*.

Yang Feng and Trevor Cohn. 2013. A Markov model of machine translation using non-parametric bayesian inference. In *Proceedings of ACL*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of ACL*.

Jonathan Graehl, Kevin Knight, and Jonathan May. 2008. Training tree transducers. *Computational Linguistics*, 34(3):391–427, September.

Greg Hanneman and Alon Lavie. 2013. Improving syntax-augmented machine translation by coarsening the label set. In *Proceedings of NAACL*.

Daniel Hsu, Sham M. Kakade, and Tong Zhang. 2009. A Spectral Algorithm for Learning Hidden Markov Models. In *Proceedings of COLT*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*.

Zhongqiang Huang, Martin Čmejrek, and Bowen Zhou. 2010. Soft syntactic constraints for hierarchical phrase-based translation using latent syntactic distributions. In *Proceedings of EMNLP*.

Dan Klein and Christopher D. Manning. 2001. Parsing and hypergraphs. In *Proceedings of IWPT*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*.

Abby Levenberg, Chris Dyer, and Phil Blunsom. 2012. A Bayesian model for learning SCFGs with discontiguous rules. In *Proceedings of EMNLP-CoNLL*.

Percy Liang, Slav Petrov, Michael I. Jordan, and Dan Klein. 2007. The infinite PCFG using hierarchical dirichlet processes. In *Proceedings of EMNLP*.

José B. Mariño, Rafael E. Banchs, Josep M. Crego, Adrià de Gispert, Patrik Lambert, José A. R. Fonollosa, and Marta R. Costa-jussà. 2006. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549, December.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of ACL*.

Markos Mylonakis and Khalil Sima'an. 2011. Learning hierarchical translation structure with linguistic annotations. In *Proceedings of ACL*.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On Structuring Probabilistic Dependencies in Stochastic Language Modelling. *Computer Speech and Language*, 8:1–38.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Michael Paul. 2009. Overview of the IWSLT 2009 evaluation campaign. In *Proceedings of IWSLT*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of ACL*.

Takeaki Uno and Mutsunori Yagiura. 2000. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309.

Ashish Vaswani, Haitao Mi, Liang Huang, and David Chiang. 2011. Rule Markov models for fast tree-to-string translation. In *Proceedings of ACL*.

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. 2009. Preference grammars: Softening syntactic constraints to improve statistical machine translation. In *Proceedings of NAACL*.

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

Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system. In *In Proceedings LREC*.

Hao Zhang, Daniel Gildea, and David Chiang. 2008. Extracting synchronous grammar rules from word-level alignments in linear time. In *Proceedings of COLING*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, StatMT '06, pages 138–141. Association for Computational Linguistics.

Gender and Power: How Gender and Gender Environment Affect Manifestations of Power

Vinodkumar Prabhakaran
Dept. of Computer Science
Columbia University
New York, NY, USA
vinod@cs.columbia.edu

Emily E. Reid
Dept. of Computer Science
Columbia University
New York, NY, USA
eer2137@columbia.edu

Owen Rambow
CCLS
Columbia University
New York, NY, USA
rambow@ccls.columbia.edu

Abstract

We investigate the interaction of power, gender, and language use in the Enron email corpus. We present a freely available extension to the Enron corpus, with the gender of senders of 87% messages reliably identified. Using this data, we test two specific hypotheses drawn from the sociolinguistic literature pertaining to gender and power: women managers use face-saving communicative strategies, and women use language more explicitly than men to create and maintain social relations. We introduce the notion of "gender environment" to the computational study of written conversations; we interpret this notion as the gender makeup of an email thread, and show that some manifestations of power differ significantly between gender environments. Finally, we show the utility of gender information in the problem of automatically predicting the direction of power between pairs of participants in email interactions.

1 Introduction

It has long been observed that men and women communicate differently in different contexts. This phenomenon has been studied by sociolinguists, who typically rely on case studies or surveys. The availability of large corpora of naturally occurring social interactions has given us the opportunity to study language use at a broader level than before. In this paper, we use the Enron Corpus of work-related emails to examine written communication in a corporate setting. We investigate three factors that affect choices in communication: the writer's gender, the gender of his or her fellow discourse participants (what we call the "gender environment"), and the relations of organizational power he or she has to the discourse participants. We concentrate on modeling the writer's choices related to discourse structure, rather than lexical choice. Specifically, our goal is to show that gender, gender environment, and power all affect individuals' choices in complex ways, resulting in patterns in the discourse that reveal the underlying factors.

This paper makes three major contributions. First, we introduce an extension to the well-known Enron corpus of emails: we semi-automatically identify the sender's gender of 87% of email messages in the corpus. This extension will be made publicly available. Second, we use this enriched version of the corpus to investigate the interaction of hierarchical power and gender. We formalize the notion of "gender environment", which reflects the gender makeup of the discourse participants of a particular conversation. We study how gender, power, and gender environment influence discourse participants' choices in dialog. We investigate two specific hypotheses from the sociolinguistic literature, relating to face-saving use of language, and to the use of language to strengthen social relations. This contribution does not exhaust the possibilities of our corpus, but it shows how social science can benefit from advanced natural language processing techniques in analyzing corpora, allowing social scientists to tackle corpora such as the Enron corpus which cannot be examined in its entirety by hand. Third, we show that the gender information in the enriched corpus can be useful for computational tasks, specifically for training a system that predicts the direction of hierarchical power between participants in an interaction. Our use of the gender-based features boosts the accuracy of predicting the direction of power between pairs of email interactants from 68.9% to 70.2% on an unseen test set.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1965–1976,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

The paper is structured as follows. We review related work in Section 2. We present the Gender Identified Enron Corpus (our first contribution) in Section 3. Section 4 defines the problem of predicting power and the various dimensions of interaction we analyze. We turn to our second contribution, the analysis of the data, in Sections 5 and 6. Section 7 describes our third contribution, the machine learning experiments using gender-related features in the prediction of hierarchical power. We then conclude and discuss future work.

2 Related Work

There is much sociolinguistic background related to gender and language use, some of it specifically related to language use in the work environment (Kendall and Tannen, 1997; Holmes and Stubbe, 2003; Kendall, 2003; Herring, 2008). We do not provide a full discussion of this work for lack of space, but single out one paper which has particularly influenced our work. Holmes and Stubbe (2003) provide two case studies that do not look at the differences between male and female managers' communication, but at the difference between female managers' communication in more heavily female vs. more heavily male environments. They find that, while female managers tend to break many stereotypes of "feminine" communication, they have different strategies in connecting with employees and exhibiting power in the two gender environments. This work has inspired us to look at this phenomenon by including "Gender Environment" in our study. By finding the ratios of males to females on a thread, we can look at whether indicators change within a more heavily male or female thread. This notion of gender environment is supported by an idea in recent Twitter-based sociolinguistic research on gender identity and lexical variation (Bamman et al., 2014). One of the many insights from their work is that gendered linguistic behavior is oriented by a number of factors, one of which includes the speaker's audience. Their work looks at Twitter users whose linguistic style fails to identify their gender in classification experiments, and finds that the linguistic gender norms can be influenced by the style of their interlocutors.

Within the NLP community, there has been substantial research exploring language use and power. A large number of these studies are performed in the domain of organizational email where the notion of power is well defined in terms of organizational hierarchy. It is also aided by the availability of the moderately large Enron email corpus which captures email interactions in an organizational setting. Earlier approaches used simple lexical features alone (e.g. (Bramsen et al., 2011; Gilbert, 2012)) as a means to predict power. Later studies have used more complex linguistic and structural features, such as formality (Peterson et al., 2011), dialog acts (Prabhakaran and Rambow, 2013), and thread structure (Prabhakaran and Rambow, 2014). Our work is also on the Enron email corpus, and our baseline features are derived from some of this prior work. Researchers have also studied power and influence in other genres of interactions, such as online forums (Danescu-Niculescu-Mizil et al., 2012; Biran et al., 2012), multi-party chats (Strzalkowski et al., 2012) and off-line interactions such as presidential debates (Nguyen et al., 2013; Prabhakaran et al., 2013; Prabhakaran et al., 2014).

There is also some work within the NLP field on analyzing language use in relation to gender. Mohammad and Yang (2011) analyzed the way gender affects the expression of sentiments in text, while we are interested in how gender relates to manifestations of organizational power. For their study, they assigned gender for the core employees in the Enron email corpus based on whether the first name of the person was easily gender identifiable or not. If the person had an unfamiliar name or a name that could be of either gender, they marked his/her gender as *unknown* and excluded them from their study.[1] For example, the gender of the employee Kay Mann was marked as *unknown* in their gender assignment. However, in our work, we manually research and determine the gender of every core employee.

Researchers have also attempted to automatically predict the gender of email senders using supervised learning techniques based on linguistic features (Corney et al., 2002; Cheng et al., 2011; Deitrick et al., 2012), a task we do not address in this paper. These studies use datasets that are relatively smaller in size. Corney et al. (2002) use around 4K emails from 325 gender identified authors. Cheng et al. (2011) use around 9K emails from 108 gender identified authors. Deitrick et al. (2012) use around 18K emails from 144 gender

[1] http://www.saifmohammad.com/WebDocs/dir-email-gender.txt

identified authors. The dataset we offer is much larger in size, with around 97K emails whose authors are gender identified. We believe that our resource will aid further research in this area.

3 Gender Identified Enron Corpus

3.1 Enron Corpus

In our work, we use the version of Enron email corpus released by Yeh and Harnly (2006). The corpus contains emails from the mailboxes of 145 core employees who held top managerial positions within Enron at the time of bankruptcy. Yeh and Harnly (2006) preprocessed the corpus to combine multiple email addresses belonging to the same entity and identify each entity in the corpus with a unique identifier. The corpus contains a total of 111,933 messages. This version of the corpus has been enriched later by Agarwal et al. (2012) with gold organizational power relations, manually determined using information from Enron organizational charts. It includes relations of 1,518 employees and captures dominance relations between 13,724 pairs of them. This information enables us to study the manifestations of power in these interactions, in relation to gender.

In this version of the corpus, the thread structure of email messages is reconstructed, with the missing messages restored from other emails in which they were quoted. This allows us to go beyond isolated messages and study the dialog structure within email threads. There were 34,156 unique discourse participants across all the email threads present in the corpus. Manually determining the gender of all the discourse participants in the corpus is not feasible. Hence, we adopt a two-step approach through which we reliably identify the gender of a large majority of entities in the email threads within the corpus. We manually determine the gender of the 145 core employees who have a bigger representation in the corpus, and we systemically determine the gender of the rest of the discourse participants using the Social Security Administration's baby names database. We adopt a conservative approach so that we assign a gender only when the name of the participant meets a very low ambiguity threshold.

3.2 Manual Gender Assignment

We researched each of the 145 core employees using web search and found public records about them or articles referring to them. In order to make sure that the results are about the same person we want, we added the word 'enron' to the search queries. Within the public records returned for each core employee, we looked for instances in which they were being referred to either using a gender revealing pronoun (*he*/*him*/*his* vs. *she*/*her*) or using a gender revealing addressing form (*Mr.* vs. *Mrs.*/*Ms.*/*Miss*). Since these employees held top managerial positions within Enron at the time of bankruptcy, it was fairly easy to find public records or articles referring to them. For example, the page we found for Kay Mann clearly identifies her gender.[2] We were able to correctly determine the gender of each of the 145 core employees in this manner. A benefit of manually determining the gender of these core employees is that it ensures a high coverage of 100% confident gender assignments in the corpus.

3.3 Automatic Gender Assignment

As mentioned in Section 3.1, our corpus contains a large number of discourse participants in addition to the 145 core employees for which we manually identified the gender. To attempt to find the gender of these other discourse participants, we first determine their first names and then find how ambiguous the names are by querying the Social Security Administration's (SSA) baby names dataset. We first describe how we calculate an ambiguity score for a name using the SSA dataset and then describe how we use it to determine the gender of discourse participants in our corpus.

3.3.1 SSA Names and Gender Dataset

The US Social Security Administration maintains a dataset of baby names, gender, and name count for each year starting with the 1880s, for names with at least five counts.[3] We used this dataset in order to determine the gender ambiguity of a name. The Enron data set contains emails from 1998 to 2001. We estimate the common age range for a large, corporate firm like Enron at 24-67,[4] so we used the SSA data from 1931-1977 to calculate ambiguity scores for our purposes.

For each name n in the database, let $mp(n)$ and $fp(n)$ denote the percentages of males and females with the name n. Then, we calculate the ambiguity score $AS(n)$ as $100 - |mp(n) - fp(n)|$.

[2] http://www.prnewswire.com/news-releases/kay-mann-joins-noble-as-general-counsel-57073687.html

[3] http://www.ssa.gov/oact/babynames/limits.html

[4] http://www.bls.gov/cps/demographics.htm

The value of $AS(n)$ varies between 0 and 100. A name that is 'perfectly unambiguous' would have an ambiguity score of 0, while a 'perfectly ambiguous' name (i.e., 50%/50% split between genders) would have an ambiguity score of 100. We assign the likely gender of the name to be the one with the higher percentage, if the ambiguity score is below a threshold AS_T.

$$G(n) = \begin{cases} M, & \text{if } AS(n) \leq AS_T \text{ and } mp(n) > fp(n) \\ F, & \text{if } AS(n) \leq AS_T \text{ and } mp(n) \leq fp(n) \\ I, & \text{if } AS(n) > AS_T \end{cases}$$

Around 88% of the names in the SSA dataset have $AS(n) = 0$. We choose a very conservative threshold of $AS_T = 10$ for our gender assignments, which assigns gender to around 93% names in the SSA dataset.[5]

3.3.2 Identifying the First Name

Each discourse participant in our corpus has at least one email address and zero or more names associated with it. The name field is automatically assembled by Yeh and Harnly (2006), where they captured the different names from email headers, which are populated from individual email clients and do not follow a standard format. Not all discourse participants are human; some may refer to organizational groups (e.g., HR Department) or anonymous corporate email accounts (e.g., a webmaster account, do-not-reply address etc.). The name field may sometimes be empty, contain multiple names, contain an email address, or show other irregularities. Hence, it is nontrivial to determine the first name of our discourse participants. We used the heuristics below to extract the most likely first name for each discourse participant.

- If the name field contains two words, pick the second or first word, depending on whether a comma separates them or not.
- If the name field contains three words and a comma, choose the second and third words (a likely first and middle name, respectively). If the name field contains three words but no comma, choose the first and second words (again, a likely first and middle name).
- If the name field contains an email address, pick the portion from the beginning of the string to a '.','_' or '-'; if the email address is in camel case, take portion from the beginning of the string to the first upper case letter.
- If the name field is empty, apply the above rule to the email address field to pick a name.

The above heuristics create a list of candidate names for each discourse participant which we then query for an ambiguity score (Section 3.3.1) and the likely gender. We find the candidate name with the lowest ambiguity score that passes the threshold and assign the associated gender to the discourse participant. If none of the candidate names for a discourse participant passes the threshold, we assign the gender to be 'I' (Indeterminate). We also assign the gender to be 'I', if none of the candidate names is present in the SSA dataset. This will occur if the name is a first name that is not in the database (an unusual or international name; e.g., *Vladi*), or if no true first name was found (e.g., the name field was empty and the email address was only a pseudonym). This will also include most of the cases where the discourse participant is not a human.

3.3.3 Coverage and Accuracy

We evaluated the coverage and accuracy of our gender assignment system on the manually assigned gender data of the 145 core people. We obtained a coverage of 90.3%, i.e., for 14 of the 145 core people, the ambiguity score was higher than the threshold. Of the 131 people the system assigned a gender to, we obtained an accuracy of 89.3% in correctly identifying the gender. We investigated the errors and found that all errors were caused due to incorrectly identifying the first name. These errors arise because the name fields are automatically populated and sometimes the core discourse participants' name fields include their secretaries. While this is common for people in higher managerial positions, we expect this not to happen in the middle management and below, to which most of the automatically gender-assigned discourse participants belong.

3.4 Corpus Statistics and Divisions

We apply the gender assignment system described above to all discourse participants of all email threads in the entire Enron corpus described in Section 3.1. Table 1 shows the coverage of gender assignment in our corpus at different levels: unique discourse participants, messages and threads. In Table 2, we show the male/female percentage split of all unique discourse participants, as well as the split at the level of messages (i.e., messages sent by males vs. females).

[5] In the corpus that will be released, we retain the $AS(n)$ of each name, so that the users of this resource can decide the threshold that suit their needs.

	Count (%)
Total unique discourse participants	34,156
- gender identified	23,009 (67.3%)
Total messages	111,933
- senders gender identified	97,255 (86.9%)
Total threads	36,615
- all senders gender identified	26,015 (71.1%)
- all participants gender identified	18,030 (49.2%)

Table 1: Coverage of Gender Identification at various level: unique discourse participants, messages and threads

	Male	Female
Unique Discourse Participants	66.1%	33.9%
Message Senders	58.2%	41.8%

Table 2: Male/Female split across a) all unique participants who were gender identified, b) all messages whose senders were gender identified

We divide the entire corpus into Train, Dev and Test sets at the thread level, through random sampling, with a distribution of 50%, 25% and 25% each. The number of threads and messages in each subdivision is shown in Table 3.

	Total	**Train**	**Dev**	**Test**
Threads	36,615	18,498	8,973	9,144
Messages	111,933	56,447	27,565	27,921

Table 3: Train/Test/Dev breakup of the entire corpus

We also create a sub-corpus of the threads called *All Participants Gender Identified* (APGI), containing the 18,030 threads for which the gender assignment system succeeded in assigning the genders of all participants, including senders and all recipients (To and CC). For the analysis and experiments presented in the rest of this paper, we use 17,788 threads from this APGI subset, excluding the remaining 242 threads that were used for previous manual annotation efforts.

4 Manifestations of Power

We use the gender information of the participants to investigate how the gender of the sender and recipients affect the manifestations of hierarchical power in interactions. In order to do this, we use the interaction analysis framework from our prior work (Prabhakaran and Rambow, 2014). In this section, we give a brief overview of the problem formulation and the structural features we used.

4.1 Hierarchically Related Interacting Pairs

Let t denote an email thread and M_t denote the set of all messages in t. Also, let P_t be the set of all participants in t, i.e., the union of senders and recipients (*To* and *CC*) of all messages in M_t. We are interested in analyzing the power relations between pairs of participants who interact within a given email thread. Not every pair of participants $(p_1, p_2) \in P_t \times P_t$ interact with one another within t. Let $IM_t(p_1, p_2)$ denote the set of *Interaction Messages* — non-empty messages in t in which either p_1 is the sender and p_2 is one of the recipients or vice versa. We call the set of (p_1, p_2) such that $|IM_t(p_1, p_2)| > 0$ the *interacting participant pairs* of t (IPP_t). For every $(p_1, p_2) \in IPP_t$, we query the set of dominance relations in the gold hierarchy and assign their hierarchical power relation ($HP(p_1, p_2)$) to be *superior* if p_1 dominates p_2, and *subordinate* if p_2 dominates p_1. We exclude pairs that do not exist in the gold hierarchy from our analysis and call the remaining set *related interacting participant pairs* ($RIPP_t$). Table 4 shows the total number of pairs in IPP_t and $RIPP_t$ from all the threads in the APGI subset of our corpus and across Train, Dev and Test sets.

Description	**Total**	**Train**	**Dev**	**Test**
# of threads	17,788	8,911	4,328	4,549
$\sum_t \lvert IPP_t \rvert$	74,523	36,528	18,540	19,455
$\sum_t \lvert RIPP_t \rvert$	4,649	2,260	1,080	1,309

Table 4: Data Statistics
Row 1 presents the total number of threads in different subsets of the corpus. Row 2 and 3 present the number of interacting participant pairs (IPP) and related interacting participant pairs ($RIPP$) in those subsets.

4.2 Structural Features

Now, we describe various features that capture the structure of interaction between the pairs of participants in a thread. Each feature f is extracted with respect to a person p over a reference set of messages M (denoted f^p_M). For a pair (p_1, p_2), we extract 4 versions of each feature f: $f^{p_1}_{IM_t(p_1,p_2)}$, $f^{p_2}_{IM_t(p_1,p_2)}$, $f^{p_1}_{M_t}$ and $f^{p_2}_{M_t}$. The first two capture behavior of each person of the pair in interactions between themselves, while the third and fourth capture their overall behavior in the entire thread. We group our features into three categories — THR^{STR}, THR^{META} and DIA. THR^{STR} captures the thread structure in terms of verbosity and

positional features of messages (e.g., how many emails did a person send). THR$^{\text{META}}$ contain email header meta-data based features that capture the thread structure (e.g., how many recipients were there). Both sets of features do not perform any NLP analysis on the the content of the emails. DIA captures the pragmatics of the dialog and requires a deeper analysis of the email content (e.g., did they issue any requests).

THR$^{\text{STR}}$: This feature set includes two kinds of features — positional and verbosity. The positional features are a boolean feature to denote whether p sent the first message (Initiate), and the relative positions of p's first and last messages (FirstMsgPos and LastMsgPos) in M. The verbosity features are p's message count (MsgCount), message ratio (MsgRatio), token count (TokenCount), token ratio (TokenRato) and tokens per message (TokenPerMsg), all calculated over M.

THR$^{\text{META}}$: This feature set includes the average number of recipients (AvgRecipients) and *To* recipients (AvgToRecipients) in emails sent by p, the percentage of emails p received in which he/she was in the *To* list (InToList%), boolean features denoting whether p added or removed people when responding to a message (AddPerson and RemovePerson), average number of replies received per message sent by p (ReplyRate) and average number of replies received from the other person of the pair to messages where he/she was a *To* recipient (ReplyRateWithinPair). ReplyRateWithinPair applies only to $IM_t(p_1, p_2)$.

DIA: We use dialog acts (DA) and overt displays of power (ODP) tags to model the structure of interactions within the message content. We obtain DA and ODP tags using automatic taggers trained on manual annotations. The DA tagger (Omuya et al., 2013) obtained an accuracy of 92%. The ODP tagger (Prabhakaran et al., 2012) obtained an accuracy of 96% and F-measure of 54%. The DA tagger labels each sentence to be one of the 4 dialog acts: Request Action, Request Information, Inform, and Conventional. The ODP Tagger identifies sentences (mostly requests) that express additional constraints on their addressee, beyond those introduced by the dialog act. For example, the sentence "Please come to my office right now" is considered as an ODP, while "It would be great if you could come to my office now" is not, even though both issue the same request. For more details on ODP, we refer the reader to (Prabhakaran et al., 2012). We use 5 features: ReqAction, ReqInform, Inform, Conventional, and ODPCount to capture the number of sentences in messages sent by p that have each of these labels. We also use a feature to capture the number of p's messages with a request that did not get a reply, i.e., dangling request percentage (DanglingReq%), over all messages sent by p.

Feature Name	**Mean**$(f^X_{IM_t})\|X =$			
	F_{sub}	F_{sup}	M_{sub}	M_{sup}
THR$^{\text{META}}$				
AvgRecipients***	4.76	5.74	5.58	4.98
AvgToRecipients***	3.63	4.73	3.84	3.80
InToList%$^{\cdot}$	0.83	0.86	0.84	0.83
ReplyRate***	0.72	0.86	0.70	0.61
AddPerson	0.58	0.66	0.59	0.68
RemovePerson	0.55	0.60	0.54	0.65
THR$^{\text{STR}}$				
Initiate	0.38	0.24	0.39	0.30
FirstMsgPos*	0.18	0.25	0.19	0.22
LastMsgPos**	0.34	0.33	0.34	0.39
MsgCount***	0.92	0.61	0.93	0.91
MsgRatio***	0.33	0.23	0.33	0.32
TokenCount	76.5	41.0	102.0	54.3
TokenRatio	0.38	0.23	0.40	0.27
TokenPerMsg***	90.2	67.9	118.2	53.2
DIA^{PR}				
Conventional	0.55	0.43	0.64	0.56
Inform	3.50	1.96	4.51	2.53
ReqAction**	0.07	0.06	0.05	0.10
ReqInform	0.29	0.21	0.20	0.16
DanglingReq%	0.06	0.12	0.07	0.18
ODPCount***	0.10	0.07	0.09	0.13

Table 5: ANOVA results and group means for Hierarchical Power and Gender
F_{sub}: Female subordinates; F_{sup}: Female superiors; M_{sub}: Male subordinates; M_{sup}: Male superiors; * ($p < .05$); ** ($p < .01$); *** ($p < .001$)

5 Gender and Power

In this subsection, we analyze the impact of gender on the expression of power in email. We perform an ANOVA test on all features described in Section 4.2 keeping both Hierarchical Power and Gender as independent variables. We perform this on the Train subset of the APGI subset of our corpus. Table 5 shows the results for thread level version of the features (we obtain similar significance results at the interaction level as well). As can be seen from the ANOVA results, the mean values of many features differ significantly for the factorial

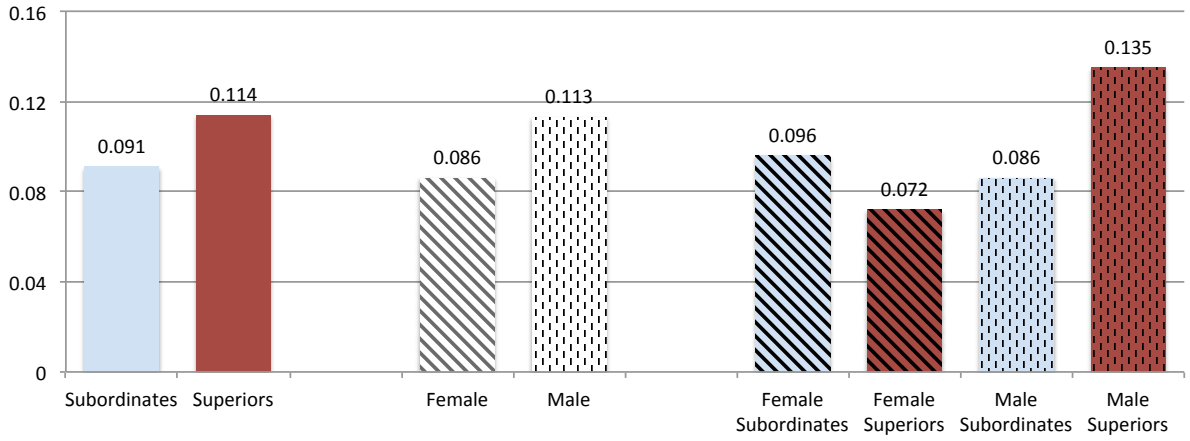


Figure 1: Mean values of ODPCounts in different groups: Subordinates vs. Superiors; Female vs. Male; across all combinations of Hierarchical Power and Gender.

groups of Hierarchical Power and Gender. For example, ReplyRate was highly significant; female superiors obtain the highest reply rate.

It is crucial to note that ANOVA only determines that there is a significant difference between groups, but does not tell which groups are significantly different. In order to ascertain that, we must use the Tukey's HSD (Honest Significant Difference) Test. We do not describe the analysis of all our features to that depth in this paper due to space limitations. Instead, we investigate specific hypotheses which we have derived from sociolinguistic literature. The first hypothesis we investigate is:

- **Hypothesis 1**: Female superiors tend to use "face-saving" strategies at work that include conventionally polite requests and impersonalized directives, and that avoid imperatives (Herring, 2008).

As a stand-in for a face-threatening communicative strategy, we use our "Overt Display of Power" feature (ODP). An ODP limits the addressee's range of possible responses, and thus threatens his or her (negative) face.[6] We thus reformulate our hypothesis as follows: the use of ODP by superiors changes when looking at the splits by gender, with female superiors using fewer ODPs than male superiors. We look further into the ANOVA analysis of the thread-level ODPCount treating Hierarchical Power and Gender as independent variables. Figure 1 shows the mean values of ODP counts in each group of participants. A summary of the results follows.

Hierarchical Power was significant. Subordinates had an average of 0.091 ODP counts and Superiors had an average of 0.114 ODP counts. Gender was also significant; Females had an average of 0.086 ODP counts and Males had an average of 0.113 ODP counts. When looking at the factorial groups of Hierarchical Power and Gender, however, several results were very highly significant. The significantly different pairs of groups, as per the Tukey's HSD test, are Male Superiors/Male Subordinates, Male Superiors/Female Superiors, and Male Superiors/Female Subordinates. Male Superiors used the most ODPs, with an average of 0.135 counts. Somewhat surprisingly, Female Superiors used the *least* of the entire group, with an average of 0.072 counts. Among Subordinates, Females actually used slightly more ODP, with an average of 0.096 counts. Male Subordinates had an average of 0.086 ODP counts. However, the differences among these three groups (Female Superiors, Female Subordinates, and Male Subordinates) are not significant.

The results confirm our hypothesis: female superiors use fewer ODPs than male superiors. However, we also see that among women, there is no significant difference between superiors and subordinates, and the difference between superiors and subordinates in general (which is significant) is entirely due to men. This in fact shows that a more specific (and more interesting) hypothesis than our original hypothesis is validated: only male superiors use more ODPs than subordinates.

[6]For a discussion of the notion of "face", see (Brown and Levinson, 1987).

6 Gender Environment and Power

We now turn to gender environments and their relation to the expression of power in written dialogs. We again start with a hypothesis based on the sociolinguistic literature.

- **Hypothesis 2**: Women use language to create and maintain social relations, for example, they use more small talk (based on a reported "stereotype" in (Holmes and Stubbe, 2003)).

We first define more formally what we mean by "gender environment" (Section 6.1), and then investigate our hypothesis (Section 6.2).

6.1 The Notion of "Gender Environment"

The notion of "gender environment" refers to the gender composition of a group who are communicating. In the sociolinguistic studies we have consulted (Holmes and Stubbe, 2003; Herring, 2008), the notion refers to a stable work group who interact regularly. Since we are interested in studying email conversations (threads), we adapt the notion to refer to a single thread at a time. Furthermore, we assume that a discourse participant makes communicative decisions based on (among other factors) his or her own gender, and based on the genders of the people he or she is communicating with in a given conversation (i.e., email thread). We therefore consider the "gender environment" to be specific to each discourse participant and to describe the other participants from his or her point of view. Put differently, we use the notion of "gender environment" to model a discourse participant's (potential) audience in a conversation. For example, a conversation among five women and one man looks like an all-female audience from the man's point of view, but a majority-female audience from the women's points of view.

We define the gender environment of a discourse participant p in a thread t as follows. As discussed, we assume that the gender environment is a property of each discourse participant p in thread t. We take the set of all discourse participants of the thread t, P_t (see Section 4.1), and exclude p from it: $P_t \setminus \{p\}$. We then calculate the percentage of women in this set.[7] We obtain three groups by setting thresholds on these percentages. Finer-grained gender environments resulted in partitions of the data with very few instances, since most of our data involves fairly balanced gender ratios. The three gender environments we use are the following:

- **Female Environment**: if the percentage of women in $P_t \setminus \{p\}$ is above 66.7%.
- **Mixed Environment**: if the percentage of women in $P_t \setminus \{p\}$ is between 33.3% and 66.7%.
- **Male Environment**: if the percentage of women in $P_t \setminus \{p\}$ is below 33.3%

Across all threads and discourse participants in the threads, we have 791 female, 2087 mixed and 1642 male gender environments.

6.2 Gender Environment and Conventional Dialog Acts

We now turn to testing Hypothesis 2. We have at present no way of testing for "small talk" as opposed to work-related talk, so we instead test Hypothesis 2 by asking how many conventional dialog acts a person performs. Conventional dialog acts serve not to convey information or requests (both of which would typically be work-related in the Enron corpus), but to establish communication (greetings) and to manage communication (sign-offs); since communication is an important way of creating and maintaining social relations, we can say that conventional dialog acts serve the purpose of easing conversations and thus of maintaining social relations. Since this aspect of language is specifically dependent on a group of people (it is an inherently social function), we assume that the relevant feature is not simply Gender, but Gender Environment. Specifically, we make our Hypothesis 2 more precise by saying that a higher number of conventional dialog acts is used in Female Environments. We use the thread level version of the feature ConventionalCount.

Figure 2 shows the mean values of ConventionalCount in each sub-group of participants. Hierarchical Power was highly significant as per ANOVA results. Subordinates use conventional language more (0.60 counts) than Superiors (0.52). Gender is a very highly significant variable; Males use 0.60 counts on average, whereas

[7] We note that one could also define the notion of gender environment at the level of individual emails: not all emails in a thread involve the same set of participants. We leave this to future work.

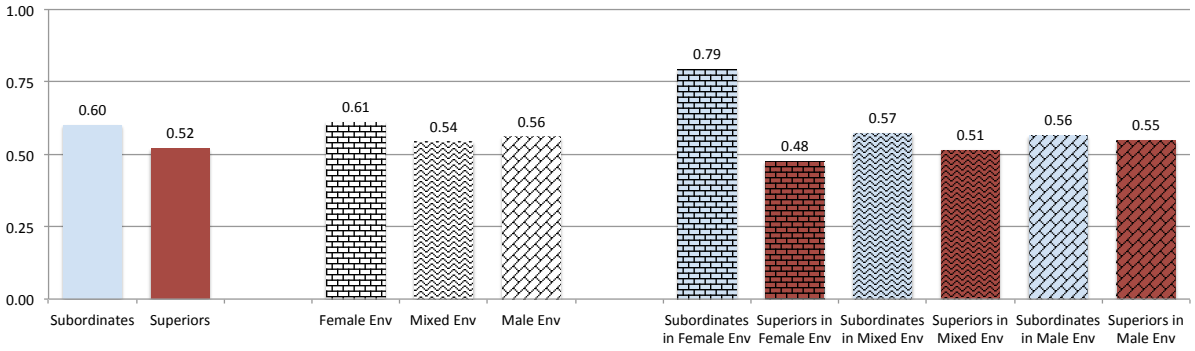


Figure 2: Mean values of Conventional Counts: Subordinates vs. Superiors; across all Gender Environments; across all combinations of Hierarchical Power and Gender Environments.

Females use 0.50. This result is somewhat surprising, but does not invalidate our Hypothesis 2, since our hypothesis is not formulated in terms of Gender, but in terms of Gender Environment. The analysis of Gender Environment at first appears to be a negative result: while the averages by Gender Environment differ, the differences are not significant. However, the groups defined by both Hierarchical Power *and* Gender Environment have highly significant differences. Subordinates in Female Environments use the most conventional language of all six groups, with an average of 0.79. Superiors in Female Environments use the least, with an average of 0.48. Mixed Environments and Male Environments differ, but are more similar to each other than to Female Environments. In fact, in the Tukey HSD test, the only significant pairs are exactly the set of subordinates in Female Environments paired with each other group (Superiors in Female Environments, and Subordinates and Superiors in Mixed Environments and Male Environments). That is, Subordinates in Female environments use significantly more conventional language than any other group, but the remaining groups do not differ significantly from each other.

Our hypothesis is thus only partially verified: while gender environment is a crucial aspect of the use of conventional DAs, we also need to look at the power status of the writer. In fact only subordinates in female environments use more conventional DAs than any other group (as defined by power status and gender environment). While our hypothesis is not fully verified, we interpret the results to mean that subordinates are more comfortable in female environments to use a style of communication which includes more conventional DAs than outside the female environments.

7 Predicting Power in Participant Pairs

In this section, we use the formulation of the power prediction problem presented in our prior work (Prabhakaran and Rambow, 2014). Given a thread t and a pair of participants $(p_1, p_2) \in RIPP_t$, we want to automatically detect $HP(p_1, p_2)$. We use the SVM-based supervised learning system from (Prabhakaran and Rambow, 2014) that can predict $HP(p_1, p_2)$ to be either *superior* or *subordinate* based on the interaction within a thread t for any pair of participants $(p_1, p_2) \in RIPP_t$. The order of participants in (p_1, p_2) is fixed such that p_1 is the sender of the first message in $IM_t(p_1, p_2)$. The power prediction system is built using the ClearTK (Ogren et al., 2008) wrapper for SVMLight (Joachims, 1999) package. It uses a quadratic kernel to capture feature-feature interactions, which is very important as we see in Section 5 and 6. We use the Train, Dev and Test subsets of the APGI subset of our corpus for our experiments. We use the related interacting participant pairs in threads from the Train set to train our models and optimize our performance on those from the Dev set. We report results on both Dev and Test sets.

In addition to the features described in Section 4.2, the power prediction system presented in (Prabhakaran and Rambow, 2014) uses a lexical feature set (LEX) that captures word ngrams, POS (part of speech) ngrams and mixed ngrams, since lexical features have been established to be very useful for power prediction. Mixed ngrams are word ngrams where words belonging to open classes are replaced with their POS tags. We add two gender-based feature sets: GEN containing the gender of both persons of the pair and ENV containing the gender environment feature.

Table 6 presents the results obtained using various feature combinations. We experimented using all subsets of {LEX, THR^{STR}, THR^{META}, DIA, GEN, ENV } on the Dev set; we report the most interesting results here. The majority baseline (*subordinate*) obtains an accuracy of 55.8%. Using the gender-based features alone performs only slightly better than the majority baseline. We use the best performing feature subset from (Prabhakaran and Rambow, 2014) (LEX + THR^{META}) as another baseline, which obtains an accuracy of 68.2%. Adding the GEN features improves the performance to 70.6%. Further adding the ENV features improves the performance, but only marginally to 70.7% (our overall best result, an improvment of 2.4% points). The best performing feature set without using LEX was the combination of DIA, THR^{META} and GEN (67.3%). Removing the gender features from this reduced the performance to 64.6%. Similarly, the best performing feature set which do not use the content of emails at all was THR^{STR} + THR^{META} + GEN (66.6). Removing the gender features decreases the accuracy by a larger margin (5.4% accuracy reduction to 63.0).

We interpret the differences in absolute improvement as follows: the gender-based features on their own are not very useful, and gain predictive value only when paired with other features. This is because the other features in fact make quite different predictions depending on gender and/or gender environment. However, the content features (and in particular the lexical features) are so powerful on their own that the relative contribution of the gender-based features decreases again. Nonetheless, we take these results as validation of the claim that gender-based features enhance the value of other features in the task of predicting power relations.

We performed another experiment where we partitioned the data into two subsets according to the gender of the first person of the pair and trained two separate models to predict power. At test time, we chose the appropriate model based on the gender of the first person of the pair. However, this did not improve the performance.

On our blind test set, the majority baseline obtains an accuracy of 57.9% and the (Prabhakaran and Rambow, 2014) baseline obtains an accuracy of 68.9%. On adding the gender-based features, the accuracy of the system improves to 70.2%.

Description	**Accuracy**
Majority (Always Subordinate)	55.83
GEN	57.59
GEN + ENV	57.59
Baseline (LEX + THR^{META})	68.24
Baseline (LEX + THR^{META}) + GEN	70.56
Baseline (LEX + THR^{META}) + GEN + ENV	**70.74**
DIA + THR^{META} + GEN	67.31
DIA + THR^{META}	64.63
THR^{STR} + THR^{META} + GEN	66.57
THR^{STR} + THR^{META}	62.96

Table 6: Accuracies on feature subsets (Dev set). THR^{META}: meta-data; THR^{STR}: structural; DIA: dialog-act; GEN: gender; ENV: gender environment; LEX: ngrams;

8 Conclusion

We presented a new, freely available resource: the Gender Identified Enron Corpus, and explored the relation between power, gender, and language using this resource. We also introduced the notion of gender environment, and showed that the manifestations of power differ significantly between gender environments. We also showed that the gender-related features helps in improving power prediction. In future work, we will explore machine learning algorithms which capture the interactions between features better than our SVM with quadratic kernel.

We expect our corpus to be a rich resource for social scientists interested in the effect of power and gender on language use. We will investigate several other sociolinguistic-inspired research questions; for example, do the strategies managers use for "effectiveness" of communication differ based on gender environments?

While our findings pertain to the Enron data set, we believe that the insights and techniques from this study can be extended to other genres in which there is an independent notion of hierarchical power, such as moderated online forums.

Acknowledgments

This paper is based upon work supported by the DARPA DEFT Program. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. We thank several anonymous reviewers for their constructive feedback.

References

Apoorv Agarwal, Adinoyi Omuya, Aaron Harnly, and Owen Rambow. 2012. A comprehensive gold standard for the enron organizational hierarchy. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–165, Jeju Island, Korea, July. Association for Computational Linguistics.

David Bamman, Jacob Eisenstein, and Tyler Schnoebelen. 2014. Gender identity and lexical variation in social media. *Journal of Sociolinguistics*, 18(2):135–160.

Or Biran, Sara Rosenthal, Jacob Andreas, Kathleen McKeown, and Owen Rambow. 2012. Detecting influencers in written online conversations. In *Proceedings of the Second Workshop on Language in Social Media*, pages 37–45, Montréal, Canada, June. Association for Computational Linguistics.

Philip Bramsen, Martha Escobar-Molano, Ami Patel, and Rafael Alonso. 2011. Extracting social power relationships from natural language. In *ACL*, pages 773–782. The Association for Computational Linguistics.

Penelope Brown and Stephen C. Levinson. 1987. *Politeness : Some Universals in Language Usage (Studies in Interactional Sociolinguistics)*. Cambridge University Press, February.

Na Cheng, R. Chandramouli, and K. P. Subbalakshmi. 2011. Author gender identification from text. *Digit. Investig.*, 8(1):78–88, July.

Malcolm Corney, Olivier de Vel, Alison Anderson, and George Mohay. 2002. Gender-preferential text mining of e-mail discourse. In *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, pages 282–289. IEEE.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: language effects and power differences in social interaction. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, New York, NY, USA. ACM.

William Deitrick, Zachary Miller, Benjamin Valyou, Brian Dickinson, Timothy Munson, and Wei Hu. 2012. Author gender prediction in an email stream using neural networks. *Journal of Intelligent Learning Systems & Applications*, 4(3).

Eric Gilbert. 2012. Phrases that signal workplace hierarchy. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, CSCW '12, pages 1037–1046, New York, NY, USA. ACM.

Susan C Herring. 2008. Gender and power in online communication. *The handbook of language and gender*, page 202.

Janet Holmes and Maria Stubbe. 2003. feminine workplaces: stereotype and reality. *The handbook of language and gender*, pages 572–599.

Thorsten Joachims. 1999. Making Large-Scale SVM Learning Practical. In Bernhard Schölkopf, Christopher J.C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, USA. MIT Press.

Shari Kendall and Deborah Tannen. 1997. Gender and language in the workplace. In *Gender and Discourse*, pages 81–105. Sage, London.

Shari Kendall. 2003. Creating gendered demeanors of authority at work and at home. *The handbook of language and gender*, page 600.

Saif Mohammad and Tony Yang. 2011. Tracking sentiment in mail: How genders differ on emotional axes. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2.011)*, pages 70–79, Portland, Oregon, June. Association for Computational Linguistics.

Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, Deborah A. Cai, Jennifer E. Midberry, and Yuanxin Wang. 2013. Modeling topic control to detect influence in conversations using nonparametric topic models. *Machine Learning*, pages 1–41.

Philip V. Ogren, Philipp G. Wetzler, and Steven Bethard. 2008. ClearTK: A UIMA toolkit for statistical natural language processing. In *Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP workshop at Language Resources and Evaluation Conference (LREC)*.

Adinoyi Omuya, Vinodkumar Prabhakaran, and Owen Rambow. 2013. Improving the quality of minority class identification in dialog act tagging. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 802–807, Atlanta, Georgia, June. Association for Computational Linguistics.

Kelly Peterson, Matt Hohensee, and Fei Xia. 2011. Email formality in the workplace: A case study on the enron corpus. In *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, pages 86–95, Portland, Oregon, June. Association for Computational Linguistics.

Vinodkumar Prabhakaran and Owen Rambow. 2013. Written dialog and social power: Manifestations of different types of power in dialog behavior. In *Proceedings of the IJCNLP*, pages 216–224, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Vinodkumar Prabhakaran and Owen Rambow. 2014. Predicting power relations between participants in written dialog from a single thread. In *Proceedings of the 52nd Annual Meeting of the Association*

for Computational Linguistics (Volume 2: Short Papers), pages 339–344, Baltimore, Maryland, June. Association for Computational Linguistics.

Vinodkumar Prabhakaran, Owen Rambow, and Mona Diab. 2012. Predicting Overt Display of Power in Written Dialogs. In *Human Language Technologies: The 2012 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Montreal, Canada, June. Association for Computational Linguistics.

Vinodkumar Prabhakaran, Ajita John, and Dorée D. Seligmann. 2013. Who had the upper hand? ranking participants of interactions based on their relative power. In *Proceedings of the IJCNLP*, pages 365–373, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Vinodkumar Prabhakaran, Ashima Arora, and Owen Rambow. 2014. Power of confidence: How poll scores impact topic dynamics in political debates. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, page 49, Baltimore, MD, USA, June. Association for Computational Linguistics.

Tomek Strzalkowski, Samira Shaikh, Ting Liu, George Aaron Broadwell, Jenny Stromer-Galley, Sarah Taylor, Umit Boz, Veena Ravishankar, and Xiaoai Ren. 2012. Modeling leadership and influence in multi-party online discourse. In *Proceedings of COLING*, pages 2535–2552, Mumbai, India, December. The COLING 2012 Organizing Committee.

Jen-Yuan Yeh and Aaron Harnly. 2006. Email thread reassembly using similarity matching. In *CEAS 2006 - The Third Conference on Email and Anti-Spam, July 27-28, 2006, Mountain View, California, USA*, Mountain View, California, USA, July.

Online Topic Model for Twitter Considering Dynamics of User Interests and Topic Trends

Kentaro Sasaki, Tomohiro Yoshikawa, Takeshi Furuhashi
Graduate School of Engineering Nagoya University
sasaki@cmplx.cse.nagoya-u.ac.jp
yoshikawa, furuhashi@cse.nagoya-u.ac.jp

Abstract

Latent Dirichlet allocation (LDA) is a topic model that has been applied to various fields, including user profiling and event summarization on Twitter. When LDA is applied to tweet collections, it generally treats all aggregated tweets of a user as a single document. Twitter-LDA, which assumes a single tweet consists of a single topic, has been proposed and has shown that it is superior in topic semantic coherence. However, Twitter-LDA is not capable of online inference. In this study, we extend Twitter-LDA in the following two ways. First, we model the generation process of tweets more accurately by estimating the ratio between topic words and general words for each user. Second, we enable it to estimate the dynamics of user interests and topic trends online based on the topic tracking model (TTM), which models consumer purchase behaviors.

1 Introduction

Microblogs such as Twitter, have prevailed rapidly in our society recently. Twitter users post a message using 140 characters, which is called a tweet. The characters limit allows users to post tweets easily about not only personal interest or real life but also public events such as traffic accidents or earthquakes. There have been many studies on how to extract and utilize such information on tweets (Diao et al., 2012; Pennacchiotti and Popescu, 2011; Sakaki et al., 2010; Weng et al., 2010).

Topic models, such as latent Dirichlet allocation (LDA) (Blei et al., 2003) are widely used to identify latent topic structure in large collections of documents. Recently, some studies have applied LDA to Twitter for user classification (Pennacchiotti and Popescu, 2011), detection of influential users (Weng et al., 2010), and so on. LDA is a generative document model, which assumes that each document is represented as a probability distribution over some topics, and that each word has a latent topic. When we apply LDA to tweets, each tweet is treated as a single document. This direct application does not work well because a tweet is very short compared with traditional media such as newspapers. To deal with the shortness of a tweet, some studies aggregated all the tweets of a user as a single document (Hong and Davison, 2010; Pennacchiotti and Popescu, 2011; Weng et al., 2010). On the other hand, Zhao et al. (2011) proposed "Twitter-LDA," which is a model that considers the shortness of a tweet. Twitter-LDA assumes that a single tweet consists of a single topic, and that tweets consist of topic and background words. Zhao et al. (2011) show that it works well at the point of semantic coherence of topics compared with LDA. However, as with the case of LDA, Twitter-LDA cannot consider a sequence of tweets because it assumes that samples are exchangeable. In Twitter, user interests and topic trends are dynamically changing. In addition, when new data comes along, a new model must be generated again with all the data in Twitter-LDA because it does not assume online inference. Therefore, it cannot efficiently analyze the large number of tweets generated everyday. To overcome these difficulties, a model that considers the time sequence and has the capability of online inference is required.

In this study, we first propose an improved model based on Twitter-LDA, which assumes that the ratio between topic and background words differs for each user. This study evaluates the proposed method based on perplexity and shows the efficacy of the new assumption in the improved model. Second, we propose a new topic model called "Twitter-TTM" by extending the improved

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1977–1985,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

model based on the topic tracking model (TTM) (Iwata et al., 2009), which models the purchase behavior of consumers and is capable of online inference. Finally, we demonstrate that Twitter-TTM can effectively capture the dynamics of user interests and topic trends in Twitter.

2 Improvement of Twitter-LDA

2.1 Improved-Model

Figure 1(a) shows the graphical representation of Twitter-LDA based on the following assumptions. There are K topics in Twitter and each topic is represented by a topic word distribution. Each user has his/her topic interests ϕ_u represented by a distribution over K topics. Topic k is assigned to each tweet of user u depending on the topic interests ϕ_u. Each word in the tweet assigned by topic k is generated from a background word distribution θ_B or a topic word distribution θ_k. Whether the word is a background word or a topic word is determined by a latent value y. When $y = 0$, the word is generated from the background word distribution θ_B, and from the topic word distribution θ_k when $y = 1$. The latent value y is chosen according to a distribution π. In other words, the ratio between background and topic words is determined by π.

In Twitter-LDA, π is common for all users, meaning that the rate between background and topic words is the same for each user. However, this assumption could be incorrect, and the rate could differ for each user. Thus, we develop an improved model based on Twitter-LDA, which assumes that π is different for each user, as shown in Figure 1(b). In the improved model, the rate between background and topic words for user u is determined by a user-specific distribution π_u. The improved model is expected to infer the generative process of tweets more efficiently.

2.2 Experiment for Improved Model

We performed an experiment to compare the predictive performances of LDA, TTM, and the improved model shown in Section 2.1. In this experiment, LDA was applied as the method to aggregate all tweets of a user as a single document. The original Twitter data set contains 14,305 users and 292,105 tweets collected on October 18, 2013. We then removed words that occurred less than 20 times and stop words. Retweets[1] were treated as the same as other general tweets because they reflected the user's interests. After the above preprocessing, we obtained the final dataset with 14,139 users, 252,842 tweets, and 7,763 vocabularies. Each model was inferred with collapsed Gibbs sampling (Griffiths and Steyvers, 2004) and the iteration was set at 500. For a fair comparison, the hyper parameters in these models were optimized in each Gibbs sampling iteration by maximizing likelihood using fixed iterations (Minka, 2000).

This study employs perplexity as the evaluation index, which is the standard metric in information retrieval literature. The perplexity of a held-out test set is defined as

$$perplexity = exp\Big(-\frac{1}{N}\sum_{u}\log p(\mathbf{w_u})\Big) \quad (1)$$

where $\mathbf{w_u}$ represents words are contained in the tweets of user u and N is the number of words in the test set. A lower perplexity means higher predictive performance. We set the number of topics K at 50, 100, 150, 200, and 250 and evaluated the perplexity for each model in each K via a 10-fold cross-validation.

The results are shown in Table 1, which shows that the improved model performs better than the other models for any K. Therefore, the new assumption of the improved model, that the rate between background and topic words is different for each user, could be more appropriate. LDA performance worsens with an increase in K because the aggregated tweets of a single user neglect the topic of each tweet.

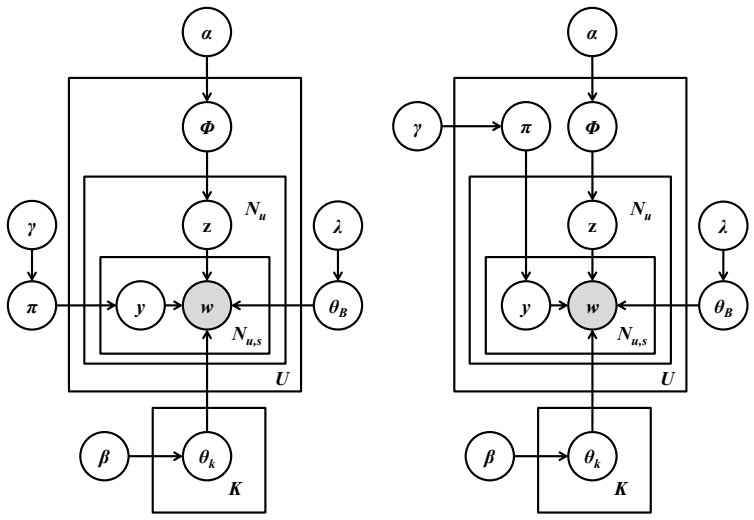
Table 2 shows examples of the tweets of users with high and low rates of background words. The users with a high background words rate tend to use basic words that are often used in any topics, such as "like," "about," and "people," and they tend to tweet about their personal lives. On the other hand, for users with a low background words rate, topical words are often used such as "Arsenal," "Justin," and "Google". They tend to tweet about their interests, including music, sports, and movies.

3 Twitter-TTM

3.1 Model Extension based on Topic Tracking Model

We extend the improved model shown in Section 2.1 considering the time sequence and capabil-

[1] Republishing a tweet written by another Twitter user.



(a) Twitter-LDA (b) Improved-model

Figure 1: Graphical representation of Twitter-LDA and Improved-model

Table 1: Perplexity of each model in 10 runs

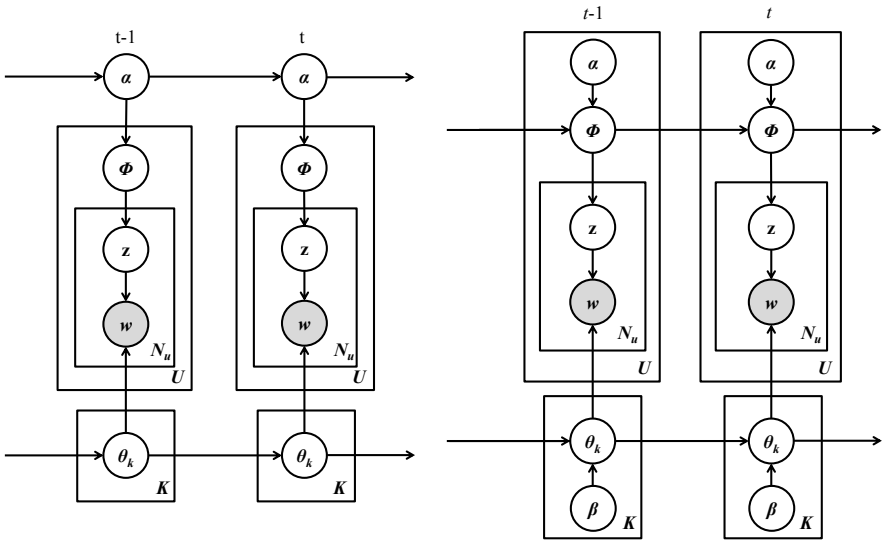
Number of topic K	LDA	Twitter-LDA	Improved-model
50	1586.7 (14.4)	2191.0 (28.4)	**1555.3** (36.7)
100	1612.7 (11.9)	1933.9 (23.6)	**1471.7** (22.3)
150	1635.3 (11.2)	1760.1 (15.7)	**1372.3** (20.0)
200	1655.2 (13.0)	1635.4 (22.1)	**1289.5** (13.3)
250	1672.7 (17.2)	1542.8 (12.5)	**1231.1** (11.9)

Table 2: Example of tweets of users with high and low rate of background words

High rate of background words	Low rate of background words
I hope today goes quickly	Team Arsenal v will Ozil be
I want to work in a cake	Making Justin smile and laugh as he is working on music
All need your support please	Google nexus briefly appears in Google play store

ity of online inference based on TTM (Iwata et al., 2009). TTM is a probabilistic consumer purchase behavior model based on LDA for tracking the interests of each user and the trends in each topic. Other topic models considering the dynamics of topics include the dynamic topic model (DTM) (Blei and Lafferty, 2006) and topic over time (ToT) (Wang and McCallum, 2006). DTM is a model for analyzing the time evolution of topics in time-ordered document collections. It does not track the interests of each user as shown in Figure 2(a) because it assumes that a user (document) has only one time stamp. ToT requires all the data over time for inference, thus, it is not appropriate for application to continuously generated data such as Twitter. We consider a model must be capable of online inference and track the dynamics of user interests and topic trends for modeling tweets. Since TTM has these abilities, we adapt it to the improved model described in Section 2.

Figure 2(b) shows the graphical representation of TTM. TTM assumes that the mean of user interests at the current time is the same as that at the previous time, unless new data is observed. Formally, the current interest $\phi_{t,u}$ are drawn from the following Dirichlet distribution in which the mean is the previous interest $\hat{\phi}_{t-1,u}$ and the precision is



(a) DTM (b) TTM

Figure 2: Graphical representation of DTM and TTM

$\alpha_{t,u}$

$$p(\phi_{t,u}|\hat{\phi}_{t-1,u},\alpha_{t,u}) \propto \prod_k \phi_{t,u,k}^{\alpha_{t,u}\hat{\phi}_{t-1,u,k}-1} \quad (2)$$

where $\phi_{t,u,k}$ represents the probability that user u is interested in topic k at time t. t is a discrete variable and can be arbitrarily set as the unit time interval, e.g., at one day or one week. The precision $\alpha_{t,u}$ represents the interest persistence of how consistently user u maintains his/her interests at time t compared with the previous time $t-1$. $\alpha_{t,u}$ is estimated for each time period and each user because interest persistence depends on both time and users. As mentioned above, the current topic trend $\theta_{t,k}$ is drawn from the following Dirichlet distribution with the previous trend $\hat{\theta}_{t-1,k}$

$$p(\theta_{t,k}|\hat{\theta}_{t-1,k},\beta_{t,k}) \propto \prod_v \theta_{t,k,v}^{\beta_{t,k}\hat{\theta}_{t-1,k,v}-1} \quad (3)$$

where $\theta_{t,k,v}$ represents the probability that word v is chosen in topic k at time t.

Here our proposed Twitter-TTM adapts the above TTM assumptions to the improved model. That is, we extend the improved model whereby user interest $\phi_{t,u}$ and topic trend $\theta_{t,k}$ depend on previous states. Time dependency is not considered on θ_B and π_u because they can be regarded as being independent of time.

Figures 3 and 4 show the generative process and a graphical representation of Twitter-TTM, respectively. Twitter-TTM can capture the dynamics of user interests and topic trends in Twitter considering the features of tweets online. Moreover, Twitter-TTM can be extended to capture long-term dependences, as described in Iwata et al. (2009).

3.2 Model Inference

We use a stochastic expectation-maximization algorithm for Twitter-TTM inference, as described in Wallach (2006) in which Gibbs sampling of latent values and maximum joint likelihood estimation of parameters are alternately iterated. At time t, we estimate user interests $\mathbf{\Phi}_t = \{\hat{\phi}_{t,u}\}_{u=1}^U$, topic trends $\mathbf{\Theta}_t = \{\hat{\theta}_{t,k}\}_{k=1}^K$, background word distribution $\theta_{t,B}$, word usage rate distribution $\pi_{t,u}$, interest persistence parameters $\boldsymbol{\alpha_t} = \{\alpha_{t,u}\}_{u=1}^U$, and trend persistence parameters $\boldsymbol{\beta_t} = \{\beta_{t,k}\}_{k=1}^K$ using the previous time interests $\hat{\mathbf{\Phi}}_{t-1}$ and trends $\hat{\mathbf{\Theta}}_{t-1}$.

We employ collapsed Gibbs sampling to infer the latent variables. Let D_t be a set of tweets and Z_t, Y_t be a set of latent variables z, y at time t. We can integrate the parameters in the joint distribu-

tion as follows:

$$
\begin{aligned}
&p(D_t, Y_t, Z_t | \hat{\boldsymbol{\Phi}}_{t-1}, \hat{\boldsymbol{\Theta}}_{t-1}, \boldsymbol{\alpha_t}, \boldsymbol{\beta_t}, \lambda, \gamma) \\
&= \Big(\frac{\Gamma(2\gamma)}{\Gamma(\gamma)^2}\Big)^U \prod_u \frac{\Gamma(\gamma + n_{t,u,B})\Gamma(\gamma + n_{t,u,K})}{\Gamma(2\gamma + n_{t,u})} \\
&\times \frac{\Gamma(V\lambda)}{\Gamma(\lambda)^V} \frac{\prod_v \Gamma(n_{t,B,v} + \lambda)}{\Gamma(n_{t,B} + V\lambda)} \\
&\times \prod_k \frac{\Gamma(\beta_{t,k})}{\Gamma(n_{t,k} + \beta_{t,k})} \prod_v \frac{\Gamma(n_{t,k,v} + \beta_{t,k}\hat{\theta}_{t-1,k,v})}{\Gamma(\beta_{t,k}\hat{\theta}_{t-1,k,v})} \\
&\times \prod_u \frac{\Gamma(\alpha_{t,u})}{\Gamma(c_{t,u} + \alpha_{t,u})} \prod_k \frac{\Gamma(c_{t,u,k} + \alpha_{t,u}\hat{\phi}_{t-1,u,k})}{\Gamma(\alpha_{t,u}\hat{\phi}_{t-1,u,k})},
\end{aligned} \tag{4}
$$

where $n_{t,u,B}$ and $n_{t,u,K}$ are the number of background and topic words of user u at time t, $n_{t,B,v}$ is the number of times that word v is assigned as a background word at time t, $n_{t,k,v}$ is the number of times that word v is assigned to topic k at time t, $c_{t,u,k}$ is the number of tweets assigned to topic k for user u at time t. In addition, $n_{t,u} = n_{t,u,B} + n_{t,uK}$, $n_{t,B} = \sum_v n_{t,B,v}$, $n_{t,K} = \sum_k n_{t,k} = \sum_k \sum_v n_{t,k,v}$, $n_{t,u} = \sum_k n_{t,u,k}$, and $c_{t,u} = \sum_k c_{t,u,k}$.

Given the assignment of all other latent variables, we derive the following formula calculated from eq.(4) to infer a latent topic,

$$
\begin{aligned}
&p(z_i = k | D_t, Y_t, Z_{t\backslash i}, \hat{\boldsymbol{\Phi}}_{t-1}, \hat{\boldsymbol{\Theta}}_{t-1}, \boldsymbol{\alpha_t}, \boldsymbol{\beta_t}) \\
&\propto \frac{c_{t,u,k\backslash i} + \alpha_{t,u}\hat{\phi}_{t-1,u,k}}{c_{t,u\backslash i} + \alpha_{t,u}} \frac{\Gamma(n_{t,k\backslash i} + \beta_{t,k})}{\Gamma(n_{t,k} + \beta_{t,k})} \\
&\times \prod_v \frac{\Gamma(n_{t,k,v} + \beta_{t,k}\hat{\theta}_{t-1,k,v})}{\Gamma(n_{t,k,v\backslash i} + \beta_{t,k}\hat{\theta}_{t-1,k,v})},
\end{aligned} \tag{5}
$$

where $i = (t, u, s)$, thus z_i represents a topic assigned to the s-th tweet of user u at time t, and $\backslash i$ represents a count excluding the i-th tweet.

Then, when $z_i = k$ is given, we derive the following formula to infer a latent variable y_j,

$$
\begin{aligned}
&p(y_j = 0 | D_t, Y_{t\backslash j}, Z_t, \lambda, \gamma) \\
&\propto \frac{n_{t,B,v\backslash j} + \lambda}{n_{t,B\backslash j} + V\lambda} \frac{n_{t,u,B\backslash j} + \gamma}{n_{t,u\backslash j} + 2\gamma},
\end{aligned} \tag{6}
$$

$$
\begin{aligned}
&p(y_j = 1 | D_t, Y_{t\backslash j}, Z_t, \hat{\boldsymbol{\Theta}}_{t-1}, \boldsymbol{\beta_t}, \gamma) \\
&\propto \frac{n_{t,k,v\backslash j} + \beta_{t,k}\hat{\theta}_{t-1,k,v}}{n_{t,k\backslash j} + \beta_{t,k}} \frac{n_{t,u,K\backslash j} + \gamma}{n_{t,u\backslash j} + 2\gamma},
\end{aligned} \tag{7}
$$

where $j = (t, u, s, n)$, thus y_j represents a latent variable assigned to the n-th word in the s-th tweet of user u at time t, and $\backslash j$ represents a count excluding the j-th word.

The persistence parameters $\boldsymbol{\alpha_t}$ and $\boldsymbol{\beta_t}$ are estimated by maximizing the joint likelihood eq.(4), using a fixed point iteration (Minka, 2000). The update formulas are as follows:

$$
\alpha_{t,u}^{new} = \alpha_{t,u} \frac{\sum_k \hat{\phi}_{t-1,u,k} A_{t,u,k}}{\Psi(c_{t,u} + \alpha_{t,u}) - \Psi(\alpha_{t,u})}, \tag{8}
$$

where $A_{t,u,k} = \Psi(c_{t,u,k} + \alpha_{t,u}\hat{\phi}_{t-1,u,k}) - \Psi(\alpha_{t,u}\hat{\phi}_{t-1,u,k})$, and

$$
\beta_{t,k}^{new} = \beta_{t,k} \frac{\sum_v \hat{\theta}_{t-1,k,v} B_{t,k,v}}{\Psi(n_{t,k} + \beta_{t,k}) - \Psi(\beta_{t,k})}, \tag{9}
$$

where $B_{t,k,v} = \Psi(n_{t,k,v} + \beta_{t,k}\hat{\theta}_{t-1,k,v}) - \Psi(\beta_{t,k}\hat{\theta}_{t-1,k,v})$. We can estimate latent variables Z_t, Y_t, and parameters $\boldsymbol{\alpha_t}$ and $\boldsymbol{\beta_t}$ by iterating Gibbs sampling with eq.(5), eq.(6), and eq.(7) and maximum joint likelihood with eq.(8) and eq.(9). After the iterations, the means of $\phi_{t,u,k}$ and $\theta_{t,k,v}$ are obtained as follows.

$$
\hat{\phi}_{t,u,k} = \frac{c_{t,u,k} + \alpha_{t,u}\hat{\phi}_{t-1,u,k}}{c_{t,u} + \alpha_{t,u}}, \tag{10}
$$

$$
\hat{\theta}_{t,k,v} = \frac{n_{t,k,v} + \beta_{t,k}\hat{\theta}_{t-1,k,v}}{n_{t,k} + \beta_{t,k}}. \tag{11}
$$

These estimates are used as the hyper parameters of the prior distributions at the next time period $t + 1$.

4 Related Work

Recently, topic models for Twitter have been proposed. Diao et al. (2012) proposed a topic model that considers both the temporal information of tweets and user's personal interests. They applied their model to find bursty topics from Twitter. Yan et al. (2013) proposed a biterm topic model (BTM), which assumes that a word-pair is independently drawn from a specific topic. They demonstrated that BTM can effectively capture the topics within short texts such as tweets compared with LDA. Chua and Asur (2013) proposed two topic models considering time order and tweet intervals to extract the tweets summarizing a given event. The models mentioned above do not consider the dynamics of user interests, nor

1. Draw $\theta_{t,B}$ ∼Dirichlet(λ)
2. For each topic $k = 1, ..., K$,
   (a) draw $\theta_{t,k}$ ∼Dirichlet($\beta_{t,k}\hat{\theta}_{t-1,k}$)
3. For each user $u = 1, ..., U$,
   (a) draw $\phi_{t,u}$ ∼Dirichlet($\alpha_{t,u}\hat{\phi}_{t-1,u}$)
   (b) draw $\pi_{t,u}$ ∼Beta(γ)
   (c) for each tweet $s = 1, ..., N_u$
      i. draw $z_{t,u,s}$ ∼Multinomial($\phi_{t,u}$)
      ii. for each word $n = 1, ..., N_{u,s}$
         A. draw $y_{t,u,s,n}$ ∼Bernoulli($\pi_{t,u}$)
         B. draw $w_{t,u,s,n}$ ∼ Multinomial($\theta_{t,B}$) if $y_{t,u,s,n} = 0$ or Multinomial($\theta_{t,z_{t,u,s}}$) if $y_{t,u,s,n} = 1$

Figure 3: Generative process of tweets in Twitter-TTM

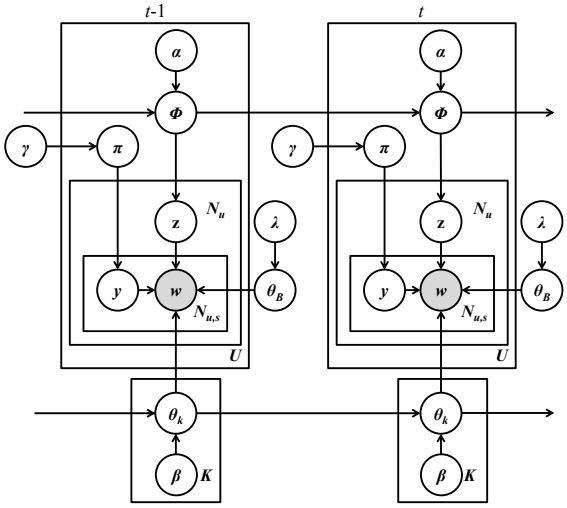


Figure 4: Graphical model of Twitter-TTM

do they have the capability of online inference; thus, they cannot efficiently model the large number of tweets generated everyday, whereas Twitter-TTM can capture the dynamics of user interests and topic trends and has the capability of online inference.

Some online topic models have also been proposed. TM-LDA was proposed by Wang et al. (2012), which can efficiently model online the topics and topic transitions that naturally arise in a tweet stream. Their model learns the transition parameters among topics by minimizing the prediction error on topic distribution in subsequent tweets. However, the TM-LDA does not consider dynamic word distributions. In other words, their model can not capture the dynamics of topic trends. Lau et al. (2012) proposed a topic model implementing a dynamic vocabulary based on online LDA (OLDA) (AlSumait et al., 2008) and applied it to track emerging events on Twitter. An online variational Bayes algorithm for LDA is also proposed (Hoffman et al., 2010). However, these methods are based on LDA and do not consider the shortness of a tweet. Twitter-TTM tackles the shortness of a tweet by assuming that a single tweet consists of a single topic. This assumption is based on the following observation: a tweet is much shorter than a normal document, so a single tweet rarely contains multiple topics but rather a single one.

5 Experiment

5.1 Setting

We evaluated the effectiveness of the proposed Twitter-TTM using an actual Twitter data set. The original Twitter data set contains 15,962 users and 4,146,672 tweets collected from October 18 to 31, 2013. We then removed words that occurred less than 30 times and stop words. After this preprocessing, we obtained the final data set with 15,944 users, 3,679,481 tweets, and 30,096 vocabularies.

We compared the predictive performance of Twitter-TTM with LDA, TTM, Twitter-LDA, Twitter-LDA+TTM, and the improved model based on the perplexity for the next time tweets. Twitter-LDA+TTM is a combination of Twitter-LDA and TTM. It is equivalent to Twitter-TTM, except that the rate between background and topic words is different for each user. We set the number of topics K at 100, the iteration of each model at 500, and the unit time interval at one day. The hyper parameters in these models were optimized in each Gibbs sampling iteration by maximizing likelihood using fixed iterations (Minka, 2000). The inferences of LDA, Twitter-LDA, and the improved model were made for current time tweets.

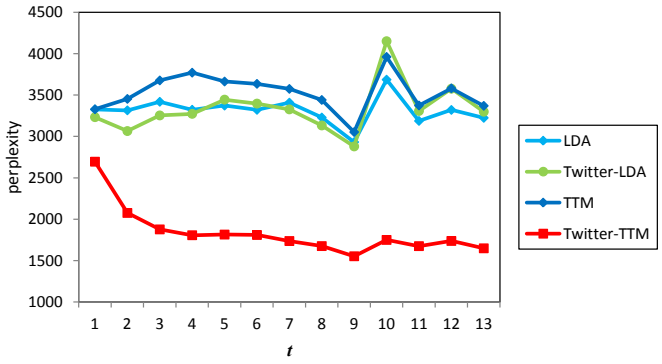
5.2 Result

Figure 5 shows the perplexity of each model for each time, where $t = 1$ in the horizontal axis represents October 18, $t = 2$ represents October 19, ..., and $t = 13$ represents October 31. The perplexity at time t represents the predictive performance of each model inferred by previous time tweets to

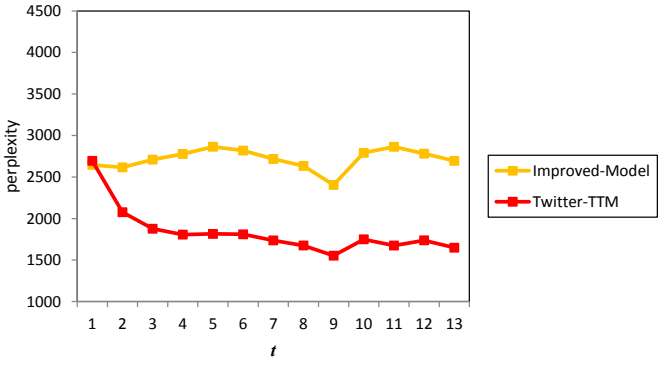
the current time tweets. Note that at $t = 1$, the performance of LDA and TTM, that of Twitter-LDA and Twitter-LDA+TTM, and that of Twitter-TTM and the improved model were found to be equivalent.

As shown in Figure 5(a), the proposed Twitter-TTM shows lower perplexity compared with conventional models, such as LDA, Twitter-LDA, and TTM at any time, which implies that Twitter-TTM can appropriately model the dynamics of user interests and topic trends in Twitter. TTM could not have perplexity lower than LDA although it considers the dynamics. If LDA could not appropriately model the tweets, then the user interests $\hat{\mathbf{\Phi}}_{t-1}$ and topic trends $\hat{\mathbf{\Theta}}_{t-1}$ in the previous time are not estimated well in TTM. Figure 5(b) shows the perplexities of the improved model and Twitter-TTM. From $t = 2$, Twitter-TTM shows lower perplexity than the improved model for each time. The reason for the high perplexity of the improved model is that it does not consider the dynamics. Twitter-TTM also shows lower perplexity than Twitter-LDA+TTM for each time, as shown in Figure 5(c), because Twitter-TTM's assumption that the rate between background and topic words is different for each user is more appropriate, as demonstrated in Section 2.2. These results imply that Twitter-TTM also outperforms other conventional methods, such as DTM, OLDA, and TM-LDA, which do not consider the shortness of a tweet or the dynamics of user interests or topic trends .

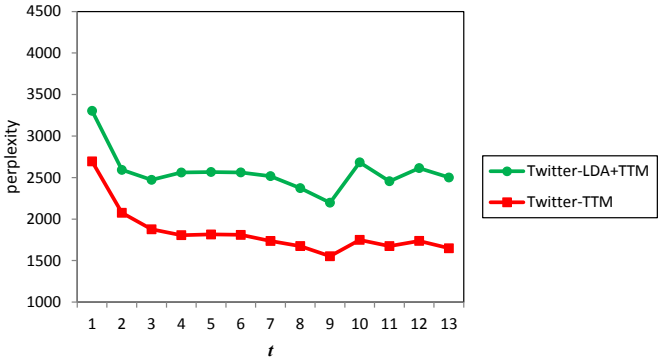
Table 3 shows two topic examples of the topic evolution analyzed by Twitter-TTM, and Figure 6 shows the trend persistence parameters β of each topic at each time. The persistence parameters of the topic "Football" are lower than those of "Birthday" because it is strongly affected by trends in the real world. In fact, the top words in "Football" change more dynamically than those of "Birthday." For example, in the "Football" topic, though 'Arsenal' is usually popular, 'Madrid' becomes more popular on October 24.



(a) Comparison with LDA, Twitter-LDA, and TTM



(b) Comparison with Improved-model



(c) Comparison with Twitter-LDA+TTM

Figure 5: Perplexity for each time

6 Conclusion

We first proposed an improved model based on Twitter-LDA, which estimates the rate between background and topic words for each user. We demonstrated that the improved model could model tweets more efficiently than LDA and Twitter-LDA. Next we proposed a novel probabilistic topic model for Twitter, called Twitter-TTM, which can capture the dynamics of user interests and topic trends and is capable of online inference. We evaluated Twitter-TTM using an actual Twitter data set and demonstrated that it could model more accurately tweets than conventional

methods.

The proposed method currently needs to predetermine the number of topics each time, and it is fixed. In future work, we plan to extend the proposed method to capture the birth and death of topics along the timeline with a variable number of topics, such as the model proposed by Ahmed (Ahmed and Xing, 2010). We also plan to apply the proposed method to content recommendations and trend analysis in Twitter to investigate this method further.

References

Amr Ahmed and Eric P. Xing. 2010. Timeline: A dynamic hierarchical Dirichlet process model for recovering birth/death and evolution of topics in text stream. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI)*, 20–29.

Loulwah AlSumait, Daniel Barbará and Carlotta Domeniconi. 2008. On-line LDA: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 3-12.

David M. Blei., and John D. Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine learning (ICML)*, 113-120.

David M. Blei, Andrew Y. Ng and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3: 993-1022.

Freddy C. T. Chua and Sitaram Asur. 2013. Automatic summarization of events from social media. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM)*.

Qiming Diao, Jing Jiang, Feida Zhu and Ee-Peng Lim 2012. Finding bursty topics from microblogs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, 536–544.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. In *Proceedings of the National Academy of Sciences of the United States of America*, 101(1):5228-5235.

Matthew D. Hoffman, Francis Bach and David M. Blei. 2010. Online learning for latent dirichlet allocation. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 856–864.

Liangjie Hong and Brian D. Davison. 2010. Empirical study of topic modeling in twitter. In *Proceedings of the First Workshop on Social Media Analytics (SOMA)*, 80–88.

Tomoharu Iwata, Shinji Watanabe, Takeshi Yamada. and Naonori Ueda. 2009. Topic tracking model for analyzing consumer purchase behavior. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*,1427–1432.

JeyHan Lau, Nigel Collier and Timothy Baldwin. 2012. On-line trend analysis with topic models:# twitter trends detection topic model online. In *Proceedings of the 23th International Conference on Computational Linguistics (COLING)*, 1519–1534.

Thomas P. Minka 2000. Estimating a Dirichlet distribution *Technical report, MIT*.

Marco Pennacchiotti and Ana-Maria Popescu. 2011. A machine learning approach to Twitter user classification. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM)*, 281–288.

Takeshi Sakaki, Makoto Okazaki and Yutaka Matsuo. 2010. Earthquake shakes Twitter users: realtime event detection by social sensors. In *Proceedings of the World Wide Web Conference (WWW)*, 851–860.

Hanna M. Wallach 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 977–984.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, 424–433.

Yu Wang, Eugene Agichtein and Michele Benz. 2012. TM-LDA: efficient online modeling of the latent topic transitions in social media. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, 123–131.

Jianshu Weng, Ee Peng Lim, Jing Jiang and Qi He. 2010. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM)*, 261–270.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan and Xueqi Cheng 2013. A biterm topic model for short texts. In *Proceedings of the World Wide Web Conference (WWW)*, 1445–1456.

Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan and Xiaoming Li. 2011. Comparing twitter and traditional media using topic models. In *Advances in Information Retrieval*, 338–349.

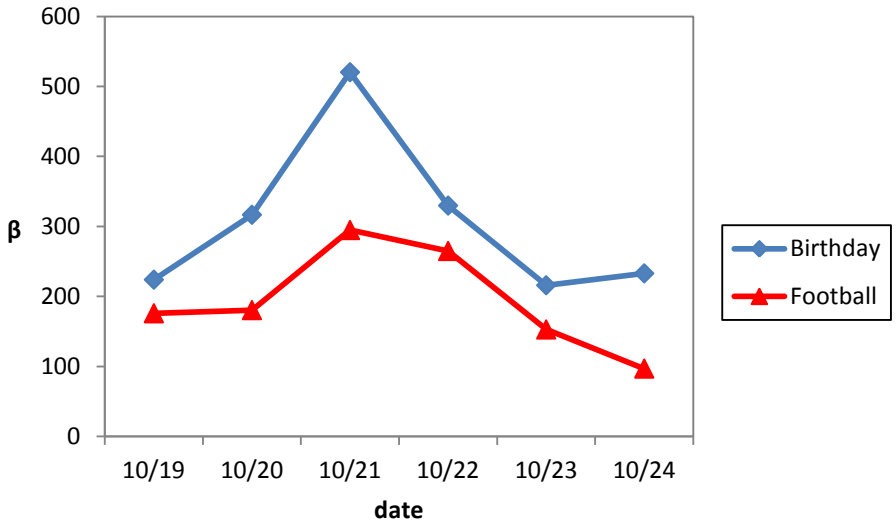


Figure 6: Trend persistence parameters β of each topic at each time estimated by Twitter-TTM

Table 3: Two examples of topic evolution analyzed by Twitter-TTM

Label	Date	Top words
Birthday	10/18	birthday,happy,maria,hope,good,love,thanks,bday,lovely,enjoy
	10/19	happy,birthday,good,hope,thank,enjoy,love,bday,lovely,great
	10/20	birthday,happy,hope,good,love,lovely,great,enjoy,thank,beautiful
	10/21	birthday,happy,hope,good,beautiful,love,lovely,bday,great,thank
	10/22	birthday,happy,hope,good,beautiful,love,bless,thank,today,bday
	10/23	birthday,happy,thank,good,love,hope,beautiful,enjoy,channing,wish
	10/24	birthday,happy,thank,love,hope,good,beautiful,fresh,thanks,jamz
Football	10/18	arsenal,ozil,game,team,cazorla,league,wenger,play,season,good
	10/19	goal,liverpool,gerrard,arsenal,ozil,league,newcastle,suarez,goals,team
	10/20	arsenal,ozil,goal,ramsey,norwich,goals,league,wilshere,mesut,premier
	10/21	arsenal,goal,goals,league,townsend,spurs,player,season,wenger,ozil
	10/22	arsenal,goal,wenger,ozil,league,arsene,goals,birthday,happy,team
	10/23	arsenal,dortmund,ozil,fans,wilshere,borussia,ramsey,lewandowski,giroud,league
	10/24	madrid,goals,ronaldo,cska,real,league,city,moscow,champions,yaya

Self-disclosure topic model for classifying and analyzing Twitter conversations

JinYeong Bak*
Department of Computer Science
KAIST
Daejeon, South Korea
jy.bak@kaist.ac.kr

Chin-Yew Lin
Microsoft Research
Beijing 100080, P.R. China
cyl@microsoft.com

Alice Oh
Department of Computer Science
KAIST
Daejeon, South Korea
alice.oh@kaist.edu

Abstract

Self-disclosure, the act of revealing oneself to others, is an important social behavior that strengthens interpersonal relationships and increases social support. Although there are many social science studies of self-disclosure, they are based on manual coding of small datasets and questionnaires. We conduct a computational analysis of self-disclosure with a large dataset of naturally-occurring conversations, a semi-supervised machine learning algorithm, and a computational analysis of the effects of self-disclosure on subsequent conversations. We use a longitudinal dataset of 17 million tweets, all of which occurred in conversations that consist of five or more tweets directly replying to the previous tweet, and from dyads with twenty of more conversations each. We develop self-disclosure topic model (SDTM), a variant of latent Dirichlet allocation (LDA) for automatically classifying the level of self-disclosure for each tweet. We take the results of SDTM and analyze the effects of self-disclosure on subsequent conversations. Our model significantly outperforms several comparable methods on classifying the level of self-disclosure, and the analysis of the longitudinal data using SDTM uncovers significant and positive correlation between self-disclosure and conversation frequency and length.

1 Introduction

Self-disclosure is an important and pervasive social behavior. People disclose personal information about themselves to improve and maintain relationships (Jourard, 1971; Joinson and Paine, 2007). A common instance of self-disclosure is the start of a conversation with an exchange of names and additional self-introductions. Another example of self-disclosure, shown in Figure 1c, where the information disclosed about a family member's serious illness, is much more personal than the exchange of names. In this paper, we seek to understand this important social behavior using a large-scale Twitter conversation data, automatically classifying the level of self-disclosure using machine learning and correlating the patterns with conversational behaviors which can serve as proxies for measuring intimacy between two conversational partners.

Twitter conversation data, explained in more detail in section 4.1, enable an extremely large scale study of naturally-occurring self-disclosure behavior, compared to traditional social science studies. One challenge of such large scale study, though, remains in the lack of labeled ground-truth data of self-disclosure level. That is, naturally-occurring Twitter conversations do not come tagged with the level of self-disclosure in each conversation. To overcome that challenge, we propose a semi-supervised machine learning approach using probabilistic topic modeling. Our self-disclosure topic model (SDTM) assumes that self-disclosure behavior can be modeled using a combination of simple linguistic features (e.g., pronouns) with automatically discovered semantic themes (i.e., topics). For instance, an utterance "I am finally through with this disastrous relationship" uses a first-person pronoun and contains a topic about personal relationships.

In comparison with various other models, SDTM shows the highest accuracy, and the resulting conversation frequency and length patterns on self-disclosure are shown different over time. Our contributions to the research community include the following:

*This work was done when JinYeong Bak was a visiting student at Microsoft Research, Beijing, China.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1986–1996,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

- We present key features and prior knowledge for identifying self-disclosure level, and show relevance of it with experiment results (Sec. 2).
- We present a topic model that explicitly includes the level of self-disclosure in a conversation using linguistic features and the latent semantic topics (Sec. 3).
- We collect a large dataset of Twitter conversations over three years and annotate a small subset with self-disclosure level (Sec. 4).
- We compare the classification accuracy of SDTM with other models and show that it performs the best (Sec. 5).
- We correlate the self-disclosure patterns and conversation behaviors to show that there is significant relationship over time (Sec. 6).

A fabio capello is the manager are u sure its someone else whos playing lol
B common you guys England manager is Roy Hodgson
A noooooo we mean the manager before!
B haha!! the manager before Roy was Fabio yes, Roy became Manager in May after Fabio resigned in February
A ohhhhhhhhhhh we learn something new everyday! Haha

(a) A G level Twitter conversation

A Today's my mother's birthday and she was extremely happy when I informed her I'm applying for Phoenix soon. Happy Bi rthday mom! :D
B HAHA, nice! Tell her I said Happy Birthday and give her a ki ss and hug for me! :3
A that is a bit problematic. My mommy is not here lol
B HAHA! I figured that'd be the case. Well I'm off tomorrow so I guess I'll do it myself tomorrow. XP
A lol She gets home around 6 or 6:30

(b) A M level Twitter conversation

A My mom has just been taken to the hospital by ambulance. Please pray for her. Thank you
B Hugs **A**. Glad your mom is doing better.
A thanks, she is in hospital & is very disoriented.
B My dad was like that when he was in the hospital. Talked to ppl who had been dead for years.
A yeah, she did that too.
it is so scary to see her that way....
B Extra hugs sweetie. I am glad it wasn't a stroke.

(c) A H level Twitter conversation

Figure 1: An example of a Twitter conversation (from annotated dataset) with G, M and H level of self-disclosure.

2 Self-Disclosure

In this section, we look at social science literature for definition of the levels of self-disclosure. Using that definition, we devise an approach to automatically identify the levels of self-disclosure in a large corpus of OSN conversations. We discuss three approaches, first, using first-person pronoun features, and second, extracting seed words and phrases from the Twitter conversation corpus, and third, extracting seed words and phrases from an external corpus of anonymously posted secrets, and we demonstrate the efficacy of those approaches with an annotated corpus.

2.1 Self-disclosure (*SD*) level

To analyze self-disclosure, researchers categorize self-disclosure language into three levels: G (general) for no disclosure, M for medium disclosure, and H for high disclosure (Vondracek and Vondracek, 1971; Barak and Gluck-Ofri, 2007). Utterances that contain general (non-sensitive) information about the self or someone close (e.g., a family member) are categorized as M. Examples are personal events, past history, or future plans. Utterances about age, occupation and hobbies are also included. Utterances that contain sensitive information about the self or someone close are categorized as H. Sensitive information includes personal characteristics, problematic behaviors, physical appearance and wishful ideas. Generally, these are thoughts and information that one would keep as secrets to himself. All other utterances, those that do not contain information about the self or someone close are categorized as G. Examples include gossip about celebrities or factual discourse about current events. Figure 1 shows Twitter conversation examples with G, M and H levels from annotated dataset (see Section 4.2 for a detailed description of the annotated dataset).

2.2 G Level of Self-Disclosure

An obvious clue of self-disclosure is the use of first-person pronouns. For example, phrases such as 'I live' or 'My name is' indicate that the utterance contains personal information. In previous research, the simple method of counting first-person pronouns was used to measure the degree of self-disclosure (Joinson, 2001; Barak and Gluck-Ofri, 2007). Consequently, the absence of a first-person pronoun signals that the utterance belongs in the G level of self-disclosure. We verify this pattern with a dataset of Tweets annotated with G, M, and H levels. We divide the annotated Tweets into two classes, G and M/H. Then we compute mutual information of each unigram, bigram, or trigram feature to see which features are most discriminative. As Table 1 shows, 18 out of 30

Category	Words/Expressions
Unigram	my, I, I'm, I'll, but, was, I've, love, dad, have
Bigram	I love, I was, I have, my dad, go to, my mom, with my, have to, to go, my mum
Trigram	I have a, is going to, to go to, want to go, and I was, going to miss, I love him, I think I, I was like, I wish I

Table 1: High ranked words and expressions by mutual information between G and M/H level in annotated conversations.

Category	Words - SECRET	Words - Annotated
physical appearance	acne, hair, overweight, stomach, chest, hand, scar, thighs, chubby	ankle, face, toe, skin
mental/ physical condition	addicted, bulimia, doctor, illness, alcoholic, disease, drugs, pills	ache, epilepsy, pain, chiropractor, codeine

Table 3: Example words for identifying H level of SD from secret posts (2nd column) and annotated data (3rd column). Categories are hand-labeled.

most highly ranked discriminative features contain a first-person pronoun.

2.3 M Level of Self-Disclosure

Utterances with M level include two types: 1) information related with past events and future plans, and 2) general information about self (Barak and Gluck-Ofri, 2007). For the former, we add as seed trigrams 'I have been' and 'I will'. For the latter, we use seven types of information generally accepted to be personally identifiable information (McCallister, 2010), as listed in the left column of Table 2. To find the appropriate trigrams for those, we take Twitter conversation data (described in Section 4.1) and look for trigrams that begin with 'I' and 'my' and occur more than 200 times. We then check each one to see whether it is related with any of the seven types listed in the table. As a result, we find 57 seed trigrams for M level. Table 2 shows several examples.

Type	Trigram
Name	My name is, My last name
Birthday	My birthday is, My birthday party
Location	I live in, I lived in, I live on
Contact	My email address, My phone number
Occupation	My job is, My new job
Education	My high school, My college is
Family	My dad is, My mom is, My family is

Table 2: Example seed trigrams for identifying M level of SD. There are 51 of these used in SDTM.

2.4 H Level of Self-Disclosure

Utterances with H level express secretive wishes or sensitive information that exposes self or someone close (Barak and Gluck-Ofri, 2007). These are generally kept as secrets. With this intuition, we crawled 26,523 posts from *Six Billion Secrets*[1] site where users post secrets anonymously[2]. We call this external dataset SECRET. Unlike G and M levels, evidence of H level of self-disclosure tends to be topical, such as physical appearance, mental and physical illnesses, and family problems, so we take an approach of fitting a topic model driven by seed words. A similar approach has been successful in sentiment classification (Jo and Oh, 2011; Kim et al., 2013).

A critical component of this approach is the set of seed words with which to drive the discovery of topics that are most indicative of H level self-disclosure. To extract the seed words that express secretive personal information, we compute mutual information (Manning et al., 2008) with SECRET and 24,610 randomly selected tweets. We select 1,000 words with high mutual information and filter out stop words. Table 3 shows some of these words. To extract seed trigrams of secretive wishes, we again look for trigrams that start with 'I' or 'my', occur more than 200 times, and select trigrams of wishful thinking, such as 'I want to', and 'I wish I'. In total, there are 88 seed words and 8 seed trigrams for H.

Since SECRET is quite different from Twitter, we must show that posts in SECRET are semantically similar to the H level Tweets. Rather than directly comparing SECRET posts and Tweets, we use the same method of extracting discriminative word features from the annotated H level Tweets (see Section 4.2). Table 3 shows the seed words extracted from SECRET as well as the annotated Tweets. Because the annotated dataset consists of only 200 conversations, the coverage of the topics seems narrower than the much larger SECRETS, but both datasets show similarities in the topics. This, combined with the results of the model with the two sets of seed words (see Section 5 for the results), shows that SECRETS is an effective and simple-to-obtain substitute for an annotated corpus of H level of self-disclosure.

[1] http://www.sixbillionsecrets.com

[2] This site is regularly monitored for spam.

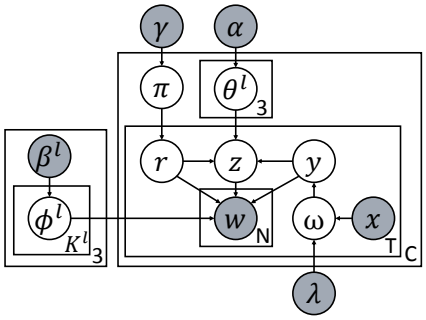


Figure 2: Graphical model of SDTM

Notation	Description
G; M; H	{general; medium; high} SD level
C; T; N	Number of conversations; tweets; words
K^G; K^M; K^H	Number of topics for {G; M; H}
c; ct	Conversation; tweet in conversation c
y_{ct}	SD level of tweet ct, G or M/H
r_{ct}	SD level of tweet ct, M or H
z_{ct}	Topic of tweet ct
w_{ctn}	n^{th} word in tweet ct
$\boldsymbol{\lambda}$	Learned Maximum entropy parameters
$\boldsymbol{x}_{ct}$	First-person pronouns features
$\boldsymbol{\omega}_{ct}$	Distribution over SD level of tweet ct
$\boldsymbol{\pi}_c$	SD level proportion of conversation c
$\boldsymbol{\theta}_c^G$; $\boldsymbol{\theta}_c^M$; $\boldsymbol{\theta}_c^H$	Topic proportion of {G; M; H} in conversation c
$\boldsymbol{\phi}^G$; $\boldsymbol{\phi}^M$; $\boldsymbol{\phi}^H$	Word distribution of {G; M; H}
$\boldsymbol{\alpha}$; $\boldsymbol{\gamma}$	Dirichlet prior for θ; π
$\boldsymbol{\beta}^G$, $\boldsymbol{\beta}^M$; $\boldsymbol{\beta}^H$	Dirichlet prior for ϕ^G; ϕ^M; ϕ^H
n_{cl}	Number of tweets assigned SD level l in conversation c
n_{ck}^l	Number of tweets assigned SD level l and topic k in conversation c
n_{kv}^l	Number of instances of word v assigned SD level l and topic k
m_{ctkv}	Number of instances of word v assigned topic k in tweet ct

Table 4: Summary of notations used in SDTM

3 Self-Disclosure Topic Model

This section describes our model, the self-disclosure topic model (SDTM), for classifying self-disclosure level and discovering topics for each self-disclosure level.

3.1 Model

In section 2, we discussed different approaches to identifying each level of self-disclosure, based on social science literature, annotated and unannotated Tweets, and an external corpus of secret posts. In this section, we describe our self-disclosure topic model, based on the widely used latent Dirichlet allocation (Blei et al., 2003), which incorporates those approaches.

Figure 2 illustrates the graphical model of

1. For each level $l \in \{\texttt{G}, \texttt{M}, \texttt{H}\}$:
   For each topic $k \in \{1, \ldots, K^l\}$:
   Draw $\boldsymbol{\phi}_k^l \sim Dir(\boldsymbol{\beta}^l)$
2. For each conversation $c \in \{1, \ldots, C\}$:
   (a) Draw $\boldsymbol{\theta}_c^G \sim Dir(\boldsymbol{\alpha})$
   (b) Draw $\boldsymbol{\theta}_c^M \sim Dir(\boldsymbol{\alpha})$
   (c) Draw $\boldsymbol{\theta}_c^H \sim Dir(\boldsymbol{\alpha})$
   (d) Draw $\boldsymbol{\pi}_c \sim Dir(\boldsymbol{\gamma})$
   (e) For each message $t \in \{1, \ldots, T\}$:
      i. Observe first-person pronouns features $\boldsymbol{x}_{ct}$
      ii. Draw $\boldsymbol{\omega}_{ct} \sim MaxEnt(\boldsymbol{x}_{ct}, \boldsymbol{\lambda})$
      iii. Draw $y_{ct} \sim Bernoulli(\boldsymbol{\omega}_{ct})$
      iv. If $y_{ct} = 0$ which is G level:
         A. Draw $z_{ct} \sim Mult(\boldsymbol{\theta}_c^G)$
         B. For each word $n \in \{1, \ldots, N\}$:
            Draw word $w_{ctn} \sim Mult(\boldsymbol{\phi}_{z_{ct}}^G)$
      Else which can be M or H level:
         A. Draw $r_{ct} \sim Mult(\boldsymbol{\pi}_c)$
         B. Draw $z_{ct} \sim Mult(\boldsymbol{\theta}_c^{r_{ct}})$
         C. For each word $n \in \{1, \ldots, N\}$:
            Draw word $w_{ctn} \sim Mult(\boldsymbol{\phi}_{z_{ct}}^{r_{ct}})$

Figure 3: Generative process of SDTM.

SDTM and how those approaches are embodied in it. The first approach based on the first-person pronouns is implemented by the observed variable $\boldsymbol{x}_{ct}$ and the parameters $\boldsymbol{\lambda}$ from a maximum entropy classifier for G vs. M/H level. The approach of seed words and phrases for levels M and H is implemented by the three separate word-topic probability vectors for the three levels of SD: $\boldsymbol{\phi}^l$ which has a Bayesian informative prior $\boldsymbol{\beta}^l$ where $l \in \{G, M, H\}$, the three levels of self-disclosure. Table 4 lists the notations used in the model and the generative process, and Figure 3 describes the generative process.

3.2 Classifying G vs M/H levels

Classifying the SD level for each tweet is done in two parts, and the first part classifies G vs. M/H levels with first-person pronouns (*I, my, me*). In the graphical model, y is the latent variable that represents this classification, and $\boldsymbol{\omega}$ is the distribution over y. x is the observation of the first-person pronoun in the tweets, and $\boldsymbol{\lambda}$ are the parameters learned from the maximum entropy classifier. With the annotated Twitter conversation dataset (described in Section 4.2), we experimented with several classifiers (Decision tree, Naive Bayes) and chose the maximum entropy classifier because it performed the best, similar to other joint topic models (Zhao et al., 2010; Mukherjee et al., 2013).

3.3 Classifying M vs H levels

The second part of the classification, the M and the H level, is driven by informative priors with seed words and seed trigrams. In the graphical model, r is the latent variable that represents this classification, and $\boldsymbol{\pi}$ is the distribution over r. $\boldsymbol{\gamma}$ is a non-informative prior for $\boldsymbol{\pi}$, and $\boldsymbol{\beta}^l$ is an informative prior for each SD level by seed words. For example, we assign a high value for the seed word 'acne' for $\boldsymbol{\beta}^H$, and a low value for 'My name is'. This approach is the same as joint models of topic and sentiment (Jo and Oh, 2011; Kim et al., 2013).

3.4 Inference

For posterior inference of SDTM, we use collapsed Gibbs sampling which integrates out latent random variables $\boldsymbol{\omega}, \boldsymbol{\pi}, \boldsymbol{\theta}$, and $\boldsymbol{\phi}$. Then we only need to compute $\boldsymbol{y}, \boldsymbol{r}$ and $\boldsymbol{z}$ for each tweet. We compute full conditional distribution $p(y_{ct} = j', r_{ct} = l', z_{ct} = k' | \boldsymbol{y}_{-ct}, \boldsymbol{r}_{-ct}, \boldsymbol{z}_{-ct}, \boldsymbol{w}, \boldsymbol{x})$ for tweet ct as follows:

$$p(y_{ct} = 0, z_{ct} = k' | \boldsymbol{y}_{-ct}, \boldsymbol{r}_{-ct}, \boldsymbol{z}_{-ct}, \boldsymbol{w}, \boldsymbol{x}) \propto \frac{exp(\boldsymbol{\lambda}_0 \cdot \boldsymbol{x}_{ct})}{\sum_{j=0}^{1} exp(\boldsymbol{\lambda}_j \cdot \boldsymbol{x}_{ct})} \, g(c, t, l', k'),$$

$$p(y_{ct} = 1, r_{ct} = l', z_{ct} = k' | \boldsymbol{y}_{-ct}, \boldsymbol{r}_{-ct}, \boldsymbol{z}_{-ct}, \boldsymbol{w}, \boldsymbol{x}) \propto \frac{exp(\boldsymbol{\lambda}_1 \cdot \boldsymbol{x}_{ct})}{\sum_{j=0}^{1} exp(\boldsymbol{\lambda}_j \cdot \boldsymbol{x}_{ct})} (\gamma_{l'} + n_{cl'}^{(-ct)}) \, g(c, t, l', k'),$$

where $\boldsymbol{z}_{-ct}, \boldsymbol{r}_{-ct}, \boldsymbol{y}_{-ct}$ are $\boldsymbol{z}, \boldsymbol{r}, \boldsymbol{y}$ without tweet ct, $m_{ctk'(\cdot)}$ is the marginalized sum over word v of $m_{ctk'v}$ and the function $g(c, t, l', k')$ as follows:

$$g(c, t, l', k') = \frac{\Gamma(\sum_{v=1}^{V} \beta_v^{l'} + n_{k'v}^{l'-(ct)})}{\Gamma(\sum_{v=1}^{V} \beta_v^{l'} + n_{k'v}^{l'-(ct)} + m_{ctk'(\cdot)})} \left(\frac{\alpha_{k'} + n_{ck'}^{l'(-ct)}}{\sum_{k=1}^{K} \alpha_k + n_{ck}^{l'}} \right) \prod_{v=1}^{V} \frac{\Gamma(\beta_v^{l'} + n_{k'v}^{l'-(ct)} + m_{ctk'v})}{\Gamma(\beta_v^{l'} + n_{k'v}^{l'-(ct)})}.$$

4 Data Collection and Annotation

To test our self-disclosure topic model, we use a large dataset of conversations consisting of Tweets over three years such that we can analyze the relationship between self-disclosure behavior and conversation frequency and length over time. We chose to crawl Twitter because it offers a practical and large source of conversations (Ritter et al., 2010). Others have also analyzed Twitter conversations for natural language and social media research (boyd et al., 2010; Danescu-Niculescu-Mizil et al., 2011), but we collect conversations from the same set of dyads over several months for a unique longitudinal dataset. We also make sure that each conversation is at least five tweets, and that each dyad has at least twenty conversations.

Users	Dyads	Conv's	Tweets
101,686	61,451	1,956,993	17,178,638

Table 5: Dataset of Twitter conversations. We chose conversations consisting of five or more tweets each. We chose dyads with twenty or more conversations.

4.1 Collecting Twitter conversations

We define a Twitter conversation as a chain of tweets where two users are consecutively replying to each other's tweets using the Twitter reply button. We initialize the set of users by randomly sampling thirteen users who reply to other users in English from the Twitter public streams[3]. Then we crawl each user's public tweets, and look at users who are mentioned in those tweets. It is a breadth-first search in the network defined by users as nodes and edges as conversations. We run this search for dyads until the depth of four, and filter out users who tweet in a non-English language. We use an open source tool for detecting English tweets[4]. To protect users' privacy, we replace Twitter userid, usernames and url in tweets with random strings. This dataset consists of 101,686 users, 61,451 dyads, 1,956,993 conversations and 17,178,638 tweets which were posted between August 2007 to July 2013. Table 5 summarizes the dataset.

4.2 Annotating self-disclosure level

To measure the accuracy of our model, we randomly sample 301 conversations, each with ten or fewer tweets, and ask three judges, fluent in English and graduate students/researchers, to annotate each tweet with the level of self-disclosure. Judges first read and discussed the definitions and examples of self-disclosure level shown in (Barak and Gluck-Ofri, 2007), then they worked separately on a Web-based platform.

As a result of annotation, there are 122 G level converstaions, 147 M level and 32 H level con-

[3] https://dev.twitter.com/docs/api/streaming

[4] https://github.com/shuyo/ldig

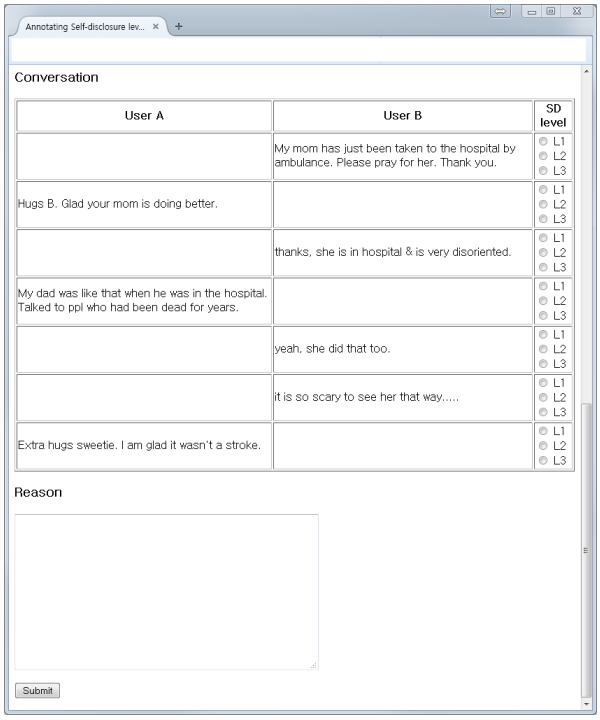


Figure 4: Screenshot of annotation web-based platform. Annotators read a Twitter conversation and annotate self-disclosure level to each tweet.

versations, and inter-rater agreement using Fleiss kappa (Fleiss, 1971) is 0.68, which is substantial agreement result (Landis and Koch, 1977).

5 Classification of Self-Disclosure Level

This section describes experiments and results of SDTM as well as several other methods for classification of self-disclosure level.

We first start with the annotated dataset in section 4.2 in which each tweet is annotated with SD level. We then aggregate all of the tweets of a conversation, and we compute the proportions of tweets in each SD level. When the proportion of tweets at `M` or `H` level is equal to or greater than 0.2, we take the level of the larger proportion and assign that level to the conversation. When the proportions of tweets at `M` or `H` level are both less than 0.2, we assign `G` to the SD level. The reason for setting 0.2 as the threshold is that a conversation containing tweets with `H` or `M` level of self-disclosure usually starts with a greeting or a general comment, and contains one or more questions or comments before or after the self-disclosure tweet.

We compare SDTM with the following methods for classifying conversations for SD level:

- LDA (Blei et al., 2003): A Bayesian topic model. Each conversation is treated as a document. Used in previous work (Bak et al., 2012).
- MedLDA (Zhu et al., 2012): A supervised topic model for document classification. Each conversation is treated as a document and response variable can be mapped to a SD level.
- LIWC (Tausczik and Pennebaker, 2010): Word counts of particular categories[5]. Used in previous work (Houghton and Joinson, 2012).
- Bag of Words + Bigrams + Trigrams (BOW+): A bag of words, bigram and trigram features. We exclude features that appear only once or twice.
- Seed words and trigrams (SEED): Occurrences of seed words/trigrams from SECRET which are described in section 3.3.
- SDTM with seed words from annotated Tweets (SDTM−): To compare with SDTM below using seed words from SECRET, this uses seed words from the annotated data described in section 2.4.
- ASUM (Jo and Oh, 2011): A joint model of sentiments and topics. We map each SD level to one sentiment and use the same seed words/trigrams from SECRET as in SDTM below. Used in previous work (Bak et al., 2012).
- First-person pronouns (FirstP): Occurrence of first-person pronouns which are described in section 3.2. To identify first-person pronouns, we tagged parts of speech in each tweet with the Twitter POS tagger (Owoputi et al., 2013).
- First-person pronouns + Seed words/trigrams (FP+SE1): First-person pronouns and seed words/trigrams from SECRET.
- Two stage classifier with First-person pronouns + Seed words/trigrams (FP+SE2): A

[5]personal pronouns, 3rd person singular words, family words, human words, sexual words, etc

Method	*Acc*	G F_1	M F_1	H F_1	*Avg* F_1
LDA	49.2	0.00	0.65	0.05	0.23
MedLDA	43.3	0.41	0.52	0.09	0.34
LIWC	49.2	0.34	0.61	0.18	0.38
BOW+	54.1	0.50	0.59	0.15	0.41
SEED	54.4	0.52	0.60	0.14	0.42
ASUM	56.6	0.32	0.70	0.38	0.47
SDTM−	60.4	0.57	0.70	0.14	0.47
FirstP	63.2	0.63	0.69	0.10	0.47
FP+SE1	61.0	0.61	0.67	0.16	0.48
FP+SE2	60.4	**0.64**	0.69	0.17	0.50
SDTM	**64.5**	0.61	**0.71**	**0.43**	**0.58**

Table 6: SD level classification accuracies and F-measures using annotated data. *Acc* is accuracy, and G F_1 is F-measure for classifying the G level. *Avg* F_1 is the macroaveraged value of G F_1, M F_1 and H F_1. SDTM outperforms all other methods compared. The difference between SDTM and FirstP is statistically significant (p-value < 0.05 for accuracy, < 0.0001 for *Avg* F_1).

two stage classifier with first-person pronouns and seed words/trigrams from SECRET. In the first stage, the classifier identifies G with first-person pronouns. Then in the second stage, the classifier uses seed words and trigrams to identify M and H levels.

- SDTM: Our model with first-person pronouns and seed words/trigrams from SECRET.

SEED, LIWC, LDA and FirstP cannot be used directly for classification, so we use Maximum entropy model with outputs of each of those models as features[6]. BOW+ uses SVM with a radial basis kernel which performs better than all other settings tried including maximum entropy. We split the data randomly into 80/20 for train/test. We run MedLDA, ASUM and SDTM 20 times each and compute the average accuracies and F-measure for each level. We run LDA and MedLDA with various number of topics from 80 to 140, and 120 topics shows best outputs. So we set 120 topics for LDA, MedLDA and ASUM, 60; 40; 40 topics for SDTM K^G, K^M and K^H respectively which is best perform from 40; 40; 40 to 60; 60; 60 topics. We assume that a conversation has few topics and self-disclosure levels, so we set $\alpha = \gamma = 0.1$ (Tang et al., 2014). To incorporate the seed words and trigrams into ASUM and SDTM, we initialize $\boldsymbol{\beta}^G, \boldsymbol{\beta}^M$ and $\boldsymbol{\beta}^H$ differently. We assign a high value of 2.0 for each seed word and trigram for that level, and a low value of 10^{-6} for each word that is a seed word for another level, and a default value of 0.01 for all other words. This approach is the same as previous papers (Jo and Oh, 2011; Kim et al., 2013).

As Table 6 shows, SDTM performs better than the other methods for accuracy as well as F-measure. LDA and MedLDA generally show the lowest performance, which is not surprising given these models are quite general and not tuned specifically for this type of semi-supervised classification task. BOW which is simple word features also does not perform well, showing especially low F-measure for the H level. LIWC and SEED perform better than LDA, but these have quite low F-measure for G and H levels. ASUM shows better performance for classifying H level than others, confirming the effectiveness of a topic modeling approach to this difficult task, but not as well as SDTM. FirstP shows good F-measure for the G level, but the H level F-measure is quite low, even lower than SEED. Combining first-person pronouns and seed words and trigrams (FP+SE1) shows better than each feature alone, and the two stage classifier (FP+SE2) which is a similar approach taken in SDTM shows better results. Finally, SDTM classifies G and M level at a similar accuracy with FirstP, FP+SE1 and FP+SE2, but it significantly improves accuracy for the H level compared to all other methods.

6 Relations of Self-Disclosure and Conversation Behaviors

In this section, we investigate whether there is a relationship between self-disclosure and conversation behaviors over time. Self-disclosure is one way to maintain and improve relationships (Jourard, 1971; Joinson and Paine, 2007). So two people's intimacy changes over time has relationship with self-disclosure in their conversation. However, it is hard to identify intimacy between users in large scale online social network. So we choose conversation behaviors such as conversation frequency and length which can be treated as proxies for measuring intimacy between two people (Emmers-Sommer, 2004; Bak et al., 2012).

[6]It performs better than other classifiers (C4.5, Naive-Bayes, SVM with linear kernel, polynomial kernel and radial basis)

With SDTM, we can automatically classify the SD level of a large number of conversations, so we investigate whether there is a similar relationship between self-disclosure in conversations and subsequent conversation behaviors with the same partner on Twitter.

For comparing conversation behaviors over time, we divided the conversations into two sets for each dyad. For the $initial$ period, we include conversations from the dyad's first conversation to 20 days later. And for the $subsequent$ period, we include conversations during the subsequent 10 days. We compute proportions of conversation for each SD level for each dyad in the $initial$ and $subsequent$ periods.

More specifically, we ask the following three questions:

1. If a dyad shows high conversation frequency at a particular time period, would they display higher SD in their subsequent conversations?

2. If a dyad displays high SD level in their conversations at a particular time period, would their subsequent conversations be longer?

3. If a dyad displays high overall SD level, would their conversations increase in length over time more than dyads with lower overall SD level?

6.1 Experiment Setup

We first run SDTM with all of our Twitter conversation data with 150; 120; 120 topics for SDTM K^G, K^M and K^H respectively. The hyper-parameters are the same as in section 5. To handle a large dataset, we employ a distributed algorithm (Newman et al., 2009), and run with 28 threads.

Table 7 shows some of the topics that were prominent in each SD level by KL-divergence. As expected, `G` level includes general topics such as food, celebrity, soccer and IT devices, `M` level includes personal communication and birthday, and finally, `H` level includes sickness and profanity.

We define a new measurement, SD level score for a dyad in the period, which is a weighted sum of each conversation with SD levels mapped to 1, 2, and 3, for the levels `G`, `M`, and `H`, respectively.

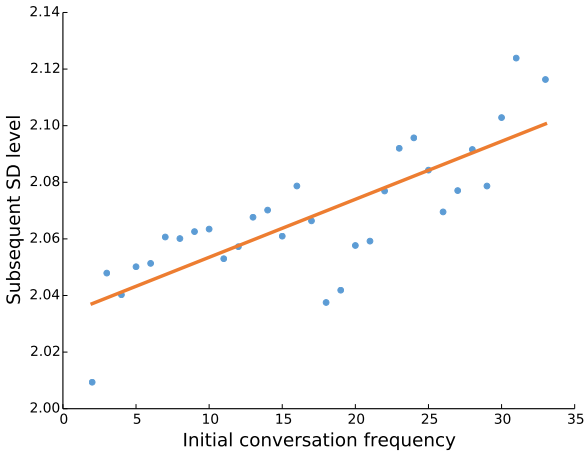


Figure 5: Relationship between initial conversation frequency and subsequent SD level. The solid line is the linear regression line, and the coefficient is 0.0020 with $p < 0.0001$, which shows a significant positive relationship.

6.2 Does high frequency of conversation lead to more self-disclosure?

We investigate whether the $initial$ conversation frequency is correlated with the SD level in the $subsequent$ period. We run linear regression with the initial conversation frequency as the independent variable, and SD level in the subsequent period as the dependent variable.

The regression coefficient is 0.0020 with low p-value ($p < 0.0001$). Figure 5 shows the scatter plot. We can see that the slope of the regression line is positive.

6.3 Does high self-disclosure lead to longer conversations?

Now we investigate the effect of the self-disclosure level to conversation length. We run linear regression with the intial SD level score as the independent variable, and the rate of change in conversation length between $initial$ period and $subsequent$ period as the dependent variable. Conversation length is measured by the number of tweets in a conversation.

The result of regression is that the independent variable's coefficient is 0.048 with a low p-value ($p < 0.0001$). Figure 6 shows the scatter plot with the regression line, and we can see that the slope of regression line is positive.

G level			M level			H level		
101	184	176	36	104	82	113	33	19
chocolate	obama	league	send	twitter	going	ass	better	lips
butter	he's	win	email	follow	party	bitch	sick	kisses
good	romney	game	i'll	tumblr	weekend	fuck	feel	love
cake	vote	season	sent	tweet	day	yo	throat	smiles
peanut	right	team	dm	following	night	shit	cold	softly
milk	president	cup	address	account	dinner	fucking	hope	hand
sugar	people	city	know	fb	birthday	lmao	pain	eyes
cream	good	arsenal	check	followers	tomorrow	shut	good	neck
make	going	chelsea	link	facebook	come	dick	cough	arms
love	time	liverpool	need	followed	i'll	kick	bad	head
yum	party	won	message	omg	family	face	i've	smirks
hot	election	football	let	right	fun	hoe	need	slowly
cookies	gop	united	sure	saw	friends	lmfao	sore	hair
banana	paul	final	thanks	page	tonight	nigga	flu	face
bread	way	away	my email	timeline	plans	bi	today	chest

Table 7: High ranked topics in each level by comparing KL-divergence with other level's topics

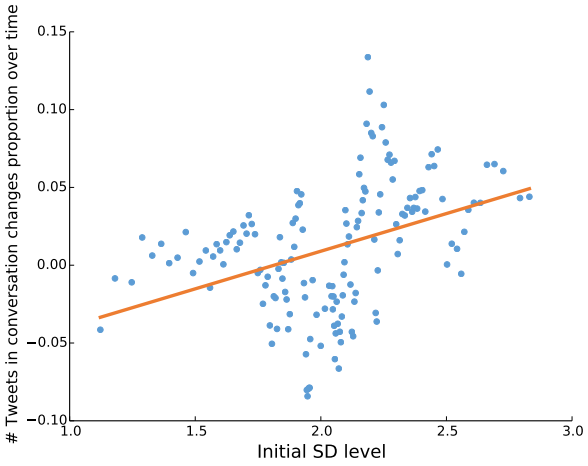


Figure 6: Relationship between initial SD level and conversation length changes over time. The solid line is the linear regression line, and the coefficient is 0.048 with $p < 0.0001$, which shows a significant positive relationship.

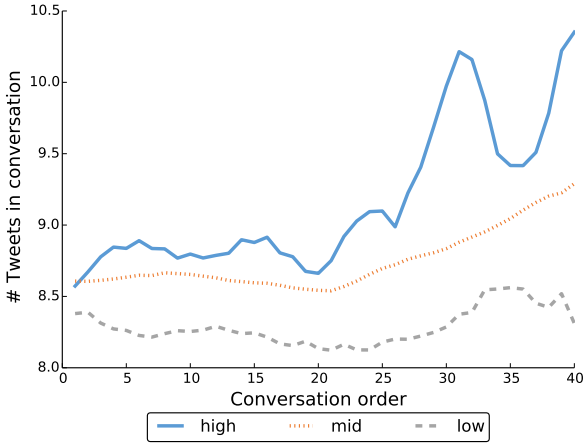


Figure 7: Changes in conversation length over time. We divide dyads into three groups by SD level score as low, medium, and high. Conversation length noticeably increases over time in the medium and high groups, but only slight in the low group.

6.4 Is there a difference in conversation length patterns over time depending on overall SD level?

Now we investigate the conversation length changes over time with three groups, low, medium, and high, by overall SD level. Then we investigate changes in conversation length over time.

Figure 7 shows the results of this investigation. First, conversations are generally lengthier when SD level is high. This phenomenon is also observed in figure 6, but here we can see it as a long-term persistent pattern. Second, conversation length increases consistently and significantly for the high and medium groups, but for the low SD group, there is not a significant increase of conversation length over time.

7 Related Work

Prior work on quantitatively analyzing self-disclosure has relied on user surveys (Ledbetter et

al., 2011; Trepte and Reinecke, 2013) or human annotation (Barak and Gluck-Ofri, 2007; Courtney Walton and Rice, 2013). These methods consume much time and effort, so they are not suitable for large-scale studies. In prior work closest to ours, Bak et al. (2012) showed that a topic model can be used to identify self-disclosure, but that work applies a two-step process in which a basic topic model is first applied to find the topics, and then the topics are post-processed for binary classification of self-disclosure. We improve upon this work by applying a single unified model of topics and self-disclosure for high accuracy in classifying the three levels of self-disclosure.

Subjectivity which is aspect of expressing opinions (Pang and Lee, 2008; Wiebe et al., 2004) is related with self-disclosure, but they are different dimensions of linguistic behavior. Because there indeed are many high self-disclosure tweets that are subjective, but there are also counter examples in annotated dataset. The tweet "England manager is Roy Hodgson." is low self-disclosure and low subjectivity, "I have barely any hair left." is high self-disclosure but low subjectivity, and "Senator stop lying!" is low self-disclosure but high subjectivity.

8 Conclusion and Future Work

In this paper, we have presented the self-disclosure topic model (SDTM) for discovering topics and classifying SD levels from Twitter conversation data. We devised a set of effective seed words and trigrams, mined from a dataset of secrets. We also annotated Twitter conversations to make a ground-truth dataset for SD level. With annotated data, we showed that SDTM outperforms previous methods in classification accuracy and F-measure. We publish the source code of SDTM and the dataset include annotated Twitter conversations and SECRET publicly[7].

We also analyzed the relationship between SD level and conversation behaviors over time. We found that there is a positive correlation between initial SD level and subsequent conversation length. Also, dyads show higher level of SD if they initially display high conversation frequency. Finally, dyads with overall medium and high SD level will have longer conversations over time. These results support previous results in social psychology research with more robust results from a large-scale dataset, and show the effectiveness of computationally analyzing at SD behavior.

There are several future directions for this research. First, we can improve our modeling for higher accuracy and better interpretability. For instance, SDTM only considers first-person pronouns and topics. Naturally, there are other linguistic patterns that can be identified by humans but not captured by pronouns and topics. Second, the number of topics for each level is varied, and so we can explore nonparametric topic models (Teh et al., 2006) which infer the number of topics from the data. Third, we can look at the relationship between self-disclosure behavior and general online social network usage beyond conversations. We will explore these directions in our future work.

[7] http://uilab.kaist.ac.kr/research/EMNLP2014

Acknowledgments

We would like to thank Jing Liu and Wayne Xin Zhao for inspiring discussions, and the anonymous reviewers for helpful comments. Alice Oh is supported by ICT R&D program of MSIP/IITP [10041313, UX-oriented Mobile SW Platform].

References

JinYeong Bak, Suin Kim, and Alice Oh. 2012. Self-disclosure and relationship strength in twitter conversations. In *Proceedings of ACL*.

Azy Barak and Orit Gluck-Ofri. 2007. Degree and reciprocity of self-disclosure in online forums. *CyberPsychology & Behavior*, 10(3):407–417.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

danah boyd, Scott Golder, and Gilad Lotan. 2010. Tweet, tweet, retweet: Conversational aspects of retweeting on twitter. In *Proceedings of HICSS*.

S Courtney Walton and Ronald E Rice. 2013. Mediated disclosure on twitter: The roles of gender and identity in boundary impermeability, valence, disclosure, and stage. *Computers in Human Behavior*, 29(4):1465–1474.

Cristian Danescu-Niculescu-Mizil, Michael Gamon, and Susan Dumais. 2011. Mark my words!: Linguistic style accommodation in social media. In *Proceedings of WWW*.

Tara M Emmers-Sommer. 2004. The effect of communication quality and quantity indicators on intimacy and relational satisfaction. *Journal of Social and Personal Relationships*, 21(3):399–411.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

David J Houghton and Adam N Joinson. 2012. Linguistic markers of secrets and sensitive self-disclosure in twitter. In *Proceedings of HICSS*.

Yohan Jo and Alice H Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of WSDM*.

Adam N Joinson and Carina B Paine. 2007. Self-disclosure, privacy and the internet. *The Oxford handbook of Internet psychology*, pages 237–252.

Adam N Joinson. 2001. Self-disclosure in computer-mediated communication: The role of self-awareness and visual anonymity. *European Journal of Social Psychology*, 31(2):177–192.

Sidney M Jourard. 1971. Self-disclosure: An experimental analysis of the transparent self.

Suin Kim, Jianwen Zhang, Zheng Chen, Alice Oh, and Shixia Liu. 2013. A hierarchical aspect-sentiment model for online reviews. In *Proceedings of AAAI*.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Andrew M Ledbetter, Joseph P Mazer, Jocelyn M DeGroot, Kevin R Meyer, Yuping Mao, and Brian Swafford. 2011. Attitudes toward online social connection and self-disclosure as predictors of facebook communication and relational closeness. *Communication Research*, 38(1):27–53.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge.

Erika McCallister. 2010. *Guide to protecting the confidentiality of personally identifiable information*. DIANE Publishing.

Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Sharon Meraz. 2013. Public dialogue: Analysis of tolerance in online discussions. In *Proceedings of ACL*.

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2009. Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10:1801–1828.

Olutobi Owoputi, Brendan OConnor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of HLT-NAACL*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Proceedings of HLT-NAACL*.

Jian Tang, Zhaoshi Meng, Xuanlong Nguyen, Qiaozhu Mei, and Ming Zhang. 2014. Understanding the limiting factors of topic modeling via posterior contraction analysis. In *Proceedings of The 31st International Conference on Machine Learning*, pages 190–198.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of Language and Social Psychology*.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2006. Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476).

Sabine Trepte and Leonard Reinecke. 2013. The reciprocal effects of social network site use and the disposition for self-disclosure: A longitudinal study. *Computers in Human Behavior*, 29(3):1102 – 1112.

Sarah I Vondracek and Fred W Vondracek. 1971. The manipulation and measurement of self-disclosure in preadolescents. *Merrill-Palmer Quarterly of Behavior and Development*, 17(1):51–58.

Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Computational linguistics*, 30(3):277–308.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of EMNLP*.

Jun Zhu, Amr Ahmed, and Eric P Xing. 2012. Medlda: maximum margin supervised topic models. *Journal of Machine Learning Research*, 13:2237–2278.

Major Life Event Extraction from Twitter based on Congratulations/Condolences Speech Acts

Jiwei Li[1], **Alan Ritter**[2], **Claire Cardie**[3] **and Eduard Hovy**[4]
[1]Computer Science Department, Stanford University, Stanford, CA 94305, USA
[2]Department of Computer Science and Engineering, the Ohio State University, OH 43210, USA
[3]Computer Science Department, Cornell University, Ithaca, NY 14853, USA
[4]Language Technology Institute, Carnegie Mellon University, PA 15213, USA
jiweil@stanford.edu ritter.1492@osu.edu
cardie@cs.cornell.edu ehovy@andrew.cmu.edu

Abstract

Social media websites provide a platform for anyone to describe significant events taking place in their lives in realtime. Currently, the majority of personal news and life events are published in a textual format, motivating information extraction systems that can provide a structured representations of major life events (weddings, graduation, etc...). This paper demonstrates the feasibility of accurately extracting major life events. Our system extracts a fine-grained description of users' life events based on their published tweets. We are optimistic that our system can help Twitter users more easily grasp information from users they take interest in following and also facilitate many downstream applications, for example realtime friend recommendation.

1 Introduction

Social networking websites such as Facebook and Twitter have recently challenged mainstream media as the freshest source of information on important news events. In addition to an important source for breaking news, social media presents a unique source of information on private events, for example a friend's engagement or college graduation (examples are presented in Figure 1). While a significant amount of previous work has investigated event extraction from Twitter (e.g., (Ritter et al., 2012; Diao et al., 2012)), existing approaches mostly focus on public bursty event extraction, and little progress has been made towards the problem of automatically extracting the major life events of ordinary users.

A system which can automatically extract major life events and generate fine-grained descriptions as in Figure 1 will not only help Twitter users with the problem of information overload by summarizing important events taking place in their friends lives, but could also facilitate downstream applications such as friend recommendation (e.g., friend recommendation in realtime to people who were just admitted into the same university, get the same jobs or internships), targeted online advertising (e.g., recommend baby care products to newly expecting mothers, or wedding services to new couples), information extraction, etc.

Before getting started, we first identify a number of key challenges in extracting significant life events from user-generated text, which account the reason for the lack of previous work in this area:

Challenge 1: Ambiguous Definition for Major Life Events Major life event identification is an open-domain problem. While many types of events (e.g., marriage, engagement, finding a new job, giving birth) are universally agreed to be important, it is difficult to robustly predefine a list of characteristics for important life events on which algorithms can rely for extraction or classification.

Challenge 2: Noisiness of Twitter Data: The user-generated text found in social media websites such as Twitter is extremely noisy. The language used to describe life events is highly varied and ambiguous and social media users frequently discuss public news and mundane events from their daily lives, for instance what they ate for lunch.

Even for a predefined life event category, such as marriage, it is still difficult to accurately identify mentions. For instance, a search for the keyphrase "get married" using Twitter Search[1] results in a large number of returned results that do not correspond to a personal event:

- I want to **get married** once. No divorce & no cheating, just us two till the end.
  (error: wishes)

[1]https://twitter.com/search?q=get~married

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1997–2007,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

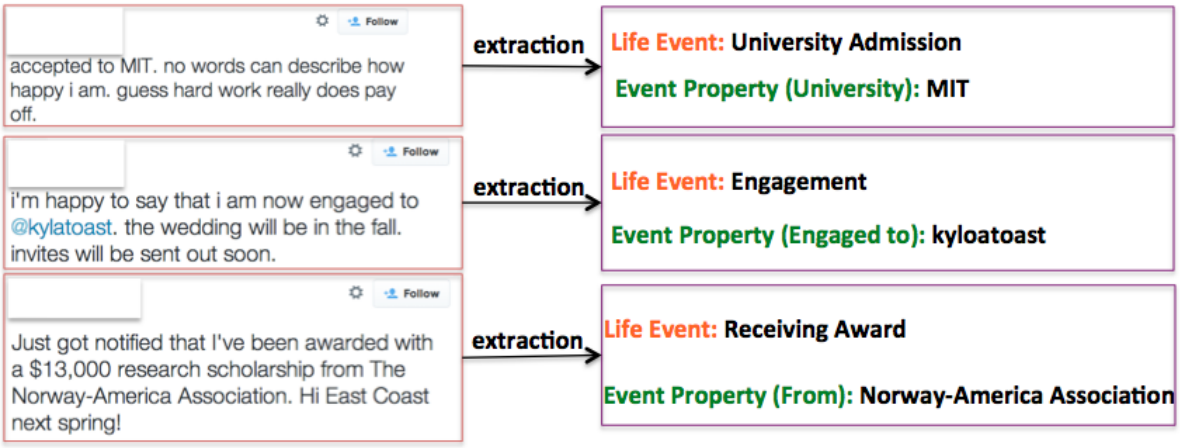


Figure 1: Examples of users mentioning personal life events on Twitter.

- Can Adam Sandler and Drew Barrymore just drop the pretense and **get married** already? (error: somebody else)
- I **got married** and had kids on purpose (error: past)

Challenge 3: the Lack of Training Data Collecting sufficient training data in this task for machine learning models is difficult for a number of reasons: (1) A traditional, supervised learning approach, requires explicit annotation guidelines for labeling, though it is difficult to know which categories are most representative in the data apriori. (2) Unlike public events which are easily identified based on message volume, significant private events are only mentioned by one or several users directly involved in the event. Many important categories are relatively infrequent, so even a large annotated dataset may contain just a few or no examples of these categories, making classification difficult.

In this paper, we present a pipelined system that addresses these challenges and extracts a structured representation of individual life events based on users' Twitter feeds. We exploit the insight to automatically gather large volumes of major life events which can be used as training examples for machine learning models. Although personal life events are difficult to identify using traditional approaches due to their highly diverse nature, we noticed that users' followers often directly reply to such messages with CONGRATULATIONS or CONDOLENCES speech acts, for example:

User1: *I got accepted into Harvard !*
User2: *Congratulations !*

These speech acts are easy to identify with high precision because the possible ways to express them are relatively constrained. Instead of directly inspecting tweets to determine whether they correspond to major life events, we start by identifying replies corresponding to CONGRATULATIONS or CONDOLENCES, and then retrieve the message they are in response to, which we assume refer to important life events.

The proposed system automatically identifies major life events and then extracts correspondent event properties. Through the proposed system, we demonstrate that it is feasible to automatically reconstruct a detailed list of individual life events based on users' Twitter streams. We hope that work presented in this paper will facilitate downstream applications and encourage follow-up work on this task.

2 System Overview

An overview of the components of the system is presented in Figure 2. **Pipeline1** first identifies the major life event category the input tweet talks about and filters out the irrelevant tweets and will be described in Section 4. Next, **Pipeline2**, as, demonstrated in Section 5, identifies whether the speaker is directly involved in the life event. Finally, **Pipeline3** extracts the property of event and will be illustrated in Section 6.

Section 3 serves as the preparing step for the pipelined system, describing how we collect training data in large-scale. The experimental evaluation regarding each pipeline of the system is presented in the corresponding section (i.e., Section 4,5,6) and the end-to-end evaluation will be pre-

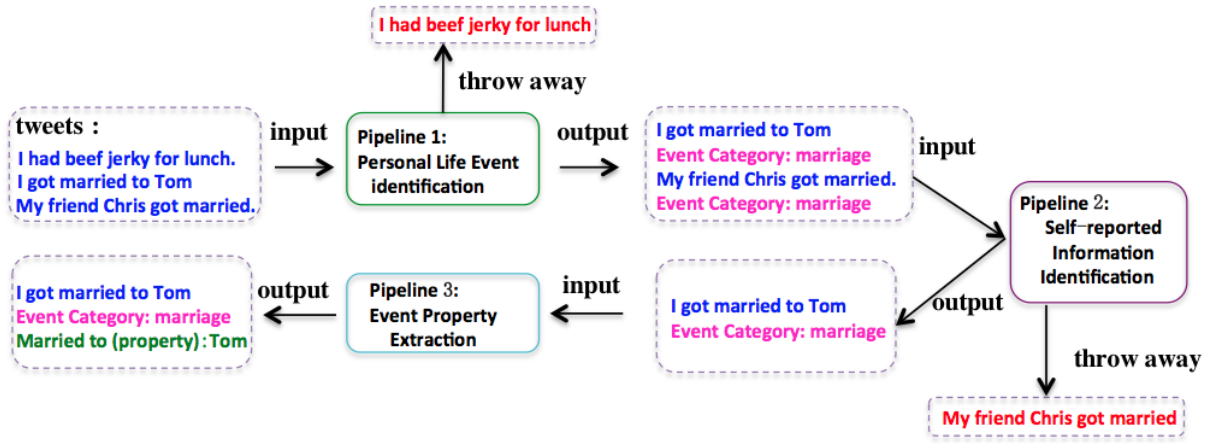


Figure 2: System Overview. Blue: original input tweets. Red: filtered out tweets. Magenta: life event category. Green: life event property. **Pipeline 1** identifies the life category the input tweet talks about (e.g., marriage, graduation) and filter out irrelevant tweets (e.g., I had beef stick for lunch). **Pipeline 2** identifies whether the speaker is directly involved in the event. It will preserve self-reported information (i.e. "I got married") and filtered out unrelated tweets (e.g., "my friend Chris got married"). **Pipeline 3** extracts the property of event (e.g. to whom the speaker married or the speaker admitted by which university).

sented in Section 7.

3 Personal Life Event Clustering

In this section, we describe how we identify common categories of major life events by leveraging large quantities of unlabeled data and obtain a collection of tweets corresponding to each type of identified event.

3.1 Response based Life Event Detection

While not all major life events will elicit CONGRATULATIONS or CONDOLENCES from a user's followers, this technique allows us to collect large volumes of high-precision personal life events which can be used to train models to recognize the diverse categories of major life events discussed by social media users.

3.2 Life Event Clustering

Based on the above intuition, we develop an approach to obtain a list of individual life event clusters. We first define a small set of seed responses which capture common CONGRATULATIONS and CONDOLENCES, including the phrases: "Congratulations", "Congrats", "Sorry to hear that", "Awesome", and gather tweets that were observed with seed responses. Next, an LDA (Blei et al., 2003)[2] based topic model is used to cluster the gathered tweets to automatically identify important categories of major life events in an unsupervised way. In our approach, we model the whole conversation dialogue as a document[3] with the response seeds (e.g., congratulation) masked out. We furthermore associate each sentence with a single topic, following strategies adopted by (Ritter et al., 2010; Gruber et al., 2007). We limit the words in our document collection to verbs and nouns which we found to lead to clearer topic representations, and used collapsed Gibbs Sampling for inference (Griffiths and Steyvers, 2004).

Next one of the authors manually inspected the resulting major life event types inferred by the model, and manually assigned them labels such as "getting a job", "graduation" or "marriage" and discarded incoherent topics[4]. Our methodology is inspired by (Ritter et al., 2012) that uses a LDA-CLUSTERING+HUMAN-IDENTIFICATION strategy to identify public events from Twitter. Similar strategies have been widely used in unsupervised information extraction (Bejan et al., 2009; Yao et al., 2011) and selectional preference

[2]Topic Number is set to 120.

[3]Each whole conversation usually contains multiple tweets and users.

[4]While we applied manual labeling and coherence evaluation in this work, an interesting direction for future work is automatically labeling major life event categories following previous work on labeling topics in traditional document-based topic models (Mimno et al., 2011; Newman et al., 2010).

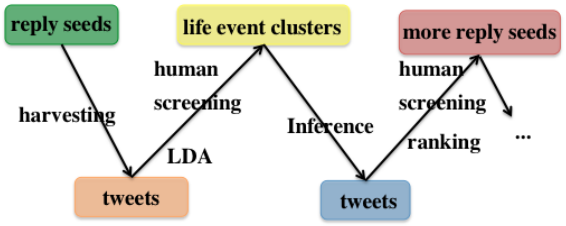


Figure 3: Illustration of bootstrapping process.

Input: Reply seed list $E = \{e\}$, Tweet conversation collection $T = \{t\}$, Retrieved Tweets Collection $D = \phi$. Identified topic list L=ϕ
Begin
While not stopping:

1. For unprocessed conversation $t \in T$ if t contains reply $e \in E$,
   - add t to D: $D = D + t$.
   - remove t from T: $T = T - t$
2. Run streaming LDA (Yao et al., 2009) on newly added tweets in D.
3. Manually Identify meaningful/trash topics, giving label to meaningful topics.
4. Add newly detected meaningful topic l to L.
5. For conversation t belonging to trash topics
   - remove t from D: $D = D - t$
6. Harvest more tweets based on topic distribution.
7. Manually identify top 20 responses to tweets harvested from Step 6.
8. Add meaningful responses to E.

End
Output: Identified topic list L. Tweet collection D.

Figure 4: Bootstrapping Algorithm for Response-based Life event identification.

modeling (Kozareva and Hovy, 2010a; Roberts and Harabagiu, 2011).

Conversation data was extracted from the CMU Twitter Warehouse of 2011 which contains a total number of 10% of all published tweets in that year.

3.3 Expanding dataset using Bootstrapping

While our seed patterns for identifying messages expressing CONGRATULATIONS and CONDOLENCES are very high precision, they don't cover all the possible ways these speech acts can be expressed. We therefore adopt a semi-supervised bootstrapping approach to expand our reply seeds and event-related tweets. Our bootstrapping approach is related to previous work on semi-supervised information harvesting (e.g., (Kozareva and Hovy, 2010b; Davidov et al., 2007)). To preserve the labeled topics from the first iteration, we apply a streaming approach to inference (Yao et al., 2009) over unlabeled tweets (those which did not match one of the response

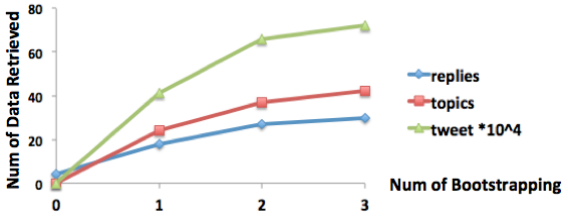


Figure 5: Illustration of data retrieved in each step of bootstrapping.

congratulations (cong, congrats); (that's) fantastic; (so) cool; (I'm) (very) sorry to hear that; (that's) great (good) new; awesome; what a pity; have fun; great; that sucks; too bad; (that's) unfortunate; how sad; fabulous; (that's) terrific; (that's) (so) wonderful; my deepest condolences;

Table 1: Responses retrieved from Bootstrapping.

seeds). We collect responses to the newly added tweets, then select the top 20 frequent replies[5]. Next we manually inspect and filter the top ranked replies, and use them to harvest more tweets. This process is then repeated with another round of inference in LDA including manual labeling of newly inferred topics, etc... An illustration of our approach is presented in Figure 3 and the details are presented in Figure 4. The algorithm outputs a collection of personal life topics L, and a collection of retrieved tweets D. Each tweet $d \in D$ is associated with a life event topic l, $l \in L$.

We repeat the bootstrapping process for 4 iterations and end up with 30 different CONGRATULATIONS and CONDOLENCES patterns (shown in Table 1) and 42 coherent event types which refer to significant life events (statistics for harvested data from each step is shown in Figure 5). We show examples of the mined topics with correspondent human labels in Table 3, grouped according to a specific kind of resemblance.

3.4 Summary and Discussion

The objective of this section is (1) identifying a category of life events (2) identifying tweets associated with each event type which can be used as candidates for latter self reported personal information and life event category identification.

We understand that the event list retrieved from our approach based on replies in the conversation is far from covering all types of personal events (especially the less frequent life events). But our

[5]We only treat the first sentence that responds to the beginning of the conversation as replies.

Life Event	Proportion
Birthday	9.78
Job	8.39
Wedding Engagement	7.24
Award	6.20
Sports	6.08
Anniversary	5.44
Give Birth	4.28
Graduate	3.86
Death	3.80
Admission	3.54
Interview Internship	3.44
Moving	3.26
Travel	3.24
Illness	2.45

Life Event	Proportion
Vacation	2.24
Relationship	2.16
Exams	2.02
Election	1.85
New Car	1.65
Running	1.42
Surgery	1.20
Lawsuit	0.64
Acting	0.50
Research	0.48
Essay	0.35
Lost Weight	0.35
Publishing	0.28
Song	0.22
OTHER	15.31

Table 2: List of automatically discovered life event types with percentage (%) of data covered.

list is still able to cover a large proportion of IMPORTANT and COMMON life events. Our latter work is focused on given a random tweet, identifying whether it corresponds to one of the 42 types of life events in our list.

Another thing worth noting here is that, while current section is not focused on self-reported information identification, we have already obtained a relatively clean set of data with a large proportion of non self-reported information related tweets being screened: people do not usually respond to non self-reported information with commonly used replies, or in other words, with replies that will pass our next step human test[6]. These non self-reported tweets would therefore be excluded from training data.

4 Life Event Identification

In this section, we focused on deciding whether a given tweet corresponds to one of the 42 predefined life events.

Our training dataset consists of approximately 72,000 tweets from 42 different categories of life events inferred by our topic model as described in Section 3. We used the top 25% of tweets for which our model assigned highest probability to each topic. For sparsely populated topics we used the top 50% of tweets to ensure sufficient coverage.

We further collected a random sample of about 10 million tweets from Twitter API[7] as non-life

[6]For example, people don't normally respond to "I want to **get married** once" (example in Challenge 2, Section 1) with "Congratulations".

[7]https://dev.twitter.com/

Human Label	Top words
Wedding &engagement	wedding, love, ring, engagement, engaged, bride, video, marrying
Relationship Begin	boyfriend, girlfriend, date, check, relationship, see, look
Anniversary	anniversary, years, year, married, celebrating, wife, celebrate, love
Relation End/ Devoice	relationship, ended, hurt, hate, devoice, blessings, single
Graduation	graduation, school, college, graduate, graduating, year, grad
Admission	admitted, university, admission, accepted, college, offer, school
Exam	passed, exam, test, school, semester, finished, exams, midterms
Research	research, presentation, journalism, paper, conference, go, writing
Essay & Thesis	essay, thesis, reading, statement, dissertation, complete, project
Job	job, accepted, announce, join, joining, offer, starting, announced, work
Interview& Internship	interview, position, accepted, internship, offered, start, work
Moving	house, moving, move, city, home, car, place, apartment, town, leaving
Travel	leave, leaving, flight, home, miss, house, airport, packing, morning
Vacation	vocation, family, trip, country, go, flying, visited, holiday, Hawaii
Winning Award	won, award, support, awards, winning, honor, scholarship, prize
Election/ Promotion/ Nomination	president, elected, run, nominated, named, promotion, cel, selected, business, vote
Publishing	book, sold, writing, finished, read, copy, review, release, books, cover
Contract	signed, contract, deal, agreements, agreed, produce, dollar, meeting
song/ video/ album release	video, song, album, check, show, see, making, radio, love
Acting	play, role, acting, drama, played, series, movie, actor, theater
Death	dies, passed, cancer, family, hospital, dad, grandma, mom, grandpa
Give Birth	baby, born, boy, pregnant, girl, lbs, name, son, world, daughter, birth
Illness	ill, hospital, feeling, sick, cold, flu, getting, fever, doctors, cough
Surgery	surgery, got, test, emergency, blood, tumor, stomachs, hospital, pain, brain
Sports	win, game, team, season, fans, played, winning, football, luck
Running	run, race, finished, race, marathon, ran, miles, running, finish, goal
New Car	car, buy, bought, cars, get, drive, pick, seat, color, dollar, meet
Lost Weight	weight, lost, week, pounds, loss, weeks, gym, exercise, running
Birthday	birthday, come, celebrate, party, friends, dinner, tonight, friend
Lawsuit	sue, sued, file, lawsuit, lawyer, dollars, illegal, court, jury.

Table 3: Example event types with top words discovered by our model.

event examples and trained a 43-class maximum entropy classifier based on the following features:

- **Word**: The sequence of words in the tweet.
- **NER**: Named entity Tag.
- **Dictionary**: Word matching a dictionaries of the top 40 words for each life event category (automatically inferred by the topic model). The feature value is the term's probability generated by correspondent event.
- **Window**: If a dictionary term exists, left and right context words within a window of 3 words and their part-of-speech tags.

Name entity tag is assigned from Ritter et al's Twitter NER system (Ritter et al., 2011). Part-of-Speech tags are assigned based on Twitter POS package (Owoputi et al., 2013) developed by CMU ARK Lab. **Dictionary** and **Window** are constructed based on the topic-term distribution obtained from the previous section.

The average precision and recall are shown in Table 4. And as we can observe, the dictionary (with probability) contributes a lot to the performance and by taking into account a more comprehensive set of information around the key word, classifier on **All** feature setting generate significantly better performance, with 0.382 prevision and 0.48 recall, which is acceptable considering (1) This is is a 43-way classification with much more negative data than positive (2) Some types of events are very close to each other (e.g., Leaving and Vocation). Note that recall is valued more than precision here as false-positive examples will be further screened in self-reported information identification process in the following section.

Feature Setting	Precision	Recall
Word+NER	0.204	0.326
Word+NER+Dictionary	0.362	0.433
All	0.382	0.487

Table 4: Average Performance of Multi-Class Classifier on Different Feature Settings. Negative examples (non important event type) are not considered.

5 Self-Reported Information Identification

Although a message might refer to a topic corresponding to a life event such as marriage, the event still might be one in which the speaker is not directly involved. In this section we describe the self reported event identification portion of our pipeline, which takes output from Section 4 and further identifies whether each tweet refers to an event directly involving the user who publishes it.

Direct labeling of randomly sampled Twitter messages is infeasible for the following reasons: (1) Class imbalance: self-reported events are relatively rare in randomly sampled Twitter messages. (2) A large proportion of self-reported information refers to mundane, everyday topics (e.g., "I just finished dinner!"). Fortunately, many of the tweets retrieved from Section 3 consist of self-reported information and describe major life events. The candidates for annotation are therefore largely narrowed down.

We manually annotated 800 positive examples of self-reported events distributed across the event categories identified in Section 3. We ensured good coverage by first randomly sampling 10 examples from each category, the remainder were sampled from the class distribution in the data. Negative examples of self-reported information consisted of a combination of examples from the original dataset[8] and randomly sampled messages gathered by searching for the top terms in each of the pre-identified topics using the Twitter Search interface [9]. Due to great varieties of negative scenarios, the negative dataset constitutes about 2500 tweets.

5.1 Features

Identifying self-reported tweet requires sophisticated feature engineering. Let u denote the term within the tweet that gets the highest possibility generated by the correspondent topic. We experimented with combinations of the following types of features (results are presented in Table **??**):

- **Bigram**: Bigrams within each tweet (punctuation included).
- **Window**: A window of $k \in \{0, 1, 2\}$ words adjacent to u and their part-of-speech tags.
- **Tense**: A binary feature indicating past tense identified in by the presence of past tense verb (VBD).
- **Factuality**: Factuality denotes whether one expression is presented as corresponding to real situations in the world (Saurí and Pustejovsky, 2007). We use Stanford PragBank[10],

[8] Most tweets in the bootstrapping output are positive.

[9] The majority of results returned by Twitter Search are negative examples.

[10] http://compprag.christopherpotts.net/factbank.html

an extension of FactBank (Saurí and Pustejovsky, 2009) which contains a list of modal words such as "might", "will", "want to" etc[11].

- **I**: Whether the subject of the tweet is first person singular.
- **Dependency**: If the subject is first person singular and the u is a verb, the dependency path between the subject and u (or non-dependency).

Tweet dependency paths were obtained from (Kong et al., 2014). As the tweet parser we use only supports one-to-one dependency path identification but no dependency properties, **Dependency** is a binary feature. The subject of each tweet is determined by the dependency link to the root of the tweet from the parser.

Among the features we explore, **Word** encodes the general information within the tweet. **Window** addresses the information around topic key word. The rest of the features specifically address each of the negative situations described in Challenge 2, Section 1: **Tense** captures past event description, **Factuality** filters out wishes or imagination, **I** and **Dependency** correspond to whether the described event involves the speaker. We built a linear SVM classifier using SVM$_{light}$ package (Joachims, 1999).

5.2 Evaluation

Feature Setting	Acc	Pre	Rec
Bigram+Window	0.76	0.47	0.44
Bigram+Window +Tense+Factuality	0.77	0.47	0.46
all	0.82	0.51	0.48

Table 5: Performance for self-report information identification regarding different feature settings.

We report performance on the task of identifying self-reported information in this subsection. We employ 5-fold cross validation and report Accuracy (Accu), Prevision (Prec) and Recall (Rec) regarding different feature settings. The **Tense**, **Factuality**, **I** and **Dependency** features positively contribute to performance respectively and the best performance is obtained when all types of features are included.

[11]Due to the colloquial property of tweets, we also introduced terms such as "gonna", "wanna", "bona".

precision	recall	F1
0.82	0.86	0.84

Table 7: Performance for identifying properties.

6 Event Property Extraction

Thus far we have described how to automatically identify tweets referring to major life events. In addition, it is desirable to extract important properties of the event, for example the name of the university the speaker was admitted to (See Figure 1). In this section we take a supervised approach to event property extraction, based on manually annotated data for a handfull of the major life event categories automatically identified by our system. While this approach is unlikely to scale to the diversity of important personal events Twitter users are discussing, our experiments demonstrate that event property extraction is indeed feasible.

We cast the problem of event property extraction as a sequence labeling task, using Conditional Random Fields (Lafferty et al., 2001) for learning and inference. To make best use of the labeled data, we trained a unified CRF model for closely related event categories which often share properties; the full list is presented in Table 6 and we labeled 300 tweets in total. Features we used include:

- word token, capitalization, POS
- left and right context words within a window of 3 and the correspondent part-of-speech tags
- word shape, NER
- a gazetteer of universities and employers borrowed from NELL[12].

We use 5-fold cross-validation and report results in Table 7.

7 End-to-End Experiment

The evaluation for each part of our system has been demonstrated in the corresponding section. We now present a real-world evaluation: to what degree can our trained system automatically identify life events in real world.

7.1 Dataset

We constructed a gold-standard life event dataset using annotators from Amazon's Mechanical Turk (Snow et al., 2008) using 2 approaches:

[12]http://rtw.ml.cmu.edu/rtw/kbbrowser/

Life Event	Property
(a) Acceptance, Graduation	Name of University/College
(b) Wedding, Engagement, Falling love	Name of Spouse/ partner/ bf/ gf
(c) Getting a job, interview, internship	Name of Enterprise
(d) Moving to New Places, Trip, Vocation, Leaving	Place, Origin, Destination
(e) Winning Award	Name of Award, Prize

Table 6: Labeling Event Property.

- Ask Twitter users to label their own tweets (Participants include friends, colleagues of the authors and Turkers from Amazon Mechanical Turk[13]).
- Ask Turkers to label other people's tweets.

For option 1, we asked participants to directly label their own published tweets. For option 2, for each tweet, we employed 2 Turkers. Due to the ambiguity in defining life events, the value cohen's kappa[14] as a measure of inter-rater agreement is 0.54; this does not show significant inter-annotator agreement. The authors examined disagreements and also verified all positively labeled tweets. The resulting dataset contains around 900 positive tweets and about 60,000 negative tweets.

To demonstrate the advantage of leveraging large quantities of unlabeled data, the first baseline we investigate is a **Supervised** model which is trained on the manually annotated labeled dataset, and evaluated using 5 fold cross validation. Our **Supervised** baseline consists of a linear SVM classifier using bag of words, NER and POS features. We also tested a second baseline that combines **Supervised** algorithm with an our self-reported information classifier, denoted as **Supervised+Self**.

Results are reported in Table 8; as we can observe, the fully supervised approach is not suitable for this task with only one digit F1 score. The explanations are as follows: (1) the labeled data can only cover a small proportion of life events (2) supervised learning does not separate important event categories and will therefore classify any tweet with highly weighted features (e.g., the mention of "I" or "marriage") as positive. By using an additional self-reported information classifier in **Supervised+Self**, we get a significant boost in precision with a minor recall loss.

[13] https://www.mturk.com/mturk/welcome

[14] http://en.wikipedia.org/wiki/Cohen's_kappa

Approach	Precision	Recall
Our approach	0.62	0.48
Supervised	0.13	0.20
Supervised+Self	0.25	0.18

Table 8: Performance for different approaches for identifying life events in real world.

Approach	Precision	Recall
Step 1	0.65	0.36
Step 2	0.64	0.43
Step 3	0.62	0.48

Table 9: Performance for different steps of bootstrapping for identifying life events in real world.

Another interesting question is to what degree the bootstrapping contributes to the final results. We keep the self-reported information classifier fixed (though it's based the ultimate identified data source), and train the personal event classifier based on topic distributions identified from each of the three steps of bootstrapping[15]. Precision and recall at various stages of bootstrapping are presented in Table 9. As bootstrapping continues, the precision remains roughly constant, but recall increases as more life events and CONGRATULATIONS and CONDOLENCES are discovered.

8 Related Work

Our work is related to three lines of NLP researches. (1) user-level information extraction on social media (2) public event extraction on social media. (3) Data harvesting in Information Extraction, each of which contains large amount of related work, to which we can not do fully justice.

User Information Extraction from Twitter Some early approaches towards understanding user level information on social media is focused on user profile/attribute prediction (e.g.,(Ciot et al., 2013)) user-specific content extraction (Diao

[15] which are 24, 38, 42-class classifiers, where 24, 38, 42 denoted the number of topics discovered in each step of bootstrapping (see Figure 5).

et al., 2012; Diao and Jiang, 2013; Li et al., 2014) or user personalization (Low et al., 2011) identification.

The problem of user life event extraction was first studied by Li and Cardie's (2014). They attempted to construct a chronological timeline for Twitter users from their published tweets based on two criterion: a personal event should be personal and time-specific. Their system does not explicitly identify a global category of life events (and tweets discussing correspondent event) but identifies the topics/events that are personal and time-specific to a given user using an unsupervised approach, which helps them avoids the nuisance of explicit definition for life event characteristics and acquisition of labeled data. However, their system has the short-coming that each personal topic needs to be adequately discussed by the user and their followers in order to be detected[16].

Public Event Extraction from Twitter Twitter serves as a good source for event detection owing to its real time nature and large number of users. These approaches include identifying bursty public topics (e.g.,(Diao et al., 2012)), topic evolution (Becker et al., 2011) or disaster outbreak (Sakaki et al., 2010; Li and Cardie, 2013) by spotting the increase/decrease of word frequency. Some other approaches are focused on generating a structured representation of events (Ritter et al., 2012; Benson et al., 2011).

Data Acquisition in Information Extraction Our work is also related with semi-supervised data harvesting approaches, the key idea of which is that some patterns are learned based on seeds. They are then used to find additional terms, which are subsequently used as new seeds in the patterns to search for additional new patterns (Kozareva and Hovy, 2010b; Davidov et al., 2007; Riloff et al., 1999; Igo and Riloff, 2009; Kozareva et al., 2008). Also related approaches are distant or weakly supervision (Mintz et al., 2009; Craven et al., 1999; Hoffmann et al., 2011) that rely on available structured data sources as a weak source of supervision for pattern extraction from related text corpora.

[16]The reason is that topic models use word frequency for topic modeling.

9 Conclusion and Discussion

In this paper, we propose a pipelined system for major life event extraction from Twitter. Experimental results show that our model is able to extract a wide variety of major life events.

The key strategy adopted in this work is to obtain a relatively clean training dataset from large quantity of Twitter data by relying on minimum efforts of human supervision, and sometimes is at the sacrifice of recall. To achieve this goal, we rely on a couple of restrictions and manual screenings, such as relying on replies, LDA topic identification and seed screening. Each part of system depends on the early steps. For example, topic clustering in Section 3 not only offers training data for event identification in Section 4, but prepares the training data for self-information identification in Section 5. .

We acknowledge that our approach is not perfect due to the following ways: (1) The system is only capable of discovering a few categories of life events with many others left unidentified. (2) Each step of the system will induce errors and negatively affected the following parts. (3) Some parts of evaluations are not comprehensive due to the lack of gold-standard data. (4) Among all pipelines, event property identification in Section 6 still requires full supervision in CRF model, making it hard to scale to every event type[17]. How to address these aspects and generate a more accurate, comprehensive and fine-grained life event list for Twitter users constitute our further work.

Acknowledgements

A special thanks is owned to Myle Ott for suggestions on bootstrapping procedure in data harvesting. The authors want to thank Noah Smith, Chris Dyer and Alok Kothari for useful comments, discussions and suggestions regarding different steps of the system and evaluations. We thank Lingpeng Kong and members of Noah's ARK group at CMU for providing the tweet dependency parser. All data used in this work is extracted from CMU Twitter Warehouse maintained by Brendan O'Connor, to whom we want to express our gratitude.

[17]We view weakly supervised life event property extraction as an interesting direction for future work.

References

Hila Becker, Mor Naaman, and Luis Gravano. 2011. Beyond trending topics: Real-world event identification on twitter. *ICWSM*, 11:438–441.

Cosmin Adrian Bejan, Matthew Titsworth, Andrew Hickl, and Sanda M Harabagiu. 2009. Nonparametric bayesian models for unsupervised event coreference resolution. In *NIPS*, pages 73–81.

Edward Benson, Aria Haghighi, and Regina Barzilay. 2011. Event discovery in social media feeds. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 389–398. Association for Computational Linguistics.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Morgane Ciot, Morgan Sonderegger, and Derek Ruths. 2013. Gender inference of twitter users in non-english contexts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, Wash*, pages 18–21.

Mark Craven, Johan Kumlien, et al. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999, pages 77–86.

Dmitry Davidov, Ari Rappoport, and Moshe Koppel. 2007. Fully unsupervised discovery of concept-specific relationships by web mining. In *Annual Meeting-Association For Computational Linguistics*, volume 45, page 232.

Qiming Diao and Jing Jiang. 2013. A unified model for topics, events and users on twitter. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1869–1879.

Qiming Diao, Jing Jiang, Feida Zhu, and Ee-Peng Lim. 2012. Finding bursty topics from microblogs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 536–544. Association for Computational Linguistics.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235.

Amit Gruber, Yair Weiss, and Michal Rosen-Zvi. 2007. Hidden topic markov models. In *International Conference on Artificial Intelligence and Statistics*, pages 163–170.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Sean P Igo and Ellen Riloff. 2009. Corpus-based semantic lexicon induction with web-based corroboration. In *Proceedings of the Workshop on Unsupervised and Minimally Supervised Learning of Lexical Semantics*, pages 18–26. Association for Computational Linguistics.

Thorsten Joachims. 1999. Making large scale svm learning practical.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah Smith. 2014. A dependency parser for tweets. In *EMNLP*.

Zornitsa Kozareva and Eduard Hovy. 2010a. Learning arguments and supertypes of semantic relations using recursive patterns. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1482–1491. Association for Computational Linguistics.

Zornitsa Kozareva and Eduard Hovy. 2010b. Not all seeds are equal: Measuring the quality of text mining seeds. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 618–626. Association for Computational Linguistics.

Zornitsa Kozareva, Ellen Riloff, and Eduard H Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *ACL*, volume 8, pages 1048–1056.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Jiwei Li and Claire Cardie. 2013. Early stage influenza detection from twitter. *arXiv preprint arXiv:1309.7340*.

Jiwei Li and Claire Cardie. 2014. Timeline generation: Tracking individuals on twitter. *WWW, 2014*.

Jiwei Li, Alan Ritter, and Eduard Hovy. 2014. Weakly supervised user profile extraction from twitter. ACL.

Yucheng Low, Deepak Agarwal, and Alexander J Smola. 2011. Multiple domain user personalization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 123–131. ACM.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In

Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 262–272. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 100–108. Association for Computational Linguistics.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL-HLT*, pages 380–390.

Ellen Riloff, Rosie Jones, et al. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI*, pages 474–479.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Kirk Roberts and Sanda M Harabagiu. 2011. Unsupervised learning of selectional restrictions and detection of argument coercions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 980–990. Association for Computational Linguistics.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM.

Roser Saurí and James Pustejovsky. 2007. Determining modality and factuality for text entailment. In *Semantic Computing, 2007. ICSC 2007. International Conference on*, pages 509–516. IEEE.

Roser Saurí and James Pustejovsky. 2009. Factbank: A corpus annotated with event factuality. *Language resources and evaluation*, 43(3):227–268.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1456–1466. Association for Computational Linguistics.

Brighter than Gold: Figurative Language in User Generated Comparisons

Vlad Niculae and **Cristian Danescu-Niculescu-Mizil**
MPI-SWS
Cornell University
vniculae@mpi-sws.org, cristian@mpi-sws.org

Abstract

Comparisons are common linguistic devices used to indicate the likeness of two things. Often, this likeness is not meant in the literal sense—for example, "I slept like a log" does not imply that logs actually sleep. In this paper we propose a computational study of figurative comparisons, or *similes*. Our starting point is a new large dataset of comparisons extracted from product reviews and annotated for figurativeness. We use this dataset to characterize figurative language in naturally occurring comparisons and reveal linguistic patterns indicative of this phenomenon. We operationalize these insights and apply them to a new task with high relevance to text understanding: distinguishing between figurative and literal comparisons. Finally, we apply this framework to explore the social context in which figurative language is produced, showing that similes are more likely to accompany opinions showing extreme sentiment, and that they are uncommon in reviews deemed helpful.

1 Introduction

> *In argument similes are like songs in love; they describe much, but prove nothing.*
>
> — Franz Kafka

Comparisons are fundamental linguistic devices that express the likeness of two things—be it entities, concepts or ideas. Given that their working principle is to emphasize the relation between the shared properties of two arguments (Bredin, 1998), comparisons can synthesize important semantic knowledge.

Often, comparisons are not meant to be understood literally. Figurative comparisons are an important figure of speech called *simile*. Consider the following two examples paraphrased from Amazon product reviews:

(1) Sterling is much cheaper than gold.

(2) Her voice makes this song shine brighter than gold.

In (1) the comparison draws on the relation between the price property shared by the two metals, *sterling* and *gold*. While (2) also draws on a common property (*brightness*), the polysemantic use (vocal timbre vs. light reflection) makes the comparison figurative.

Importantly, there is no general rule separating literal from figurative comparisons. More generally, the distinction between figurative and literal language is blurred and subjective (Hanks, 2006). Multiple criteria for delimiting the two have been proposed in the linguistic and philosophical literature—for a comprehensive review, see Shutova (2010)—but they are not without exceptions, and are often hard to operationalize in a computational framework. When considering the specific case of comparisons, such criteria cannot be directly applied.

Recently, the simile has received increasing attention from linguists and lexicographers (Moon, 2008; Moon, 2011; Hanks, 2013) as it became clearer that similes need to be treated separately from metaphors since they operate on fundamentally different principles (Bethlehem, 1996). Metaphors are linguistically simple structures hiding a complex mapping between two domains, through which many properties are transferred. For example the conceptual metaphor of *life as a journey* can be instantiated in many particular ways: *being at a fork in the road*, *reaching the end of the line* (Lakoff and Johnson, 1980). In contrast, the semantic context of similes tends to be very shallow, transferring a single property (Hanks, 2013). Their more explicit syntactic structure allows, in exchange, for more lexical creativity. As Hanks (2013) puts it, similes "tend to license all

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2008–2018,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

sorts of logical mayhem." Moreover, the overlap between the expressive range of similes and metaphors is now known to be only partial: there are similes that cannot be rephrased as metaphors, and the other way around (Israel et al., 2004). This suggests that figurativeness in similes should be modeled differently than in metaphors. To further underline the necessity of a computational model for similes, we give the first estimate of their frequency in the wild: over 30% of comparisons are figurative.[1] We also confirm that a state of the art metaphor detection system performs poorly when applied directly to the task of detecting similes.

In this work we propose a computational study of figurative language in comparisons. To this end, we build the first large collection of naturally occurring comparisons with figurativeness annotation, which we make publicly available. Using this resource we explore the linguistic patterns that characterize similes, and group them in two conceptually distinctive classes. The first class contains cues that are agnostic of the context in which the comparison appears (domain-agnostic cues). For example, we find that the higher the semantic similarity between the two arguments, the less likely it is for the comparison to be figurative—in the examples above, *sterling* is semantically very similar to *gold*, both being metals, but *song* and *gold* are semantically dissimilar. The second type of cues are domain-specific, drawing on the intuition that the domain in which a comparison is used is a factor in determining its figurativeness. We find, for instance, that the less specific a comparison is to the domain in which it appears, the more likely it is to be used in a figurative sense (e.g., in example (2), *gold* is very unexpected in the musical domain).

We successfully exploit these insights in a new prediction task relevant to text understanding: discriminating figurative comparisons from literal ones. Encouraged by the high accuracy of our system—which is within 10% of that obtained by human annotators—we automatically extend the figurativeness labels to 80,000 comparisons occurring in product reviews. This enables us to conduct a fine-grained analysis of how comparison usage interacts with their social context, opening up a research direction with applications in sentiment analysis and opinion mining. In particular we find that figurative comparisons are more likely to accompany reviews showing extreme sentiment, and that they are uncommon in opinions deemed as being helpful. To the best of our knowledge, this is the first time figurative language is tied to the social context in which it appears.

To summarize, the main contributions of this work are as follows:

- it introduces the first large dataset of comparisons with figurativeness annotations (Section 3);
- it unveils new linguistic patterns characterizing figurative comparisons (Section 4);
- it introduces the task of distinguishing figurative from literal comparisons (Section 5);
- it establishes the relation between figurative language and the social context in which it appears (Section 6).

2 Further Related Work

Corpus studies on figurative language in comparisons are scarce, and none directly address the distinction between figurative and literal comparisons. Roncero et al. (2006) observed, by searching the web for several stereotypical comparisons (e.g., *education is like a stairway*), that similes are more likely to be accompanied by explanations than equivalent metaphors (e.g., *education is a stairway*). Related to figurativeness is irony, which Veale (2012a) finds to often be lexically marked. By using a similar insight to filter out ironic comparisons, and by assuming that the rest are literal, Veale and Hao (2008) learn stereotypical knowledge about the world from frequently compared terms. A similar process has been applied to both English and Chinese by Li et al. (2012), thereby encouraging the idea that the trope behaves similarly in different languages. A related system is the *Jigsaw Bard* (Veale and Hao, 2011), a thesaurus driven by figurative conventional similes extracted from the Google N-grams. This system aims to build and generate canned expressions by using items frequently associated with the simile pattern above. An extension of the principles of the *Jigsaw Bard* is found in *Thesaurus Rex* (Veale and Li, 2013), a data-driven partition of words into ad-hoc categories. *Thesaurus Rex* is constructed using simple comparison and hypernym patterns

[1] This estimate is based on the set of noun-noun comparisons with non-identical arguments collected for this study from Amazon.com product reviews.

and is able to provide weighted lists of categories for given words.

In text understanding systems, literal comparisons are used to detect analogies between related geographical places (Lofi et al., 2014). Tandon et al. (2014) use relative comparative patterns (e.g., *X is heavier than Y*) to enrich a common-sense knowledge base. Jindal and Liu (2006) extract graded comparisons from various sources, with the objective of mining consumer opinion about products. They note that identifying objective vs. subjective comparisons—related to literality—is an important future direction. Given that many comparisons are figurative, a system that discriminates literal from figurative comparisons is essential for such text understanding and information retrieval systems.

The vast majority of previous work on figurative language focused on metaphor detection. Tsvetkov et al. (2014a) propose a cross-lingual system based on word-level conceptual features and they evaluate it on Subject-Verb-Object triples and Adjective-Noun pairs. Their features include and extend the idea of abstractness used by Turney et al. (2011) for Adjective-Noun metaphors. Hovy et al. (2013) contribute an unrestricted metaphor corpus and propose a method based on tree kernels. Bridging the gap between metaphor identification and interpretation, Shutova and Sun (2013) proposed an unsupervised system to learn source-target domain mappings. The system fits conceptual metaphor theory (Lakoff and Johnson, 1980) well, at the cost of not being able to tackle figurative language in general, and similes in particular, as similes do not map entire domains to one another. Since similes operate on fundamentally different principles than metaphors, our work proposes a computational approach tailored specifically for comparisons.

3 Background and Data

3.1 Structure of a comparison

Unlike metaphors, which are generally unrestricted, comparisons are more structured but also more lexically and semantically varied. This enables a more structured computational representation of which we take advantage. The constituents of a comparison according to Hanks (2012) are:

- the TOPIC, sometimes called tenor: it is usually a noun phrase and acts as logical subject;
- the VEHICLE: it is the object of the comparison and is also usually a noun phrase;
- the shared PROPERTY or ground: it expresses what the two entities have in common—it can be explicit but is often implicit, left for the reader to infer;
- the EVENT (eventuality or state): usually a verb, it sets the frame for the observation of the common property;
- the COMPARATOR: commonly a preposition (*like*) or part of an adjectival phrase (*better than*), it is the trigger word or phrase that marks the presence of a comparison.

The literal example (1) would be segmented as:

> [Sterling /TOPIC] [is /EVENT] much [cheaper /PROPERTY] [than /COMPARATOR] [gold /VEHICLE]

3.2 Annotation

People resort to comparisons often when making descriptions, as they are a powerful way of expressing properties by example. For this reason we collect a dataset of user-generated comparisons in Amazon product reviews (McAuley and Leskovec, 2013), where users have to be descriptive and precise, but also to express personal opinion. We supplement the data with a smaller set of comparisons from WaCky and WaCkypedia (Baroni et al., 2009) to cover more genres. In preliminary work, we experimented with dependency parse tree patterns for extracting comparisons and labeling their parts (Niculae, 2013). We use the same approach, but with an improved set of patterns, to extract comparisons with the COMPARATORS *like*, *as* and *than*.[2] We keep only the matches where the TOPIC and the VEHICLE are nouns, and the PROPERTY, if present, is an adjective, which is the typical case. Also, the head words of the constituents are constrained to occur in the distributional resources used (Baroni and Lenci, 2010; Faruqui and Dyer, 2014).[3]

[2] We process the review corpus with part-of-speech tagging using the IRC model for *TweetNLP* (Owoputi et al., 2013; Forsyth and Martell, 2007) and dependency parsing using the *TurboParser* standard model (Martins et al., 2010).

[3] Due to the strong tendency of comparisons with the same TOPIC and VEHICLE to be trivially literal in the WaCky examples, we filtered out such examples from the Amazon product reviews. We also filtered proper nouns using a capitalization heuristic.

We proceed to validate and annotate for figurativeness a random sample of the comparisons extracted using the automated process described above. The annotation is performed using crowdsourcing on the Amazon Mechanical Turk platform, in two steps. First, the annotators are asked to determine whether a displayed sentence is indeed a comparison between the highlighted words (TOPIC and VEHICLE). Sentences qualified by two out of three annotators as comparisons are used in the second round, where the task is to rate how metaphorical a comparison is. We use a scale of 1 to 4 following Turney et al. (2011), and then binarize to consider scores of 1–2 as literal and 3–4 as figurative. Finally, in this work we only consider comparisons where all three annotators agree on this binary notion of figurativeness. For both tasks, we provide guidelines mostly in the form of examples and intuition, motivated on one hand by the annotators not having specialized knowledge, and on the other hand by the observation that the literal-figurative distinction is subjective. All annotators have the *master worker* qualification, reside in the U.S. and completed a linguistic background questionnaire that verifies their experience with English. In both tasks, control sentences with confidently known labels are used to filter low quality answers; in addition, we test annotators with a simple paraphrasing task shown to be effective for eliciting and verifying linguistic attention (Munro et al., 2010). Both tasks seem relatively difficult for humans, with inter-annotator agreement given by Fleiss' k of 0.48 for the comparison identification task and of 0.54 for the figurativeness annotation after binarization. This is comparable to 0.57 reported by Hovy et al. (2013) for general metaphor labeling. We show some statistics about the collected data in Table 1. Overall, this is a costly process: out of 2400 automatically extracted comparison candidates, about 60% were deemed by the annotators to be actual comparisons and only 12% end up being selected confidently enough as figurative comparisons.

Our dataset of human-filtered comparisons, with the scores given by the three annotators, is made publicly available to encourage further work.[4] This also includes about 400 comparisons where the annotators do not agree perfectly on binary figurativeness. Such cases can be interesting to other analyses, even if we don't consider them in our experiments. It is worth noting that the existing corpora annotated for metaphor cannot be directly used to study comparisons. For example, in TroFi (Birke and Sarkar, 2006), a corpus of 6436 sentences annotated for figurativeness, we only find 42 noun-noun comparisons with sentence-level (thus noisy) figurativeness labels.

Domain	fig.	lit.	% fig.
Books	177	313	36%
Music	45	68	40%
Electronics	23	105	18%
Jewelery	9	126	7%
WaCky	19	79	19%
Total	273	609	31%

Table 1: Figurativeness annotation results. Only comparisons where all three annotators agree are considered.

4 Linguistic Insights

We now proceed to exploring the linguistic patterns that discriminate figurative from literal comparisons. We consider two broad classes of cues, which we discuss next.

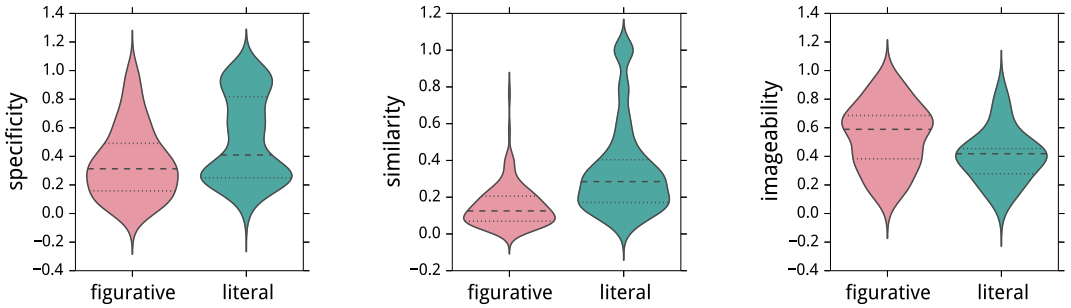
4.1 Domain-specific cues

Figurative language is often used for striking effects, and comparisons are used to describe new things in terms of something given (Hanks, 2013). Since the norms that define what is surprising and what is well-known vary across domains, we expect that such contextual information should play an important role in figurative language detection. This is a previously unexplored dimension of figurative language, and Amazon product reviews offer a convenient testbed for this intuition since category information is provided.

Specificity To estimate whether a comparison can be considered striking in a particular domain—whether it references images or ideas that are unexpected in its context—we employ a simple measure of word *specificity* with respect to a domain: the ratio of the word frequency within the domain and the word frequency in all domains being considered.[5] It should be noted that specificity is not purely a function of the word, but

[4] http://vene.ro/figurative-comparisons/

[5] We measure specificity for the VEHICLE, PROPERTY and EVENT.



(a) VEHICLE specificity. (b) TOPIC-VEHICLE similarity. (c) Imageability of the PROPERTY.

Figure 1: Distribution of some of the features we use, across literal and figurative comparisons in the test set. The profile of the plot is a kernel density estimation of the distribution, and the markers indicate the median and the first and third quartiles.

of the word and the context in which it appears. A comparison in the *music* domain that involves melodies is not surprising:

> But the title song really feels like a pretty bland vocal **melody** [...]

But the same word can play a very different role in another context, for example, *book* reviews:

> Her books are like sweet **melodies** that flow through your head.

Indeed, the word *melody* has a specificity of 96% in the *music* domain and only of 3% in the *books* domain.

An analysis on the labeled data confirms that literal comparisons do indeed tend to have more domain-specific VEHICLES (Mann-Whitney U test, $p < 0.01$) than figurative ones. Furthermore, the distribution of specificity across both types of comparisons, as shown in Figure 1a, has the appearance of a mixture model of general and specific words. Figurative comparison VEHICLES largely exhibit only the general component of the mixture.[6]

Domain label An analysis of the annotation results reveals that the percentage of comparisons that are figurative differs widely across domains, as indicated in the last column in Table 1. This suggests that simply knowing the domain of a text can serve to adjust some prior expectation about figurative language presence and therefore improve detection. We test this hypothesis using a Z-test comparing all Amazon categories. With the exception of *books* and *music* reviews, that have similar ratios, all other pairs of categories show significantly different proportions of figurative comparisons ($p < 0.01$).

4.2 Domain-agnostic cues

Linguistic studies of figurative language suggest that there is a fundamental generic notion of figurativeness. We attempt to capture this notion in the context of comparisons using syntactic and semantic information.

Topic-Vehicle similarity The default role of literal comparisons is to assert similarity of things. Therefore, we expect that a high semantic similarity between the TOPIC and the VEHICLE of a comparison is a sign of literal usage, as we previously hypothesized in preliminary work (Niculae, 2013). To test this hypothesis, we compute TOPIC-VEHICLE similarity using Distributional Memory (Baroni and Lenci, 2010), a freely available distributional semantics resource that captures word relationships through grammatical role co-occurrence.

By applying this measure to our data, we find that there is indeed an important difference between the distributions of TOPIC-VEHICLE similarity in figurative and literal comparisons (shown in Figure 1b); the means of the two distributions are significantly different (Mann-Whitney $p < 0.01$).

Metaphor-inspired features We also seek to understand to what extent insights provided by computational work on metaphor detection can be

[6]The mass around 0.25 in Figure 1a is largely explained by generic words such as *thing, others, nothing, average* and barely specific words like *veil, reputation, dream, garbage*.

	more concrete	less concrete
more imageable	*cinnamon, kiss*	*devil, happiness*
less imageable	*casque, pugilist*	*aspect, however*

Table 2: Examples of words with high and low concreteness and imageability scores from the MRC Psycholinguistic Database.

applied in the context of comparisons. To that end we consider features shown to provide state of the art performance in the task of metaphor detection (Tsvetkov et al., 2014a): abstractness, imageability and supersenses.

Abstractness and imageability features are derived from the MRC Psycholinguistic Database (Coltheart, 1981), a dictionary based on manually annotated datasets of psycholinguistic norms. Imageability is the property of a word to arouse a mental image, be it in the form of a mental picture, sound or any other sense. Concreteness is defined as "any word that refers to objects, materials or persons," while abstractness, at the other end of the spectrum, is represented by words that cannot be usually experienced by the senses (Paivio et al., 1968). Table 2 shows a few examples of words with high and low concreteness and imageability scores. Supersenses are a very coarse form of meaning representation. Tsvetkov et al. (2014a) used WordNet (Miller, 1995) semantic classes for nouns and verbs, for example *noun.body*, *noun.animal*, *verb.consumption*, or *verb.motion*. For adjectives, Tsvetkov et al. (2014b) developed and made available a novel classification in the same spirit.[7] We compute abstractness, imageability and supersenses for the TOPIC, VEHICLE, EVENT, and PROPERTY.[8] We concatenate these features with the raw vector representations of the constituents, following Tsvetkov et al. (2014a).

We find that such features relate to figurative comparisons in a meaningful way. For example, out of all comparisons with explicit properties, figurative comparisons tend to have properties that are more imageable (Mann-Whitney $p < 0.01$), as illustrated by Figure 1c. This is in agreement with Hanks (2005), who observed that similes are characterized by their appeal to sensory imagination.

Definiteness We introduce another simple but effective syntactic cue that relates to concreteness: the presence of a definite article versus an indefinite one (or none at all). We search for the indefinite articles *a* and *an* and the definite article *the* in each component of a comparison.

We find that similes tend to have indefinite articles in the VEHICLE more often and definite articles less often (Mann-Whitney $p < 0.01$). In particular, 59% of comparisons where the VEHICLE has a indefinite article are figurative, as opposed to 13% of the comparisons where VEHICLE has a definite article.

5 Prediction Task

We now turn to the task of predicting whether a comparison is figurative or literal. Not only does this task allow us to assess and compare the efficiency of the linguistic cues we discussed, but it is also highly relevant in the context of natural language understanding systems.

We conduct a logistic regression analysis, and compare the efficiency of the features derived from our analysis to a **bag of words** baseline. In addition to the features inspired by the previously described linguistic insights, we also try to computationally capture the lexical usage patterns of comparisons using a version of bag of words adapted to the comparison structure. In this **slotted bag of words** system, features correspond to occurrence of words within constituents (e.g., *bright* $\in$ PROPERTY).

We perform a stratified split of our comparison dataset into equal train and test sets (each set containing 408 comparisons, out of which 134 are figurative),[9] and use a 5-fold stratified cross validation over the training set to choose the optimal value for the logistic regression regularization parameter and the type of regularization (ℓ_1 or ℓ_2) for each feature set.[10]

[7] Following Tsvetkov et al. (2014a) we train a classifier to predict these features from a vector space representation of a word. We use the same cross-lingually optimized representation from Faruqui and Dyer (2014) and a simpler classifier, a logistic regression, which we find to perform as well as the random forests used in Tsvetkov et al. (2014a). We treat supersense prediction as a multi-label problem and apply a one-versus-all transformation, effectively learning a linear classifier for each supersense.

[8] If the PROPERTY is implicit, all corresponding features are set to zero. An extra binary feature indicates whether the PROPERTY is explicit or implicit.

[9] The entire analysis described in Section 4 is only conducted on the training set. Also, in order to ensure that we are assessing the performance of the classifier on unseen comparisons, we discard from our dataset all those with the same TOPIC and VEHICLE pair.

[10] We use the logistic regression implementation of `liblinear` (Fan et al., 2008) wrapped by the `scikit-learn` library (Pedregosa et al., 2011).

Model	# features	Acc.	P	R	F_1	AUC
Bag of words	1970	0.79	0.63	0.84	0.72	0.87
Slotted bag of words	1840	0.80	0.64	0.90	0.75	0.89
Domain-agnostic cues	357	0.81	0.70	0.74	0.72	0.90
only metaphor inspired	345	0.75	0.60	0.72	0.65	0.84
Domain-specific cues	8	0.69	0.51	0.81	0.63	0.76
All linguistic insight cues	365	0.86	0.76	0.83	0.79	0.92
Full	2202	0.88	0.80	0.84	0.82	0.94
Human	-	0.96	0.92	0.96	0.94	-

Table 3: Classification performance on the test set for the different sets of features we considered; human performance is shown for reference.

Classifier performance The performance on the classification task is summarized in Table 3. We note that the **bag of words** baseline is remarkably strong, because of common idiomatic similes that can be captured through keywords. Our **full** system (which relies on our linguistically inspired cues discussed in Section 4 in addition to slotted bag of words) significantly outperforms the bag of words baseline and the slotted bag of words system in terms of accuracy, F_1 score and AUC ($p < 0.05$),[11] suggesting that linguistic insights complement idiomatic simile matching. Importantly, a system using only our **linguistic insight cues** also significantly improves over the baseline in terms of accuracy and AUC and it is not significantly different from the full system in terms of performance, in spite of having about an order of magnitude fewer features. It is also worth noting that the **domain-specific cues** play an important role in bringing the performance to this level by capturing a different aspect of what it means for a comparison to be figurative.

The features used by the state of the art metaphor detection system of Tsvetkov et al. (2014a), adapted to the comparison structure, perform poorly by themselves and do not improve significantly over the baseline. This is consistent with the theoretical motivation that figurativeness in comparisons requires special computational treatment, as discussed in Section 1. Furthermore, the linguistic insight features not only significantly outperform the metaphor inspired features ($p < 0.05$), but are also better at exploiting larger amounts of data, as shown in Figure 2.

[11] All statistical significance results in this paragraph are obtained from 5000 bootstrap samples.

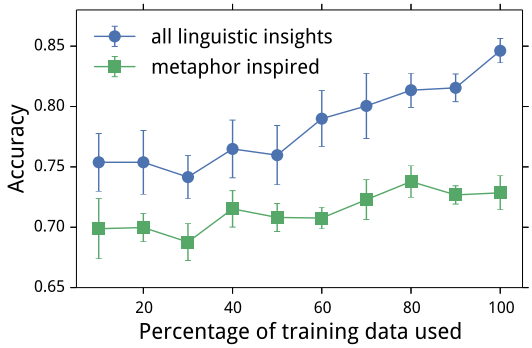


Figure 2: Learning curves. Each point is obtained by fitting a model on 10 random subsets of the training set. Error bars show 95% confidence intervals.

Comparison to human performance To gauge how well humans would perform at the classification task on the actual test data, we perform another Amazon Mechanical Turk evaluation on 140 examples from the test set. For the evaluation, we use majority voting between the three annotators,[12] and compare to the agreed labels in the dataset. Estimated human accuracy is 96%, placing our full system within 10% of human accuracy.

Feature analysis The predictive analysis we perform allows us to investigate to what extent the features inspired by our linguistic insights have discriminative power, and whether they actually cover different aspects of figurativeness.

[12] Majority voting helps account for the noise inherent to crowdsourced annotation, which is less accurate than professional annotation. Taking the less optimistic invididual turker answers, human performance is on the same level as our full system.

Feature	Coef.	Example where the feature is positively activated
TOPIC-VEHICLE similarity	−11.3	the older *man* was wiser and stronger than the *boy*
VEHICLE specificity	−5.8	the cord is more durable than the *adapter* [Electronics]
VEHICLE imageability	4.9	the explanations are as clear as *mud*
VEHICLE communication supersense	−4.6	the book reads like six short *articles*
VEHICLE indefiniteness	4.0	his fame drew foreigners to him like *a* magnet
life ∈ VEHICLE	7.1	the hero is truly larger than *life*: godlike, yet flawed
picture ∈ VEHICLE	−6.0	the necklace looks just like the *picture*
other ∈ VEHICLE	−5.9	this one is just as nice as the *other*
others ∈ VEHICLE	−5.5	some songs are more memorable than *others*
crap ∈ VEHICLE	4.7	the headphones sounded like *crap*

Table 4: Top 5 linguistic insight features (top) and slotted bag of words features (bottom) in the full model and their logistic regression coefficients. A positive coefficient means the feature indicates figurativeness.

Table 4 shows the best linguistic insight and slotted bag of words features selected by the full model. The strongest feature by far is the semantic similarity between the TOPIC and the VEHICLE. By itself, this feature gets 70% accuracy and 61% F_1 score.

The rest of the top features involve mostly the VEHICLE. This suggests that the VEHICLE is the most informative element of a comparison when it comes to figurativeness. Features involving other constituents also get selected, but with slightly lower weights, not making it to the top.

VEHICLE specificity is one of the strongest features, with positive values indicating literal comparisons. This confirms our intuition that domain information is important to discriminate figurative from literal language.

Of the adapted metaphor features, the noun communication supersense and the imageability of the VEHICLE make it to the top. Nouns with low communication rating occurring in the training set include *puddles, arrangements, carbohydrates* while nouns with high communication rating include *languages* and *subjects*.

Presence of an indefinite article in the VEHICLE is a strong indicator of figurativeness. By themselves, the definiteness and indefiniteness features perform quite well, attaining 78% accuracy and 67% F_1 score.

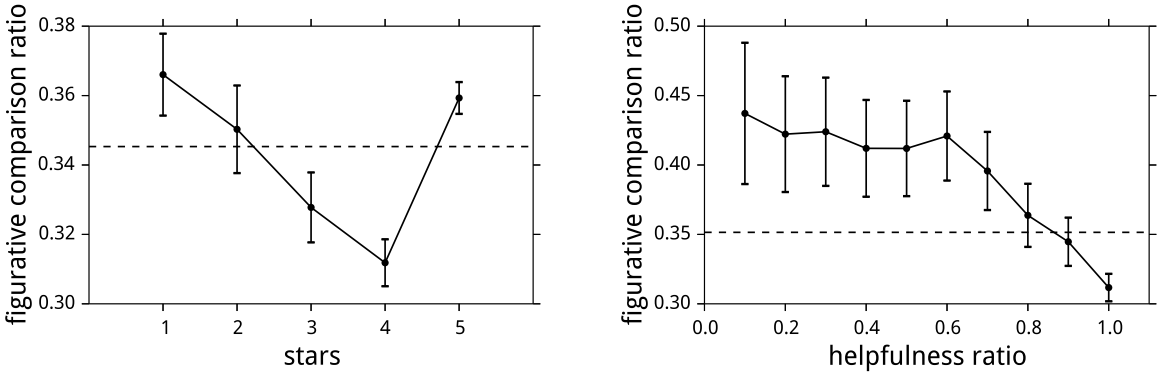
The salient bag of words features correspond to specific types of comparisons. The words *other* and *others* in the VEHICLE indicate comparisons between the same kind of arguments, for example *some songs are more memorable than others,* and these are likely to be literal. The word *picture* is specific to the review setting, as products are accompanied by photos, and for certain kinds of products, the resemblance of the product with the image is an important factor for potential buyers.[13] The bag of words systems are furthermore able to learn idiomatic comparisons by identifying common figurative VEHICLES such as *life* and *crap*, corresponding to fixed expressions such as *larger than life*.

Error analysis Many of the errors made by our full system involve indirect semantic mechanisms such as metonymy. For example, the false positive *the typeface was larger than most books* really means *larger than the typefaces found in most books*, but without the implicit expansion the meaning can appear figurative. A similar kind of ellipsis makes the example *a lot [of songs] are even better than sugar* be wrongly classified as literal. Another source of error is polysemy. Examples like *the rejuvelac formula is about 10 times better than yogurt* are misclassified because of the multiple meanings of the word *formula*, one being closely related to yogurt and food, but the more common ones being general and abstract, suggesting figurativeness.

6 Social Correlates

The advantage of studying comparisons situated in a social context is that we can understand how their usage interacts with internal and external human factors. An internal factor is the sentiment of

[13] This feature is highly correlated with the domain: it appears 25 times in the training set, 24 of which in the *jewelery* domain and once in *book* reviews.



(a) Figurative comparisons are more likely to be found in reviews with strongly polarized sentiment.

(b) Helpful reviews are less likely to contain figurative comparisons.

Figure 3: Interaction between figurative language and social context aspects. Error bars show 95% confidence intervals. The dashed horizontal line marks the average proportion of figurative comparisons. In Figure 3b the average proportion is different because we only consider reviews rated by at least 10 readers.

the user towards the reviewed product, indicated by the star rating of the review. An external factor present in the data is how helpful the review is perceived by other users. In this section we analyze how these factors interact with figurative language in comparisons.

To gain insight about fine grained interactions with human factors at larger scale, we use our classifier to find over 80,000 figurative and literal comparisons from the same four categories. The trends we reveal also hold significantly on the manually annotated data.

Sentiment While it was previously noted that similes often transmit strong affect (Hanks, 2005; Veale, 2012a; Veale, 2012b), the connection between figurativeness and sentiment was never empirically validated. The setting of product reviews is convenient for investigating this issue, since the star ratings associated with the reviews can be used as sentiment labels. We find that comparisons are indeed significantly more likely to be figurative when the users express strong opinions, i.e., in one-star or five-star reviews (Mann-Whitney $p < 0.02$ on the manually annotated data). Figure 3a shows how the proportion of figurative comparisons varies with the polarity of the review.

Helpfulness It is also interesting to understand to what extent figurative language relates to the external perception of the content in which it appears. We find that comparisons in helpful reviews[14] are less likely to be figurative. Figure 3b shows a near-constant high ratio of figurative comparisons among unhelpful and average reviews; as helpfulness increases, figurative comparisons become less frequent. We further validate that this effect is not a confound of the distribution of helpfulness ratings across reviews of different polarity by controlling for the star rating: given a fixed star rating, the proportion of figurative comparisons is still lower in helpful (helpfulness over 50%) than in unhelpful (helpfulness under 50%) reviews; this difference is significant (Mann-Whitney $p < 0.01$) for all classes of ratings except one-star. The size of the manually annotated data does not allow for star rating stratification, but the overall difference is statistically significant (Mann-Whitney $p < 0.01$). This result encourages further experimentation to determine whether there is a causal link between the use of figurative language in user generated content and its external perception.

7 Conclusions and Future Work

This work proposes a computational study of figurative language in comparisons. Starting from a new dataset of naturally occurring comparisons with figurativeness annotation (which we make publicly available) we explore linguistic patterns that are indicative of similes. We show that these

[14] In order to have reliable helpfulness scores, we only consider reviews that have been rated by at least by ten readers.

insights can be successfully operationalized in a new prediction task: distinguishing literal from figurative comparisons. Our system reaches accuracy that is within 10% of human performance, and is outperforming a state of the art metaphor detection system, thus confirming the need for a computational approach tailored specifically to comparisons. While we take a data-driven approach, our annotated dataset can be useful for more theoretical studies of the kinds of comparisons and similes people use.

We discover that domain knowledge is an important factor in identifying similes. This suggests that future work on automatic detection of figurative language should consider contextual parameters such as the topic and community where the content appears.

Furthermore, we are the first to tie figurative language to the social context in which it is produced and show its relation to internal and external human factors such as opinion sentiment and helpfulness. Future investigation into the causal effects of these interactions could lead to a better understanding of the role of figurative language in persuasion and rhetorics.

In our work, we consider common noun TOPICS and VEHICLES and adjectival PROPERTIES. This is the most typical case, but supporting other parts of speech—such as proper nouns, pronouns, and adverbs—can make a difference in many applications. Capturing compositional interaction between the parts of the comparison could lead to more flexible models that give less weight to the VEHICLE.

This study is also the first to estimate how prevalent similes are in the wild, and reports that about one third of the comparisons we consider are figurative. This is suggestive of the need to build systems that can properly process figurative comparisons in order to correctly harness the semantic information encapsulated in comparisons.

Acknowledgements

We would like to thank Yulia Tsvetkov for constructive discussion about figurative language and about her and her co-authors' work. We are grateful for the suggestions of Patrick Hanks, Constantin Orăsan, Sylviane Cardey, Izabella Thomas, Ekaterina Shutova, Tony Veale, and Niket Tandon. We extend our gratitude to Julian McAuley for preparing and sharing the Amazon review dataset. We are thankful to the anonymous reviewers, whose comments were *like a breath of fresh air*. We acknowledge the help of the Amazon Mechanical Turk annotators and of the MPI-SWS students involved in pilot experiments.

Vlad Niculae was supported in part by the European Commission, Education & Training, Erasmus Mundus: EMMC 2008-0083, Erasmus Mundus Masters in NLP & HLT.

References

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

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Louise Shabat Bethlehem. 1996. Simile and figurative language. *Poetics Today*, 17(2):203–240.

Julia Birke and Anoop Sarkar. 2006. A clustering approach for nearly unsupervised recognition of nonliteral language. In *Proceedings of EACL*.

Hugh Bredin. 1998. Comparisons and similes. *Lingua*, 105(1):67–78.

Max Coltheart. 1981. The MRC psycholinguistic database. *The Quarterly Journal of Experimental Psychology*, 33(4):497–505.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*.

Eric N Forsyth and Craig H Martell. 2007. Lexical and discourse analysis of online chat dialog. In *Proceedings of ICSC*.

Patrick Hanks. 2005. Similes and sets: The English preposition 'like'. In R. Blatná and V. Petkevic, editors, *Languages and Linguistics: Festschrift for Fr. Cermak*. Charles University, Prague.

Patrick Hanks. 2006. Metaphoricity is gradable. *Trends in Linguistic Studies and Monographs*, 171:17.

Patrick Hanks. 2012. The roles and structure of comparisons, similes, and metaphors in natural language (an analogical system). Presented at the Stockholm Metaphor Festival.

Patrick Hanks. 2013. *Lexical Analysis: Norms and Exploitations*. MIT Press.

Dirk Hovy, Shashank Srivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huiying Li, Whitney Sanders, and Eduard Hovy. 2013. Identifying metaphorical word use with tree kernels. In *Proceedings of the NAACL Workshop on Metaphors for NLP*.

M. Israel, J.R. Harding, and V. Tobin. 2004. On simile. *Language, Culture, and Mind. CSLI Publications*.

Nitin Jindal and Bing Liu. 2006. Identifying comparative sentences in text documents. In *Proceedings of SIGIR*.

George Lakoff and Mark Johnson. 1980. *Metaphors we live by*. University of Chicago Press.

Bin Li, Jiajun Chen, and Yingjie Zhang. 2012. Web based collection and comparison of cognitive properties in English and Chinese. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*.

Christoph Lofi, Christian Nieke, and Nigel Collier. 2014. Discriminating rhetorical analogies in social media. In *Proceedings of EACL*.

André FT Martins, Noah A Smith, Eric P Xing, Pedro MQ Aguiar, and Mário AT Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of EMNLP*.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of RecSys*.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Rosamund Moon. 2008. Conventionalized as-similes in English: A problem case. *International Journal of Corpus Linguistics*, 13(1):3–37.

Rosamund Moon. 2011. Simile and dissimilarity. *Journal of Literary Semantics*, 40(2):133–157.

Robert Munro, Steven Bethard, Victor Kuperman, Vicky Tzuyin Lai, Robin Melnick, Christopher Potts, Tyler Schnoebelen, and Harry Tily. 2010. Crowdsourcing and language studies: The new generation of linguistic data. In *Proceedings of the NAACL Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Vlad Niculae. 2013. Comparison pattern matching and creative simile recognition. In *Proceedings of the Joint Symposium on Semantic Processing*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL-HLT*.

Allan Paivio, John C Yuille, and Stephen A Madigan. 1968. Concreteness, imagery, and meaningfulness values for 925 nouns. *Journal of Experimental Psychology*, 76(1p2):1.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Carlos Roncero, John M Kennedy, and Ron Smyth. 2006. Similes on the internet have explanations. *Psychonomic Bulletin & Review*, 13(1):74–77.

Ekaterina Shutova and Lin Sun. 2013. Unsupervised metaphor identification using hierarchical graph factorization clustering. In *Proceedings of NAACL-HLT*.

Ekaterina Shutova. 2010. Models of metaphor in NLP. In *Proceedings of ACL*.

Niket Tandon, Gerard de Melo, and Gerhard Weikum. 2014. Smarter than you think: Acquiring comparative commonsense knowledge from the web. In *Proceedings of AAAI*.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014a. Metaphor detection with cross-lingual model transfer. In *Proceedings of ACL*.

Yulia Tsvetkov, Nathan Schneider, Dirk Hovy, Archna Bhatia, Manaal Faruqui, and Chris Dyer. 2014b. Augmenting English adjective senses with supersenses. In *Proceedings of LREC*.

Peter D Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of EMNLP*.

Tony Veale and Yanfen Hao. 2008. A context-sensitive framework for lexical ontologies. *Knowledge Engineering Review*, 23(1):101–115.

Tony Veale and Yanfen Hao. 2011. Exploiting readymades in linguistic creativity: A system demonstration of the Jigsaw Bard. In *Proceedings of ACL (System Demonstrations)*.

Tony Veale and Guofu Li. 2013. Creating similarity: Lateral thinking for vertical similarity judgments. In *Proceedings of ACL*.

Tony Veale. 2012a. A computational exploration of creative similes. *Metaphor in Use: Context, Culture, and Communication*, 38:329.

Tony Veale. 2012b. A context-sensitive, multi-faceted model of lexico-conceptual affect. In *Proceedings of ACL*.

Classifying Idiomatic and Literal Expressions Using Topic Models and Intensity of Emotions

Jing Peng & Anna Feldman
Computer Science/Linguistics
Montclair State University
Montclair, New Jersey, USA

Ekaterina Vylomova
Computer Science
Bauman State Technical University
Moscow, Russia

{pengj,feldmana}@mail.montclair.edu evylomova@gmail.com

Abstract

We describe an algorithm for automatic classification of idiomatic and literal expressions. Our starting point is that words in a given text segment, such as a paragraph, that are high-ranking representatives of a common topic of discussion are less likely to be a part of an idiomatic expression. Our additional hypothesis is that contexts in which idioms occur, typically, are more affective and therefore, we incorporate a simple analysis of the intensity of the emotions expressed by the contexts. We investigate the bag of words topic representation of one to three paragraphs containing an expression that should be classified as idiomatic or literal (a target phrase). We extract topics from paragraphs containing idioms and from paragraphs containing literals using an unsupervised clustering method, Latent Dirichlet Allocation (LDA) (Blei et al., 2003). Since idiomatic expressions exhibit the property of non-compositionality, we assume that they usually present different semantics than the words used in the local topic. We treat idioms as semantic outliers, and the identification of a semantic shift as outlier detection. Thus, this topic representation allows us to differentiate idioms from literals using local semantic contexts. Our results are encouraging.

1 Introduction

The definition of what is literal and figurative is still object of debate. Ariel (2002) demonstrates that literal and non-literal meanings cannot always be distinguished from each other. Literal meaning is originally assumed to be conventional, compositional, relatively context independent, and truth conditional. The problem is that the boundary is not clear-cut, some figurative expressions are compositional – metaphors and many idioms; others are conventional – most of the idioms. Idioms present great challenges for many Natural Language Processing (NLP) applications. They can violate selection restrictions (Sporleder and Li, 2009) as in *push one's luck* under the assumption that only concrete things can normally be pushed. Idioms can disobey typical subcategorization constraints (e.g., *in line* without a determiner before line), or change the default assignments of semantic roles to syntactic categories (e.g., in *X breaks something with Y*, Y typically is an instrument but for the idiom *break the ice*, it is more likely to fill a patient role as in *How to break the ice with a stranger*). In addition, many potentially idiomatic expressions can be used either literally or figuratively, depending on the context. This presents a great challenge for machine translation. For example, a machine translation system must translate *held fire* differently in *Now, now, hold your fire until I've had a chance to explain. Hold your fire, Bill. You're too quick to complain.* and *The sergeant told the soldiers to hold their fire. Please hold your fire until I get out of the way*. In fact, we tested the last two examples using the Google Translate engine and we got proper translations of the two neither into Russian nor into Hebrew, Spanish, or Chinese. Most current translation systems rely on large repositories of idioms. Unfortunately, these systems are not capable to tell apart literal from figurative usage of the same expression in context. Despite the common perception that phrases that can be idioms are mainly used in their idiomatic sense, Fazly et al. (2009)'s analysis of 60 idioms has shown that close to half of these also have a clear literal meaning; and of those with a literal meaning, on average around 40% of their usages are literal.

In this paper we describe an algorithm for automatic classification of idiomatic and literal expressions. Our starting point is that words in a given text segment, such as a paragraph, that are high-ranking representatives of a common topic of discussion are less likely to be a part of an idiomatic expression. Our additional hypothesis is that contexts in which idioms occur, typically, are more affective and therefore, we incorporate a simple analysis of the intensity of the emotions expressed by the contexts. We investigate the bag of words *topic* representation of one to three paragraphs containing an expression that should be classified as idiomatic or literal (a target phrase). We extract topics from paragraphs containing idioms and from paragraphs containing literals using an unsupervised clustering method, Latent Dirichlet Allocation (LDA) (Blei et al., 2003). Since idiomatic expressions exhibit the property of non-compositionality, we assume that they usually present different semantics than the words used

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2019–2027,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

in the local topic. We treat idioms as semantic outliers, and the identification of semantic shift as outlier detection. Thus, this topic representation allows us to differentiate idioms from literals using the local semantics.

The paper is organized as follows. Section 2 briefly describes previous approaches to idiom recognition or classification. In Section 3 we describe our approach in detail, including the hypothesis, the topic space representation, and the proposed algorithm. After describing the preprocessing procedure in Section 4, we turn to the actual experiments in Sections 5 and 6. We then compare our approach to other approaches (Section 7) and discuss the results (Section 8).

2 Previous Work

Previous approaches to idiom detection can be classified into two groups: 1) Type-based extraction, i.e., detecting idioms at the type level; 2) token-based detection, i.e., detecting idioms in context. Type-based extraction is based on the idea that idiomatic expressions exhibit certain linguistic properties that can distinguish them from literal expressions (Sag et al. (2002); Fazly et al. (2009)), among many others, discuss various properties of idioms. Some examples of such properties include 1) lexical fixedness: e.g., neither 'shoot the wind' nor 'hit the breeze' are valid variations of the idiom shoot the breeze and 2) syntactic fixedness: e.g., *The guy kicked the bucket* is potentially idiomatic whereas *The bucket was kicked* is not idiomatic anymore; and of course, 3) non-compositionality. Thus, some approaches look at the tendency for words to occur in one particular order, or a fixed pattern. Hearst (1992) identifies lexico-syntactic patterns that occur frequently, are recognizable with little or no precoded knowledge, and indicate the lexical relation of interest. Widdows and Dorow (2005) use Hearst's concept of lexicosyntactic patterns to extract idioms that consist of fixed patterns between two nouns. Basically, their technique works by finding patterns such as "thrills and spills", whose reversals (such as "spills and thrills") are never encountered.

While many idioms do have these properties, many idioms fall on the continuum from being compositional to being partly unanalyzable to completely non-compositional (Cook et al. (2007)). Fazly et al. (2009); Li and Sporleder (2010), among others, notice that type-based approaches do not work on expressions that can be interpreted idiomatically or literally depending on the context and thus, an approach that considers tokens in context is more appropriate for the task of idiom recognition.

A number of token-based approaches have been discussed in the literature, both supervised (Katz and Giesbrech (2006)), weakly supervised (Birke and Sarkar (2006)) and unsupervised (Sporleder and Li (2009); Fazly et al. (2009)). Fazly et al. (2009) develop statistical measures for each linguistic property of idiomatic expressions and use them both in a type-based classification task and in a token identification task, in which they distinguish idiomatic and literal usages of potentially idiomatic expressions in context. Sporleder and Li (2009) present a graph-based model for representing the lexical cohesion of a discourse. Nodes represent tokens in the discourse, which are connected by edges whose value is determined by a semantic relatedness function. They experiment with two different approaches to semantic relatedness: 1) Dependency vectors, as described in Pado and Lapata (2007); 2) Normalized Google Distance (Cilibrasi and Vitányi (2007)). Sporleder and Li (2009) show that this method works better for larger contexts (greater than five paragraphs). Li and Sporleder (2010) assume that literal and figurative data are generated by two different Gaussians, literal and non-literal and the detection is done by comparing which Gaussian model has a higher probability to generate a specific instance. The approach assumes that the target expressions are already known and the goal is to determine whether this expression is literal or figurative in a particular context. The important insight of this method is that figurative language in general exhibits less semantic cohesive ties with the context than literal language.

Feldman and Peng (2013) describe several approaches to automatic idiom identification. One of them is idiom recognition as outlier detection. They apply principal component analysis for outlier detection – an approach that does not rely on costly annotated training data and is not limited to a specific type of a syntactic construction, and is generally language independent. The quantitative analysis provided in their work shows that the outlier detection algorithm performs better and seems promising. The qualitative analysis also shows that their algorithm has to incorporate several important properties of the idioms: (1) Idioms are relatively non-compositional, comparing to literal expressions or other types of collocations. (2) Idioms violate local cohesive ties, as a result, they are semantically distant from the local topics. (3) While not all semantic outliers are idioms, non-compositional semantic outliers are likely to be idiomatic. (4) Idiomaticity is not a binary property. Idioms fall on the continuum from being compositional to being partly unanalyzable to completely non-compositional.

The approach described below is taking Feldman and Peng (2013)'s original idea and is trying to address (2) directly and (1) indirectly. Our approach is also somewhat similar to Li and Sporleder (2010) because it also relies on a list of potentially idiomatic expressions.

3 Our Hypothesis

Similarly to Feldman and Peng (2013), out starting point is that idioms are semantic outliers that violate cohesive structure, especially in local contexts. However, our task is framed as supervised classification and we rely on data annotated for idiomatic and literal expressions. We hypothesize that words in a given text

segment, such as a paragraph, that are high-ranking representatives of a common topic of discussion are less likely to be a part of an idiomatic expression in the document.

3.1 Topic Space Representation

Instead of the simple bag of words representation of a target document (segment of three paragraphs that contains a target phrase), we investigate the bag of words topic representation for target documents. That is, we extract topics from paragraphs containing idioms and from paragraphs containing literals using an unsupervised clustering method, Latent Dirichlet Allocation (LDA) (Blei et al., 2003). The idea is that if the LDA model is able to capture the semantics of a target document, an idiomatic phrase will be a "semantic" outlier of the themes. Thus, this topic representation will allow us to differentiate idioms from literals using the semantics of the local context.

Let $d = \{w_1, \cdots, w_N\}^t$ be a segment (document) containing a target phrase, where N denotes the number of terms in a given corpus, and t represents transpose. We first compute a set of m topics from d. We denote this set by

$$T(d) = \{t_1, \cdots, t_m\},$$

where $t_i = (w_1, \cdots, w_k)^t$. Here w_j represents a word from a vocabulary of W words. Thus, we have two representations for d: (1) d, represented by its original terms, and (2) $\hat{d}$, represented by its topic terms. Two corresponding term by document matrices will be denoted by M_D and $M_{\hat{D}}$, respectively, where D denotes a set of documents. That is, M_D represents the original "text" term by document matrix, while $M_{\hat{D}}$ represents the "topic" term by document matrix.

Figure 1 shows the potential benefit of topic space representation. In the figure, text segments containing target phrase "blow whistle" are projected on a two dimensional subspace. The left figure shows the projection in the "text" space, represented by the term by document matrix M_D. The middle figure shows the projection in the topic space, represented by $M_{\hat{D}}$. The topic space representation seems to provide a better separation.

We note that when learning topics from a small data sample, learned topics can be less coherent and interpretable, thus less useful. To address this issue, regularized LDA has been proposed in the literature (Newman et al., 2011). A key feature is to favor words that exhibit short range dependencies for a given topic. We can achieve a similar effect by placing restrictions on the vocabulary. For example, when extracting topics from segments containing idioms, we may restrict the vocabulary to contain words from these segments only. The middle and right figures in Figure 1 illustrate a case in point. The middle figure shows a projection onto the topic space that is computed with a restricted vocabulary, while the right figure shows a projection when we place no restriction on the vocabulary. That is, the vocabulary includes terms from documents that contain both idioms and literals.

Note that by computing $M_{\hat{D}}$, the topic term by document matrix, from the training data, we have created a vocabulary, or a set of "features" (i.e., topic terms) that is used to directly describe a query or test segment. The main advantage is that topics are more accurate when computed by LDA from a large collection of idiomatic or literal contexts. Thus, these topics capture more accurately the semantic contexts in which the target idiomatic and literal expressions typically occur. If a target query appears in a similar semantic context, the topics will be able to describe this query as well. On the other hand, one might similarly apply LDA to a given query to extract query topics, and create the query vector from the query topics. The main disadvantage is that LDA may not be able to extract topic terms that match well with those in the training corpus, when applied to the query in isolation.

3.2 Algorithm

The main steps of the proposed algorithm, called **TopSpace**, are shown below.

Input: $D = \{d_1, \cdots, d_k, d_{k+1}, \cdots, d_n\}$: training documents of k idioms and $n-k$ literals.
$Q = \{q_1, \cdots, q_l\}$: l query documents.

1. Let $DicI$ be the vocabulary determined solely from idioms $\{d_1, \cdots, d_k\}$. Similarly, let $DicL$ be the vocabulary obtained from literals $\{d_{k+1}, \cdots, d_n\}$.
2. For a document d_i in $\{d_1, \cdots, d_k\}$, apply LDA to extract a set of m topics $T(d_i) = \{t_1, \cdots, t_m\}$ using $DicI$. For $d_i \in \{d_{k+1}, \cdots, d_n\}$, $DicL$ is used.
3. Let $\hat{D} = \{\hat{d}_1, \cdots, \hat{d}_k, \hat{d}_{k+1}, \cdots, \hat{d}_n\}$ be the resulting topic representation of D.
4. Compute the term by document matrix $M_{\hat{D}}$ from $\hat{D}$, and let $DicT$ and gw be the resulting dictionary and global weight (idf), respectively.
5. Compute the term by document matrix M_Q from Q, using $DicT$ and gw from the previous step.

Output: $M_{\hat{D}}$ and M_Q

To summarize, after splitting our corpus (see section 4) into paragraphs and preprocessing it, we extract topics from paragraphs containing idioms and from paragraphs containing literals. We then compute a term by document matrix, where terms are topic terms and documents are topics extracted from the paragraphs. Our test data are represented as a term-by-document matrix as well (See the details in section 5).

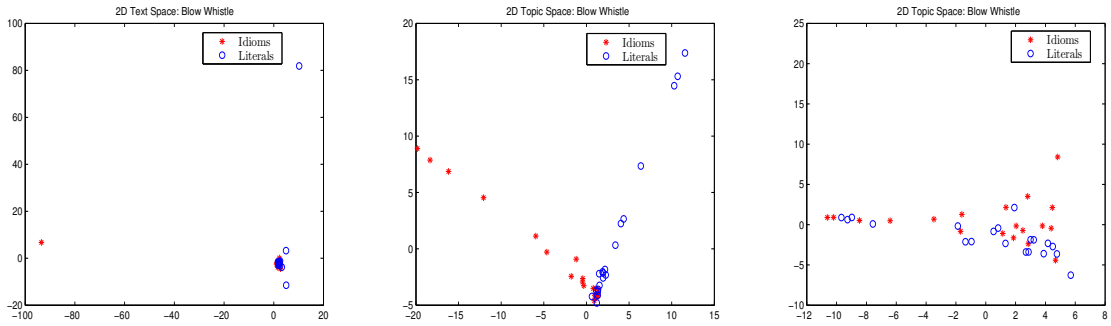


Figure 1: 2D projection of text segments containing "blow whistle." Left panel: Original text space. Middle panel: Topic space with restricted vocabulary. Right panel: Topic space with enlarged vocabulary.

3.3 Fisher Linear Discriminant Analysis

Once $M_{\hat{D}}$ and M_Q are obtained, a classification rule can be applied to predict idioms vs. literals. The approach we are taking in this work for classifying idioms vs. literals is based on Fisher's discriminant analysis (FDA) (Fukunaga, 1990). FDA often significantly simplifies tasks such as regression and classification by computing low-dimensional subspaces having statistically uncorrelated or discriminant variables. In language analysis, statistically uncorrelate or discriminant variables are extracted and utilized for description, detection, and classification. Woods et al. (1986), for example, use statistically uncorrelated variables for language test scores. A group of subjects is scored on a battery of language tests, where the subtests measure different abilities such as vocabulary, grammar or reading comprehension. Horvath (1985) analyzes speech samples of Sydney speakers to determine the relative occurrence of five different variants of each of five vowels sounds. Using this data, the speakers cluster according to such factors as gender, age, ethnicity and socio-economic class.

A similar approach has been discussed in Peng et al. (2010). FDA is a class of methods used in machine learning to find the linear combination of features that best separate two classes of events. FDA is closely related to principal component analysis (PCA), where a linear combination of features that best explains the data. Discriminant analysis explicitly exploits class information in the data, while PCA does not.

Idiom classification based on discriminant analysis has several advantages. First, as has been mentioned, it does not make any assumption regarding data distributions. Many statistical detection methods assume a Gaussian distribution of normal data, which is far from reality. Second, by using a few discriminants to describe data, discriminant analysis provides a compact representation of the data, resulting in increased computational efficiency and real time performance.

In FDA, within-class, between-class, and mixture scatter matrices are used to formulate the criteria of class separability. Consider a J class problem, where m_0 is the mean vector of all data, and m_j is the mean vector of jth class data. A within-class scatter matrix characterizes the scatter of samples around their respective class mean vector, and it is expressed by

$$S_w = \sum_{j=1}^{J} p_j \sum_{i=1}^{l_j} (x_i^j - m_j)(x_i^j - m_j)^t, \qquad (1)$$

where l_j is the size of the data in the jth class, p_j ($\sum_j p_j = 1$) represents the proportion of the jth class contribution, and t denotes the transpose operator. A between-class scatter matrix characterizes the scatter of the class means around the mixture mean m_0. It is expressed by

$$S_b = \sum_{j=1}^{J} p_j (m_j - m_0)(m_j - m_0)^t. \qquad (2)$$

The mixture scatter matrix is the covariance matrix of all samples, regardless of their class assignment, and it is given by

$$S_m = \sum_{i=1}^{l} (x_i - m_0)(x_i - m_0)^t = S_w + S_b. \qquad (3)$$

The Fisher criterion is used to find a projection matrix $W \in \Re^{q \times d}$ that maximizes

$$J(W) = \frac{|W^t S_b W|}{|W^t S_w W|}. \qquad (4)$$

In order to determine the matrix W that maximizes $J(W)$, one can solve the generalized eigenvalue problem: $S_b w_i = \lambda_i S_w w_i$. The eigenvectors corresponding to the largest eigenvalues form the columns of W. For a two class problem, it can be written in a simpler form: $S_w w = m = m_1 - m_2$, where m_1 and m_2 are the means of the two classes.

4 Data preprocessing

4.1 Verb-noun constructions

For our experiments we use the British National Corpus (BNC, Burnard (2000)) and a list of verb-noun constructions (VNCs) extracted from BNC by Fazly et al.

(2009); Cook et al. (2008) and labeled as L (Literal), I (Idioms), or Q (Unknown). The list contains only those VNCs whose frequency was greater than 20 and that occurred at least in one of two idiom dictionaries (Cowie et al., 1983; Seaton and Macaulay, 2002). The dataset consists of 2,984 VNC tokens. For our experiments we only use VNCs that are annotated as I or L.

4.2 Lemmatization

Instead of dealing with various forms of the same root, we use lemmas provided by the BNC XML annotation, so our corpus is lemmatized. We also apply the (modified) Google stop list before extracting the topics. The reason we modified the stop list is that some function words can potentially be idiom components (e.g., certain prepositions).

4.3 Paragraphs

We use the original SGML annotation to extract paragraghs from BNC. We only kept the paragraphs that contained VNCs for our experiments. We experimented with texts of one paragraph length (single paragraph contexts) and of three-paragraph length (multi-paragraph contexts). An example of multi-paragraph contexts is shown below:

So, reluctantly, I joined Jack Hobbs in not rocking the boat, reporting the play and the general uproar with perhaps too much impartiality. My reports went to all British newspapers, with special direct services by me to India, South Africa and West Indies; even to King George V in Buckingham Palace, who loved his cricket. In other words, I was to some extent leading the British public astray.

I regret I can shed little new light on the mystery of who **blew the whistle** *on the celebrated dressing-room scene after Woodfull was hit. while he was lying on the massage table after his innings waiting for a doctor, Warner and Palairet called to express sympathy.*

Most versions of Woodfull's reply seem to agree that he said. There are two teams out there on the oval. One is playing cricket, the other is not. This game is too good to be spoilt. It is time some people got out of it. Warner and Palairet were too taken aback to reply. They left the room in embarrassment.

Single paragraph contexts simply consist of the middle paragraph.

5 Experiments

5.1 Methods

We have carried out an empirical study evaluating the performance of the proposed algorithm. For comparison, the following methods are evaluated. (1) The proposed algorithm **TopSpace** (1), where the data are represented in topic space. (2) **TexSpace** algorithm, where the data are represented in original text space. For each representation, two classification schemes are applied: a) FDA (Eq. 4), followed by the nearest neighbor rule. b) SVMs with Gaussian kernels (Cristianini and Shawe-Taylor (2000)). For the nearest neighbor rule, the number of nearest neighbors is set to $\lceil n/5 \rceil$, where n denotes the number of training examples. For SVMs, kernel width and soft margin parameters are set to default values.

5.2 Data Sets

The following data sets are used to evaluate the performance of the proposed technique. These data sets have enough examples from both idioms and literals to make our results meaningful. On average, the training data is 6K word tokens. Our test data is of a similar size.

BlowWhistle: This data set has 78 examples, 27 of which are idioms and the remaining 51 are literals. The training data for **BlowWhistle** consist of 40 randomly chosen examples (20 paragraphs containing idioms and 20 paragraphs containing literals). The remaining 38 examples (7 idiomatic and 31 literals) are used as test data.

MakeScene: This data set has 50 examples, 30 of which are paragraphs containing idioms and the remaining 20 are paragraphs containing literals. The training data for **MakeScene** consist of 30 randomly chosen examples, 15 of which are paragraphs containing *make scene* as an idiom and the rest 15 are paragraphs containing *make scene* as a literal. The remaining 20 examples (15 idiomatic paragraphs and 5 literals) are used as test data.

LoseHead: This data set has 40 examples, 21 of which are idioms and the remaining 19 are literals. The training data for **LoseHead** consist of 30 randomly chosen examples (15 idiomatic and 15 literal). The remaining 10 examples (6 idiomatic and 4 literal) are used as test data.

TakeHeart: This data set has 81 examples, 61 of which are idioms and the remaining 20 are literals. The training data for **TakeHeart** consist of 30 randomly chosen examples (15 idiomatic and 15 literals). The remaining 51 examples (46 idiomatic and 5 literals) are used as test data.

5.3 Adding affect

Nunberg et al. (1994) notice that "idioms are typically used to imply a certain evaluation or affective stance toward the things they denote". Language users usually choose an idiom in non-neutral contexts. The situations that idioms describe can be positive or negative; however, the polarity of the context is not as important as the strength of the emotion expressed. So, we decided to incorporate the knowledge about the emotion strength into our algorithm. We use a database of word norms collected by Warriner et al. (2013). This database contains almost 14,000 English lemmas annotated with three components of emotions: valence (the pleasantness of a stimulus), arousal (the intensity of emotion provoked by a stimulus), and dominance

Table 1: Average accuracy of competing methods on four datasets in single paragraph contexts: A = Arousal

Model	**BlowWhistle**			**LoseHead**			**MakeScene**			**TakeHeart**		
	Prec	Recall	Acc	Prec	Recall	Acc	Prec	Recall	Acc	Prec	Recall	Acc
FDA-Topics	0.44	0.40	0.79	**0.70**	**0.90**	**0.70**	**0.82**	**0.97**	**0.81**	0.91	0.97	0.89
FDA-Topics+A	0.51	0.51	0.75	0.78	0.68	0.66	0.80	0.99	0.80	0.93	0.84	0.80
FDA-Text	**0.37**	**0.81**	**0.63**	0.60	0.88	0.58	0.82	0.89	0.77	0.36	0.38	0.41
FDA-Text+A	0.42	0.49	0.76	0.64	0.92	0.63	**0.83**	**0.95**	**0.82**	0.75	0.53	0.53
SVMs-Topics	0.08	0.39	0.59	0.28	0.25	0.45	0.59	0.74	0.61	**0.91**	**1.00**	**0.91**
SVMs-Topics+A	0.06	0.21	0.69	0.38	0.18	0.44	0.53	0.40	0.44	**0.91**	**1.00**	**0.91**
SVMs-Text	0.08	0.39	0.59	0.36	0.60	0.52	0.23	0.30	0.40	0.42	0.16	0.22
SVMs-Text+A	0.15	0.51	0.60	0.31	0.38	0.48	0.37	0.40	0.45	0.95	0.48	0.50

(the degree of control exerted by a stimulus). These components were elicited from human subjects via an Amazon Mechanical Turk crowdsourced experiment. We only used the arousal feature in our experiments because we were interested in the intensity of the emotion rather than its valence.

For a document $d = \{w_1, \cdots, w_N\}^t$, we calculate the corresponding arousal value a_i for each w_i, obtaining $d_A = \{a_1, \cdots, a_N\}^t$. Let m_A be the average arousal value calculated over the entire training data. The centered arousal value for a training document is obtained by subtracting m_A from d_A, i.e., $\bar{d}_A = d_A - m_A = \{a_1 - m_A, \cdots, a_N - m_A\}^t$. Similarly, the centered arousal value for a query is computed according to $\bar{q}_A = q_A - m_A = \{q_1 - m_A, \cdots, q_N - m_A\}^t$. That is, the training arousal mean is used to center both training and query arousal values. The corresponding arousal matrices for D, $\hat{D}$, and Q are A_D, $A_{\hat{D}}$, A_Q, respectively. To incorporate the arousal feature, we simply compute

$$\Theta_D = M_D + A_D, \tag{5}$$

and

$$\Theta_{\hat{D}} = M_{\hat{D}} + A_{\hat{D}}. \tag{6}$$

The arousal feature can be similarly incoporated into query $\Theta_Q = M_Q + A_Q$.

6 Results

Table 1 shows the average precision, recall, and accuracy of the competing methods on the four data sets over 10 runs in simple paragraph contexts. Table 2 shows the results for the multi-paragraph contexts. Note that for single paragraph contexts, we chose two topics, each having 10 terms. For multi-paragrah contexts, we had four topics, with 10 terms per topic. No optimization was made for selecting the number of topics as well as the number of terms per topic. In the tables, the best performance in terms of the sum of precision, recall and accuracy is given in boldface.

The results show that the topic representation achieved the best performance in 6 out of 8 cases. Figure 2 plots the overall aggregated performance in terms of topic vs text representations across the entire data sets, regardless of the classifiers used. Everything else being equal, this clearly shows the advantage of topics over simple text representation.

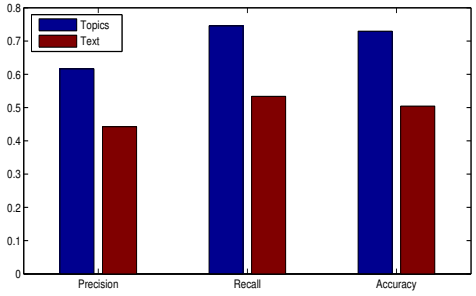


Figure 2: Aggregated performance: Topic vs text representations.

The arousal feature (Eqs 5 and 6) also improved the overall performance, particularly in text representation (Eq. 5). This can be seen in the top panel in Figure 3. In fact, in 2/8 cases, text representation coupled with the arousal feature achieved the best performance. One possible explanation is that the LDA model already performed "feature" selection (choosing topic terms), to the extent possible. Thus, any additional information such as arousal only provides marginal improvement at the best (bottom panel in Figure 3). On the other hand, original text represents "raw" features, whereby arousal information helps provide better contexts, thus improving overall performance.

Figure 4 shows a case in point: the average (sorted) arousal values of idioms and literals of the target phrase "lose head." The upper panel plots arousal values in the text space, while lower panel plots arousal values in the topic space. The plot supports the results shown in Tables 1 and 2, where the arousal feature generally improves text representation.

7 Comparisons with other approaches

Even though we used Fazly et al. (2009)'s dataset for these experiments, the direct comparison with their method is impossible here because our task is formulated differently and we do not use the full dataset for the experiments. Fazly et al. (2009)'s unsupervised

Table 2: Average accuracy of competing methods on four datasets in multiple paragraph contexts: A = Arousal

Model	**BlowWhistle**			**LoseHead**			**MakeScene**			**TakeHeart**		
	Prec	Recall	Acc	Prec	Recall	Acc	Prec	Recall	Acc	Prec	Recall	Acc
FDA-Topics	**0.62**	**0.60**	**0.83**	**0.76**	**0.97**	**0.78**	0.79	0.95	0.77	**0.93**	**0.99**	**0.92**
FDA-Topics+A	0.47	0.44	0.79	0.74	0.93	0.74	0.82	0.69	0.65	0.92	0.98	0.91
FDA-Text	0.65	0.43	0.84	0.72	0.73	0.65	0.79	0.95	0.77	0.46	0.40	0.42
FDA-Text+A	0.45	0.49	0.78	0.67	0.88	0.65	**0.80**	**0.99**	**0.80**	0.47	0.29	0.33
SVMs-Topics	0.07	0.40	0.56	0.60	0.83	0.61	0.46	0.57	0.55	0.90	1.00	0.90
SVMs-Topics+A	0.21	0.54	0.55	0.66	0.77	0.64	0.42	0.29	0.41	0.91	1.00	0.91
SVMs-Text	0.17	0.90	0.25	0.30	0.50	0.50	0.10	0.01	0.26	0.65	0.21	0.26
SVMs-Text+A	0.24	0.87	0.41	0.66	0.85	0.61	0.07	0.01	0.26	0.74	0.13	0.20

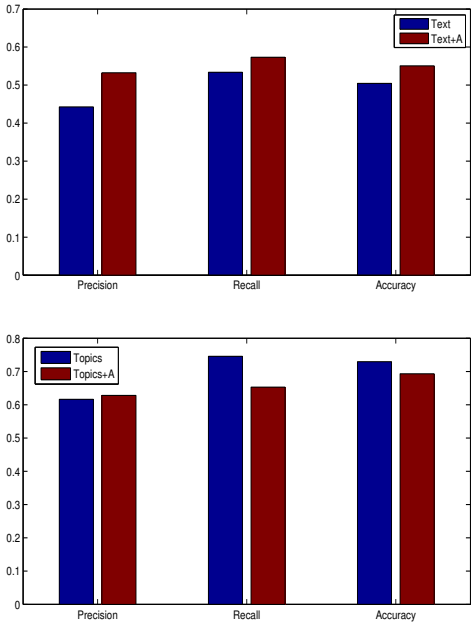


Figure 3: Aggregated performance: Text vs. text+Arousal representations (top) and Topics vs. Topics+Arousal representations (bottom).

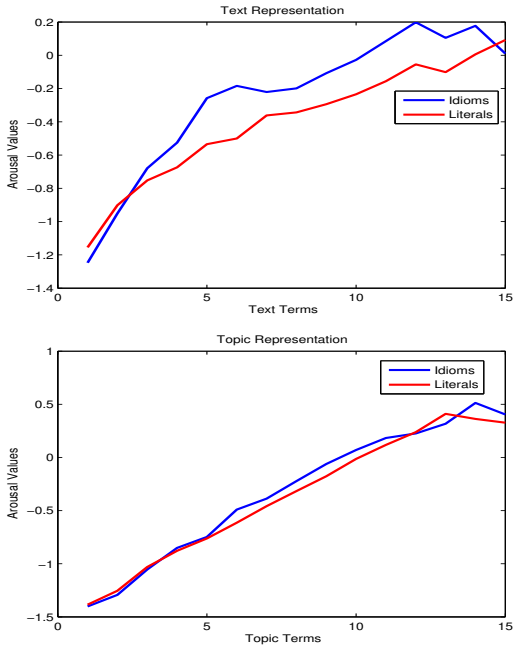


Figure 4: Average arousal values–Upper panel: Text space. Lower panel: Topic space.

model that relies on the so-called canonical forms gives 72.4% (macro-)accuracy on the extraction of idiomatic tokens when evaluated on their test data.

We cannot compare our method directly with the other methods discussed in section 2 either because each uses a different dataset or formulates the task differently (detection vs. recognition vs. identification). However, we can compare the method presented here with Feldman and Peng (2013) who also experiment with LDA, use similar data, and frame the problem as classification. Their goal, however, is to classify *sentences* as either idiomatic or literal. To obtain a discriminant subspace, they train their model on a small number of randomly selected idiomatic and non-idiomatic sentences. They then project both the training and the test data on the chosen subspace and use the three nearest neighbor (3NN) classifier to obtain accuracy. The average accuracy they report is 80%. Our method clearly outperforms the Feldman and Peng (2013) approach (at least on the dataset we use).

8 Discussion and Conclusion

We have described an algorithm for automatic classification of idiomatic and literal expressions. We have investigated the bag of words topic representation for target documents (segments of one or three paragraphs that contains a target phrase). The approach definitely outperforms the baseline model that is based on the simple bag of words representation, but it also outperforms approaches previously discussed in the literature. Our model captures the local semantics and thus is capable to identify semantic outliers (=idioms).

While we realize that the data set we use is small, the results are encouraging. We notice that using 3 paragraphs for local contexts improves the performance of the classifiers. The reason is that some paragraphs are

relatively short. A larger context provides more related terms, which gives LDA more opportunities to sample these terms.

Idioms are also relatively non-compositional. While we do not measure their non-compositionality in this approach, we indirectly touch upon this property by hypothesizing that non-compositional idiomatic expressions are likely to be far from the local topics.

We feel that incorporating the intensity of emotion expressed by the context into our model improves performance, in particular, in text representation. When we performed a qualitative analysis of the results trying to determine the causes of false positives and negatives, we noticed that there were quite a number of cases that improved after incorporating the arousal feature into the model. For example, the FDA:topic classifier labels "blow the whistle" as literal in the following context, but FDA:topics+A marks this expression as idiomatic (italicized words indicate words with relatively high arousal values):

Peter thought it all out very *carefully*. He decided the *wisest* course was to pool all he had made over the last two years, enabling Julian to purchase the lease of a high street property. This would enable them to set up a business on a more *settled* and *permanent* trading basis. Before long they opened a grocery-cum-delicatessen in a *good* position as far as passing trade was concerned. Peter's investment was not misplaced. The business did very well with the two lads greatly *appreciated* locally for their *hard* work and quality of service. The range of goods they were able to carry was *welcomed* in the area, as well as lunchtime sandwich facilities which had previously been missing in the neighbourhood.

Success was the fruit of some three years' *strenuous* work. But it was more than a *shock* when Julian admitted to Peter that he had been running up *huge debts* with their bank. Peter knew that Julian *gambled*, but he hadn't expected him to *gamble* to that level, and certainly not to use the shop as security. With continual borrowing over two years, the bank had **blown the whistle**. Everything was gone. Julian was *bankrupt.* Even if they'd had a formal partnership, which they didn't, it would have made no difference. Peter *lost* all he'd made, and with it his *chance* to help his parents and his younger brother and sister, Toby and Laura.

Peter was *heartbroken*. His father had said all along: neither a lender nor a borrower. Peter had found out the *hard* way. But as his mother observed, he was the same Peter, he'd pick himself up somehow. Once again, Peter was resolute. He made up his mind he'd never make the same *mistake* twice. It wasn't just the money or the hard work, though the waste of that was *difficult* enough to accept. Peter had been working a *debt* of *love*. He'd done all this for his parents, particularly for his father, whose *dedication* to his children had always *impressed* Peter and moved him *deeply*. And now it had all come to nothing.

Therefore, we think that idioms have the tendency to appear in more affective contexts; and we think that incorporating more sophisticated sentiment analysis into our model will improve the results.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1319846. We also thank the anonymous reviewers for useful comments. The third author thanks the Fulbright Foundation for giving her an opportunity to conduct this research at Montclair State University (MSU).

References

Ariel, M. (2002). The demise of unique concept of literal meaning. *Journal of Pragmatics 34*, 361–402.

Birke, J. and A. Sarkar (2006). A clustering approach to the nearly unsupervised recognition of nonliteral language. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*, Trento, Italy, pp. 329–226.

Blei, D., A. Ng, and M. Jordan (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research 3*, 993–1022.

Burnard, L. (2000). *The British National Corpus Users Reference Guide*. Oxford University Computing Services.

Cilibrasi, R. and P. M. B. Vitányi (2007). The google similarity distance. *IEEE Trans. Knowl. Data Eng. 19*(3), 370–383.

Cook, P., A. Fazly, and S. Stevenson (2007). Pulling their weight: Exploiting syntactic forms for the automatic identification of idiomatic expressions in context. In *Proceedings of the ACL 07 Workshop on A Broader Perspective on Multiword Expressions*, pp. 41–48.

Cook, P., A. Fazly, and S. Stevenson (2008, June). The VNC-Tokens Dataset. In *Proceedings of the LREC Workshop: Towards a Shared Task for Multiword Expressions (MWE 2008)*, Marrakech, Morocco.

Cowie, A. P., R. Mackin, and I. R. McCaig (1983). *Oxford Dictionary of Current Idiomatic English*, Volume 2. Oxford University Press.

Cristianini, N. and J. Shawe-Taylor (2000). *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge, UK: Cambridge University Press.

Fazly, A., P. Cook, and S. Stevenson (2009). Unsupervised Type and Token Identification of Idiomatic Expressions. *Computational Linguistics 35*(1), 61–103.

Feldman, A. and J. Peng (2013). Automatic detection of idiomatic clauses. In *Computational Linguistics and Intelligent Text Processing*, pp. 435–446. Springer.

Fukunaga, K. (1990). *Introduction to statistical pattern recognition*. Academic Press.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, Stroudsburg, PA, USA, pp. 539–545. Association for Computational Linguistics.

Horvath, B. M. (1985). *Variation in Australian English*. Cambridge: Cambridge University PRess.

Katz, G. and E. Giesbrech (2006). Automatic Identification of Non-compositional Multiword Expressions using Latent Semantic Analysis. In *Proceedings of the ACL/COLING-06 Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pp. 12–19.

Li, L. and C. Sporleder (2010). Using Gaussian Mixture Models to Detect Figurative Language in Context. In *Proceedings of NAACL/HLT 2010*.

Newman, D., E. V. Bonilla, and W. L. Buntine (2011). Improving topic coherence with regularized topic models. In *NIPS*, pp. 496–504.

Nunberg, G., I. A. Sag, and T. Wasow (1994). Idioms. *Language 70*(3), 491–538.

Pado, S. and M. Lapata (2007). Dependency-based construction of semantic space models. *Computational Linguistics 33*(2), 161–199.

Peng, J., A. Feldman, and L. Street (2010). Computing linear discriminants for idiomatic sentence detection. *Research in Computing Science, Special issue: Natural Language Processing and its Applications 46*, 17–28.

Sag, I. A., T. Baldwin, F. Bond, A. Copestake, and D. Flickinger (2002). Multiword expressions: A Pain in the Neck for NLP. In *Proceedings of the 3rd International Conference on Intelligence Text Processing and Computational Linguistics (CICLing 2002)*, Mexico City, Mexico, pp. 1–15.

Seaton, M. and A. Macaulay (Eds.) (2002). *Collins COBUILD Idioms Dictionary* (second ed.). HarperCollins Publishers.

Sporleder, C. and L. Li (2009). Unsupervised Recognition of Literal and Non-literal Use of Idiomatic Expressions. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, Morristown, NJ, USA, pp. 754–762. Association for Computational Linguistics.

Warriner, A. B., V. Kuperman, and M. Brysbaert (2013). Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior Research Methods 44*(4).

Widdows, D. and B. Dorow (2005). Automatic extraction of idioms using graph analysis and asymmetric lexicosyntactic patterns. In *Proceedings of the ACL-SIGLEX Workshop on Deep Lexical Acquisition*, DeepLA '05, Stroudsburg, PA, USA, pp. 48–56. Association for Computational Linguistics.

Woods, A., P. Fletcher, and A. Hughes (1986). *Statistics in Language Studies*. Cambridge: Cambridge University Press.

Learning Spatial Knowledge for Text to 3D Scene Generation

Angel X. Chang, Manolis Savva and Christopher D. Manning
Stanford University
{angelx,msavva,manning}@cs.stanford.edu

Abstract

We address the grounding of natural language to concrete spatial constraints, and inference of implicit pragmatics in 3D environments. We apply our approach to the task of text-to-3D scene generation. We present a representation for common sense spatial knowledge and an approach to extract it from 3D scene data. In text-to-3D scene generation, a user provides as input natural language text from which we extract explicit constraints on the objects that should appear in the scene. The main innovation of this work is to show how to augment these explicit constraints with learned spatial knowledge to infer missing objects and likely layouts for the objects in the scene. We demonstrate that spatial knowledge is useful for interpreting natural language and show examples of learned knowledge and generated 3D scenes.

1 Introduction

To understand language, we need an understanding of the world around us. Language describes the world and provides symbols with which we represent meaning. Still, much knowledge about the world is so obvious that it is rarely explicitly stated. It is uncommon for people to state that chairs are usually on the floor and upright, and that you usually eat a cake from a plate on a table. Knowledge of such common facts provides the context within which people communicate with language. Therefore, to create practical systems that can interact with the world and communicate with people, we need to leverage such knowledge to interpret language in context.

Spatial knowledge is an important aspect of the world and is often not expressed explicitly in natural language. This is one of the biggest challenges in grounding language and enabling natural communication between people and intelligent systems. For instance, if we want a robot that can follow commands such as "bring me a piece of cake", it needs to be imparted with an understanding of likely locations for the cake in the kitchen and that the cake should be placed on a plate.

Figure 1: Generated scene for "There is a room with a chair and a computer." Note that the system infers the presence of a desk and that the computer should be supported by the desk.

The pioneering WordsEye system (Coyne and Sproat, 2001) addressed the text-to-3D task and is an inspiration for our work. However, there are many remaining gaps in this broad area. Among them, there is a need for research into learning spatial knowledge representations from data, and for connecting them to language. Representing unstated facts is a challenging problem unaddressed by prior work and the focus of our contribution. This problem is a counterpart to the image description problem (Kulkarni et al., 2011; Mitchell et al., 2012; Elliott and Keller, 2013), which has so far remained largely unexplored by the community.

We present a representation for this form of spatial knowledge that we learn from 3D scene data and connect to natural language. We will show how this representation is useful for grounding language and for inferring unstated facts, i.e., the pragmatics of language describing physical environments. We demonstrate the use of this representation in the task of text-to-3D scene genera-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2028–2038,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

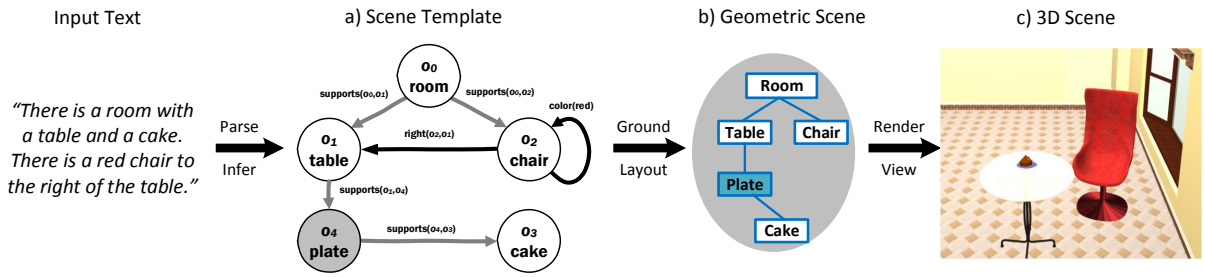


Figure 2: Overview of our spatial knowledge representation for text-to-3D scene generation. We parse input text into a scene template and infer implicit spatial constraints from learned priors. We then ground the template to a geometric scene, choose 3D models to instantiate and arrange them into a final 3D scene.

tion, where the input is natural language and the desired output is a 3D scene.

We focus on the text-to-3D task to demonstrate that extracting spatial knowledge is possible and beneficial in a challenging scenario: one requiring the grounding of natural language and inference of rarely mentioned implicit pragmatics based on spatial facts. Figure 1 illustrates some of the inference challenges in generating 3D scenes from natural language: the desk was not explicitly mentioned in the input, but we need to infer that the computer is likely to be supported by a desk rather than directly placed on the floor. Without this inference, the user would need to be much more verbose with text such as "There is a room with a chair, a computer, and a desk. The computer is on the desk, and the desk is on the floor. The chair is on the floor."

Contributions We present a spatial knowledge representation that can be learned from 3D scenes and captures the statistics of what objects occur in different scene types, and their spatial positions relative to each other. In addition, we model spatial relations (left, on top of, etc.) and learn a mapping between language and the geometric constraints that spatial terms imply. We show that using our learned spatial knowledge representation, we can infer implicit constraints, and generate plausible scenes from concise natural text input.

2 Task Definition and Overview

We define text-to-scene generation as the task of taking text that describes a scene as input, and generating a plausible 3D scene described by that text as output. More concretely, based on the input text, we select objects from a dataset of 3D models and arrange them to generate output scenes.

The main challenge we address is in transforming a scene template into a physically realizable 3D scene. For this to be possible, the system must be able to automatically specify the objects present and their position and orientation with respect to each other as constraints in 3D space. To do so, we need to have a representation of scenes (§3). We need good priors over the arrangements of objects in scenes (§4) and we need to be able to ground textual relations into spatial constraints (§5). We break down our task as follows (see Figure 2):

Template Parsing (§6.1): Parse the textual description of a scene into a set of constraints on the objects present and spatial relations between them.

Inference (§6.2): Expand this set of constraints by accounting for implicit constraints not specified in the text using learned spatial priors.

Grounding (§6.3): Given the constraints and priors on the spatial relations of objects, transform the scene template into a geometric 3D scene with a set of objects to be instantiated.

Scene Layout (§6.4): Arrange the objects and optimize their placement based on priors on the relative positions of objects and explicitly provided spatial constraints.

3 Scene Representation

To capture the objects present and their arrangement, we represent scenes as graphs where nodes are objects in the scene, and edges are semantic relationships between the objects.

We represent the semantics of a scene using a *scene template* and the geometric properties using a *geometric scene*. One critical property which is captured by our scene graph representation is that of a static support hierarchy, i.e., the order in which bigger objects physically support smaller ones: the floor supports tables, which support plates, which can support cakes. Static support and other constraints on relationships between objects are represented as edges in the scene graph.

Figure 3: Probabilities of different scene types given the presence of "knife" and "table".

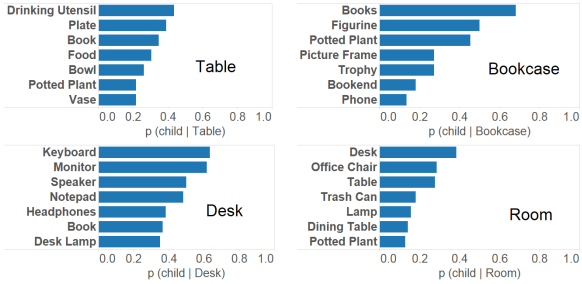


Figure 4: Probabilities of support for some most likely child object categories given four different parent object categories, from top left clockwise: dining table, bookcase, room, desk.

3.1 Scene Template

A scene template $\mathcal{T} = (\mathcal{O}, \mathcal{C}, C_s)$ consists of a set of object descriptions $\mathcal{O} = \{o_1, \ldots, o_n\}$ and constraints $\mathcal{C} = \{c_1, \ldots, c_k\}$ on the relationships between the objects. A scene template also has a scene type C_s.

Each object o_i, has properties associated with it such as category label, basic attributes such as color and material, and number of occurrences in the scene. For constraints, we focus on spatial relations between objects, expressed as predicates of the form *supported_by*(o_i, o_j) or *left*(o_i, o_j) where o_i and o_j are recognized objects.[1] Figure 2a shows an example scene template. From the scene template we instantiate concrete geometric 3D scenes. To infer implicit constraints on objects and spatial support we learn priors on object occurrences in 3D scenes (§4.1) and their support hierarchies (§4.2).

3.2 Geometric Scene

We refer to the concrete geometric representation of a scene as a "geometric scene". It consists of a set of 3D model instances – one for each object – that capture the appearance of the object. A transformation matrix that represents the position, orientation, and scaling of the object in a scene is also necessary to exactly position the object. We generate a geometric scene from a scene template by selecting appropriate models from a 3D model database and determining transformations that optimize their layout to satisfy spatial constraints. To inform geometric arrangement we learn priors on the types of support surfaces (§4.2) and the relative positions of objects (§4.4).

4 Spatial Knowledge

Our model of spatial knowledge relies on the idea of abstract scene types describing the occurrence and arrangement of different categories of objects within scenes of that type. For example, kitchens typically contain kitchen counters on which plates and cups are likely to be found. The type of scene and category of objects condition the spatial relationships that can exist in a scene.

We learn spatial knowledge from 3D scene data, basing our approach on that of Fisher et al. (2012) and using their dataset of 133 small indoor scenes created with 1723 Trimble 3D Warehouse models (Fisher et al., 2012).

4.1 Object Occurrence Priors

We learn priors for object occurrence in different scene types (such as kitchens, offices, bedrooms).

$$P_{occ}(C_o|C_s) = \frac{count(C_o \text{ in } C_s)}{count(C_s)}$$

This allows us to evaluate the probability of different scene types given lists of object occurring in them (see Figure 3). For example given input of the form "there is a knife on the table" then we are likely to generate a scene with a dining table and other related objects.

4.2 Support Hierarchy Priors

We observe the static support relations of objects in existing scenes to establish a prior over what objects go on top of what other objects. As an example, by observing plates and forks on tables most of the time, we establish that tables are more likely to support plates and forks than chairs. We estimate the probability of a parent category C_p supporting a given child category C_c as a simple conditional probability based on normalized observation counts.[2]

$$P_{support}(C_p|C_c) = \frac{count(C_c \text{ on } C_p)}{count(C_c)}$$

We show a few of the priors we learn in Figure 4 as likelihoods of categories of child objects being statically supported by a parent category object.

[1] Our representation can also support other relationships such as $larger(o_i, o_j)$.

[2] The support hierarchy is explicitly modeled in the scene dataset we use.

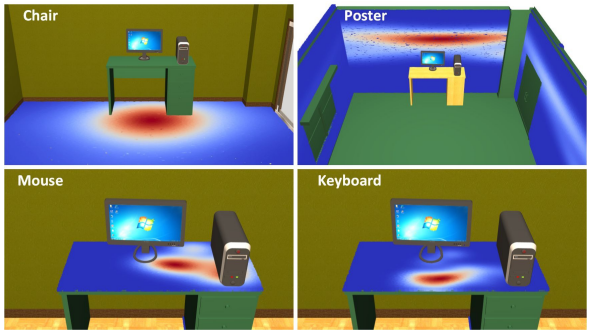


Figure 5: Predicted positions using learned relative position priors for *chair* given *desk* (top left), *poster-room* (top right), *mouse-desk* (bottom left), *keyboard-desk* (bottom right).

4.3 Support Surface Priors

To identify which surfaces on parent objects support child objects, we first segment parent models into planar surfaces using a simple region-growing algorithm based on (Kalvin and Taylor, 1996). We characterize support surfaces by the direction of their normal vector, limited to the six canonical directions: *up, down, left, right, front, back*. We learn a probability of supporting surface normal direction S_n given child object category C_c. For example, posters are typically found on walls so their support normal vectors are in the horizontal directions. Any unobserved child categories are assumed to have $P_{surf}(S_n = up|C_c) = 1$ since most things rest on a horizontal surface (e.g., floor).

$$P_{surf}(S_n|C_c) = \frac{count(C_c \text{ on surface with } S_n)}{count(C_c)}$$

4.4 Relative Position Priors

We model the relative positions of objects based on their object categories and current scene type: i.e., the relative position of an object of category C_{obj} is with respect to another object of category C_{ref} and for a scene type C_s. We condition on the relationship R between the two objects, whether they are siblings ($R = Sibling$) or child-parent ($R = ChildParent$).

$$P_{relpos}(x, y, \theta|C_{obj}, C_{ref}, C_s, R)$$

When positioning objects, we restrict the search space to points on the selected support surface. The position x, y is the centroid of the target object projected onto the support surface in the semantic frame of the reference object. The θ is the angle between the front of the two objects. We represent these relative position and orientation priors by performing kernel density estimation on the observed samples. Figure 5 shows predicted positions of objects using the learned priors.

Relation	$P(relation)$
inside(A,B)	$\frac{Vol(A \cap B)}{Vol(A)}$
outside(A,B)	1 - $\frac{Vol(A \cap B)}{Vol(A)}$
left_of(A,B)	$\frac{Vol(A \cap \text{left\_of}(B))}{Vol(A)}$
right_of(A,B)	$\frac{Vol(A \cap \text{right\_of}(B))}{Vol(A)}$
near(A,B)	$\mathbb{1}(dist(A, B) < t_{near})$
faces(A,B)	$\cos(front(A), c(B) - c(A))$

Table 1: Definitions of spatial relation using bounding boxes. Note: $dist(A, B)$ is normalized against the maximum extent of the bounding box of B. $front(A)$ is the direction of the front vector of A and $c(A)$ is the centroid of A.

Keyword	*Top Relations and Scores*
behind	$(back\_of, 0.46), (back\_side, 0.33)$
adjacent	$(front\_side, 0.27), (outside, 0.26)$
below	$(below, 0.59), (lower\_side, 0.38)$
front	$(front\_of, 0.41), (front\_side, 0.40)$
left	$(left\_side, 0.44), (left\_of, 0.43)$
above	$(above, 0.37), (near, 0.30)$
opposite	$(outside, 0.31), (next\_to, 0.30)$
on	$(supported\_by, 0.86), (on\_top\_of, 0.76)$
near	$(outside, 0.66), (near, 0.66)$
next	$(outside, 0.49), (near, 0.48)$
under	$(supports, 0.62), (below, 0.53)$
top	$(supported\_by, 0.65), (above, 0.61)$
inside	$(inside, 0.48), (supported\_by, 0.35)$
right	$(right\_of, 0.50), (lower\_side, 0.38)$
beside	$(outside, 0.45), (right\_of, 0.45)$

Table 2: Map of top keywords to spatial relations (appropriate mappings in **bold**).

5 Spatial Relations

We define a set of formal spatial relations that we map to natural language terms (§5.1). In addition, we collect annotations of spatial relation descriptions from people, learn a mapping of spatial keywords to our formal spatial relations, and train a classifier that given two objects can predict the likelihood of a spatial relation holding (§5.2).

5.1 Predefined spatial relations

For spatial relations we use a set of predefined relations: *left_of, right_of, above, below, front, back, supported_by, supports, next_to, near, inside, outside, faces, left_side, right_side*.[3] These are measured using axis-aligned bounding boxes from the viewer's perspective; the involved bounding boxes are compared to determine volume overlap or closest distance (for proximity relations; see Table 1).

[3] We distinguish *left_of(A,B)* as A being left of the left edge of the bounding box of B vs *left_side(A,B)* as A being left of the centroid of B.

Feature	#	*Description*
$delta(A, B)$	3	Delta position (x, y, z) between the centroids of A and B
$dist(A, B)$	1	Normalized distance (wrt B) between the centroids of A and B
$overlap(A, f(B))$	6	Fraction of A inside left/right/front/back/top/bottom regions wrt B: $\frac{Vol(A \cap f(B))}{Vol(A)}$
$overlap(A, B)$	2	$\frac{Vol(A \cap B)}{Vol(A)}$ and $\frac{Vol(A \cap B)}{Vol(B)}$
$support(A, B)$	2	$supported\_by(A, B)$ and $supports(A, B)$

Table 3: Features for trained spatial relations predictor.

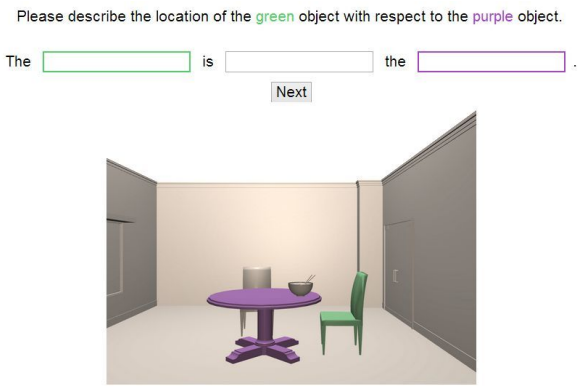


Figure 6: Our data collection task.

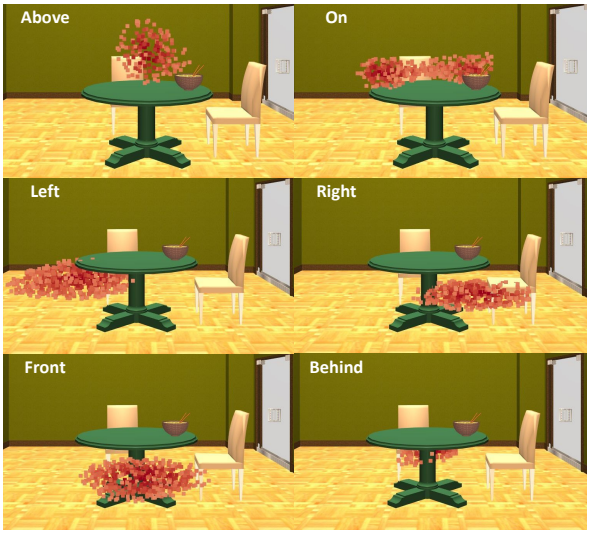


Figure 7: High probability regions where the center of another object would occur for some spatial relations with respect to a table: *above* (top left), *on* (top right), *left* (mid left), *right* (mid right), *in front* (bottom left), *behind* (bottom right).

Since these spatial relations are resolved with respect to the view of the scene, they correspond to view-centric definitions of spatial concepts.

5.2 Learning Spatial Relations

We collect a set of text descriptions of spatial relationships between two objects in 3D scenes by running an experiment on Amazon Mechanical Turk. We present a set of screenshots of scenes in our dataset that highlight particular pairs of objects and we ask people to fill in a spatial relationship of the form "The __ is __ the __" (see Fig 6). We collected a total of 609 annotations over 131 object pairs in 17 scenes. We use this data to learn priors on view-centric spatial relation terms and their concrete geometric interpretation.

For each response, we select one keyword from the text based on length. We learn a mapping of the top 15 keywords to our predefined set of spatial relations. We use our predefined relations on annotated spatial pairs of objects to create a binary indicator vector that is set to 1 if the spatial relation holds, or zero otherwise. We then create a similar vector for whether the keyword appeared in the annotation for that spatial pair, and then compute the cosine similarity of the two vectors to obtain a score for mapping keywords to spatial relations. Table 2 shows the obtained mapping. Using just the top mapping, we are able to map 10 of the 15 keywords to an appropriate spatial relation. The 5 keywords that are not well mapped are proximity relations that are not well captured by our predefined spatial relations.

Using the 15 keywords as our spatial relations, we train a log linear binary classifier for each keyword over features of the objects involved in that spatial relation (see Table 3). We then use this model to predict the likelihood of that spatial relation in new scenes.

Figure 7 shows examples of predicted likelihoods for different spatial relations with respect to an anchor object in a scene. Note that the learned spatial relations are much stricter than our predefined relations. For instance, "above" is only used to referred to the area directly above the table, not to the region above and to the left or above and in front (which our predefined classifier will all consider to be above). In our results, we show we have more accurate scenes using the trained spatial relations than the predefined ones.

Dependency Pattern	*Example Text*
`{tag:VBN}=verb >nsubjpass {}=nsubj >prep ({}=prep >pobj {}=pobj)`	The chair$_{\texttt{[nsubj]}}$ is made$_{\texttt{[verb]}}$ of$_{\texttt{[prep]}}$ wood$_{\texttt{[pobj]}}$.
attribute`(verb,pobj)(nsubj,pobj)`	`material(chair,wood)`
`{}=dobj >cop {} >nsubj {}=nsubj`	The chair$_{\texttt{[nsubj]}}$ is red$_{\texttt{[dobj]}}$.
attribute`(dobj)(nsubj,dobj)`	`color(chair,red)`
`{}=dobj >cop {} >nsubj {}=nsubj >prep ({}=prep >pobj {}=pobj)`	The table$_{\texttt{[nsubj]}}$ is next$_{\texttt{[dobj]}}$ to$_{\texttt{[prep]}}$ the chair$_{\texttt{[pobj]}}$.
spatial`(dobj)(nsubj, pobj)`	`next_to(table,chair)`
`{}=nsubj >advmod ({}=advmod >prep ({}=prep >pobj {}=pobj))`	There is a table$_{\texttt{[nsubj]}}$ next$_{\texttt{[advmod]}}$ to$_{\texttt{[prep]}}$ a chair$_{\texttt{[pobj]}}$.
spatial`(advmod)(nsubj, pobj)`	`next_to(table,chair)`

Table 4: Example dependency patterns for extracting attributes and spatial relations.

6 Text to Scene generation

We generate 3D scenes from brief scene descriptions using our learned priors.

6.1 Scene Template Parsing

During scene template parsing we identify the scene type, the objects present in the scene, their attributes, and the relations between them. The input text is first processed using the Stanford CoreNLP pipeline (Manning et al., 2014). The scene type is determined by matching the words in the utterance against a list of known scene types from the scene dataset.

To identify objects, we look for noun phrases and use the head word as the category, filtering with WordNet (Miller, 1995) to determine which objects are visualizable (under the physical object synset, excluding locations). We use the Stanford coreference system to determine when the same object is being referred to.

To identify properties of the objects, we extract other adjectives and nouns in the noun phrase. We also match dependency patterns such as "X is made of Y" to extract additional attributes. Based on the object category and attributes, and other words in the noun phrase mentioning the object, we identify a set of associated keywords to be used later for querying the 3D model database.

Dependency patterns are also used to extract spatial relations between objects (see Table 4 for some example patterns). We use Semgrex patterns to match the input text to dependencies (Chambers et al., 2007). The attribute types are determined from a dictionary using the text expressing the attribute (e.g., *attribute*(red)=color, *attribute*(round)=shape). Likewise, spatial relations are looked up using the learned map of keywords to spatial relations.

As an example, given the input "There is a room with a desk and a red chair. The chair is to the left of the desk." we extract the following objects and spatial relations:

Objects	category	attributes	keywords
o_0	room		room
o_1	desk		desk
o_2	chair	*color*:red	chair, red

Relations: $\mathit{left}(o_2, o_1)$

6.2 Inferring Implicits

From the parsed scene template, we infer the presence of additional objects and support constraints.

We can optionally infer the presence of additional objects from object occurrences based on the scene type. If the scene type is unknown, we use the presence of known object categories to predict the most likely scene type by using Bayes' rule on our object occurrence priors P_{occ} to get $P(C_s|\{C_o\}) \propto P_{occ}(\{C_o\}|C_s)P(C_s)$. Once we have a scene type C_s, we sample P_{occ} to find objects that are likely to occur in the scene. We restrict sampling to the top $n = 4$ object categories.

We can also use the support hierarchy priors $P_{support}$ to infer implicit objects. For instance, for each object o_i we find the most likely supporting object category and add it to our scene if not already present.

After inferring implicit objects, we infer the support constraints. Using the learned text to predefined relation mapping from §5.2, we can map the keywords "on" and "top" to the *supported_by* relation. We infer the rest of the support hierarchy by selecting for each object o_i the parent object o_j that maximizes $P_{support}(C_{o_j}|C_{o_i})$.

6.3 Grounding Objects

Once we determine from the input text what objects exist and their spatial relations, we select 3D models matching the objects and their associated properties. Each object in the scene template is grounded by querying a 3D models database with

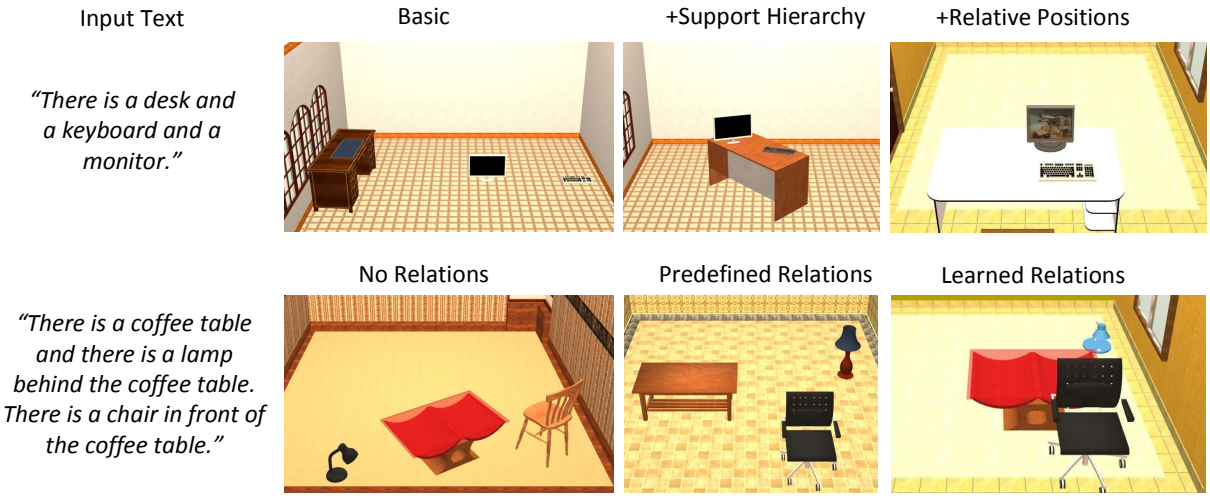


Figure 8: **Top** Generated scenes for randomly placing objects on the floor (*Basic*), with inferred *Support Hierarchy*, and with priors on *Relative Positions*. **Bottom** Generated scenes with no understanding of spatial relations (*No Relations*), scoring using *Predefined Relations* and *Learned Relations*.

the appropriate category and keywords.

We use a 3D model dataset collected from Google 3D Warehouse by prior work in scene synthesis and containing about 12490 mostly indoor objects (Fisher et al., 2012). These models have text associated with them in the form of names and tags. In addition, we semi-automatically annotated models with object category labels (roughly 270 classes). We used model tags to set these labels, and verified and augmented them manually.

In addition, we automatically rescale models so that they have physically plausible sizes and orient them so that they have a consistent up and front direction (Savva et al., 2014). We then indexed all models in a database that we query at run-time for retrieval based on category and tag labels.

6.4 Scene Layout

Once we have instantiated the objects in the scene by selecting models, we aim to optimize an overall layout score $\mathcal{L} = \lambda_{obj}\mathcal{L}_{obj} + \lambda_{rel}\mathcal{L}_{rel}$ that is a weighted sum of object arrangement $\mathcal{L}_{obj}$ score and constraint satisfaction $\mathcal{L}_{rel}$ score:

$$\mathcal{L}_{obj} = \sum_{o_i} P_{surf}(S_n|C_{o_i}) \sum_{o_j \in F(o_i)} P_{relpos}(\cdot)$$

$$\mathcal{L}_{rel} = \sum_{c_i} P_{rel}(c_i)$$

where $F(o_i)$ are the sibling objects and parent object of o_i. We use $\lambda_{obj} = 0.25$ and $\lambda_{rel} = 0.75$ for the results we present.

We use a simple hill climbing strategy to find a reasonable layout. We first initialize the positions of objects within the scene by traversing the support hierarchy in depth-first order, positioning the children from largest to first and recursing. Child nodes are positioned by first selecting a supporting surface on a candidate parent object through sampling of P_{surf}. After selecting a surface, we sample a position on the surface based on P_{relpos}. Finally, we check whether collisions exist with other objects, rejecting layouts where collisions occur. We iterate by randomly jittering and repositioning objects. If there are any spatial constraints that are not satisfied, we also remove and randomly reposition the objects violating the constraints, and iterate to improve the layout. The resulting scene is rendered and presented to the user.

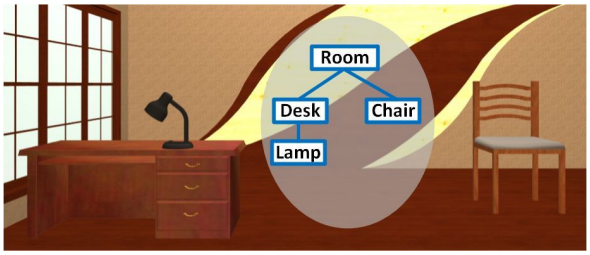


Figure 9: Generated scene for "There is a room with a desk and a lamp. There is a chair to the right of the desk." The inferred scene hierarchy is overlayed in the center.

7 Results and Discussion

We show examples of generated scenes, and compare against naive baselines to demonstrate learned priors are essential for scene generation. We

Figure 10: Generated scene for "There is a room with a poster bed and a poster."

Figure 11: Generated scene for "living room".

also discuss interesting aspects of using spatial knowledge in view-based object referent resolution (§7.2) and in disambiguating geometric interpretations of "on" (§7.3).

Model Comparison Figure 8 shows a comparison of scenes generated by our model versus several simpler baselines. The top row shows the impact of modeling the support hierarchy and the relative positions in the layout of the scene. The bottom row shows that the learned spatial relations can give a more accurate layout than the naive predefined spatial relations, since it captures pragmatic implicatures of language, e.g., left is only used for directly left and not top left or bottom left (Vogel et al., 2013).

Figure 12: **Left**: chair is selected using "the chair to the right of the table" or "the object to the right of the table". Chair is not selected for "the cup to the right of the table". **Right**: Different view results in different chair being selected for the input "the chair to the right of the table".

7.1 Generated Scenes

Support Hierarchy Figure 9 shows a generated scene along with the input text and support hierarchy. Even though the spatial relation between lamp and desk was not mentioned, we infer that the lamp is supported by the top surface of the desk.

Disambiguation Figure 10 shows a generated scene for the input "There is a room with a poster bed and a poster". Note that the system differentiates between a "poster" and a "poster bed" – it correctly selects and places the bed on the floor, while the poster is placed on the wall.

Inferring objects for a scene type Figure 11 shows an example of inferring all the objects present in a scene from the input "living room". Some of the placements are good, while others can clearly be improved.

7.2 View-centric object referent resolution

After a scene is generated, the user can refer to objects with their categories and with spatial relations between them. Objects are disambiguated by both category and view-centric spatial relations. We use the WordNet hierarchy to resolve hyponym or hypernym referents to objects in the scene. In Figure 12 (left), the user can select a chair to the right of the table using the phrase "chair to the right of the table" or "object to the right of the table". The user can then change their viewpoint by rotating and moving around. Since spatial relations are resolved with respect to the current viewpoint, a different chair is selected for the same phrase from a different viewpoint in the right screenshot.

7.3 Disambiguating "on"

As shown in §5.2, the English preposition "on", when used as a spatial relation, corresponds strongly to the *supported_by* relation. In our trained model, the *supported_by* feature also has a high positive weight for "on".

Our model for supporting surfaces and hierarchy allows interpreting the placement of "A on B" based on the categories of A and B. Figure 13 demonstrates four different interpretations for "on". Given the input "There is a cup on the table" the system correctly places the cup on the top surface of the table. In contrast, given "There is a cup on the bookshelf", the cup is placed on a supporting surface of the bookshelf, but not necessarily the top one which would be fairly high.

Figure 13: From top left clockwise: "There is a cup on the table", "There is a cup on the bookshelf", "There is a poster on the wall", "There is a hat on the chair". Note the different geometric interpretations of "on".

Given the input "There is a poster on the wall", a poster is pasted on the wall, while with the input "There is a hat on the chair" the hat is placed on the seat of the chair.

7.4 Limitations

While the system shows promise, there are still many challenges in text-to-scene generation. For one, we did not address the difficulties of resolving objects. A failure case of our system stems from using a fixed set of categories to identify visualizable objects. For example, the sense of "top" referring to a spinning top, and other uncommon object types, are not handled by our system as concrete objects. Furthermore, complex phrases including object parts such as "there's a coat on the seat of the chair" are not handled. Figure 14 shows some example cases where the context is important in selecting an appropriate object and the difficulties of interpreting noun phrases.

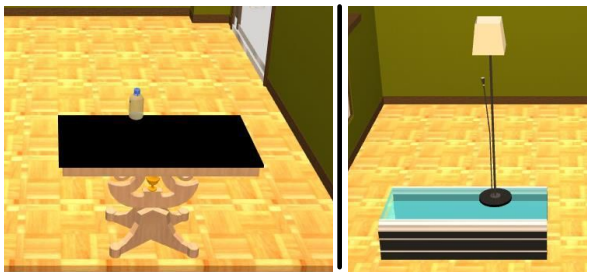


Figure 14: **Left**: A water bottle instead of wine bottle is selected for "There is a bottle of wine on the table in the kitchen". In addition, the selected table is inappropriate for a kitchen. **Right**: A floor lamp is incorrectly selected for the input "There is a lamp on the table".

In addition, we rely on a few dependency patterns for extracting spatial relations so robustness to variations in spatial language is lacking. We only handle binary spatial relations (e.g., "left", "behind") ignoring more complex relations such as "around the table" or "in the middle of the room". Though simple binary relations are some of the most fundamental spatial expressions and a good first step, handling more complex expressions will do much to improve the system.

Another issue is that the interpretation of sentences such as "the desk is covered with paper", which entails many pieces of paper placed on the desk, is hard to resolve. With a more data-driven approach we can hope to link such expressions to concrete facts.

Finally, we use a traditional pipeline approach for text processing, so errors in initial stages can propagate downstream. Failures in dependency parsing, part of speech tagging, or coreference resolution can result in incorrect interpretations of the input language. For example, in the sentence "there is a desk with a chair in front of it", "it" is not identified as coreferent with "desk" so we fail to extract the spatial relation `front_of(chair, desk)`.

8 Related Work

There is related prior work in the topics of modeling spatial relations, generating 3D scenes from text, and automatically laying out 3D scenes.

8.1 Spatial knowledge and relations

Prior work that required modeling spatial knowledge has defined representations specific to the task addressed. Typically, such knowledge is manually provided or crowdsourced – not learned from data. For instance, WordsEye (Coyne et al., 2010) uses a set of manually specified relations. The NLP community has explored grounding text to physical attributes and relations (Matuszek et al., 2012; Krishnamurthy and Kollar, 2013), generating text for referring to objects (FitzGerald et al., 2013) and connecting language to spatial relationships (Vogel and Jurafsky, 2010; Golland et al., 2010; Artzi and Zettlemoyer, 2013). Most of this work focuses on learning a mapping from text to formal representations, and does not model

implicit spatial knowledge. Many priors on real world spatial facts are typically unstated in text and remain largely unaddressed.

8.2 Text to Scene Systems

Early work on the SHRDLU system (Winograd, 1972) gives a good formalization of the linguistic manipulation of objects in 3D scenes. By restricting the discourse domain to a micro-world with simple geometric shapes, the SHRDLU system demonstrated parsing of natural language input for manipulating scenes. However, generalization to more complex objects and spatial relations is still very hard to attain.

More recently, a pioneering text-to-3D scene generation prototype system has been presented by WordsEye (Coyne and Sproat, 2001). The authors demonstrated the promise of text to scene generation systems but also pointed out some fundamental issues which restrict the success of their system: much spatial knowledge is required which is hard to obtain. As a result, users have to use unnatural language (e.g., "the stool is 1 feet to the south of the table") to express their intent. Follow up work has attempted to collect spatial knowledge through crowd-sourcing (Coyne et al., 2012), but does not address the learning of spatial priors.

We address the challenge of handling natural language for scene generation, by learning spatial knowledge from 3D scene data, and using it to infer unstated implicit constraints. Our work is similar in spirit to recent work on generating 2D clipart for sentences using probabilistic models learned from data (Zitnick et al., 2013).

8.3 Automatic Scene Layout

Work on scene layout has focused on determining good furniture layouts by optimizing energy functions that capture the quality of a proposed layout. These energy functions are encoded from design guidelines (Merrell et al., 2011) or learned from scene data (Fisher et al., 2012). Knowledge of object co-occurrences and spatial relations is represented by simple models such as mixtures of Gaussians on pairwise object positions and orientations. We leverage ideas from this work, but they do not focus on linking spatial knowledge to language.

9 Conclusion and Future Work

We have demonstrated a representation of spatial knowledge that can be learned from 3D scene data and how it corresponds to natural language. We also showed that spatial inference and grounding is critical for achieving plausible results in the text-to-3D scene generation task. Spatial knowledge is critically useful not only in this task, but also in other domains which require an understanding of the pragmatics of physical environments.

We only presented a deterministic approach for mapping input text to the parsed scene template. An interesting avenue for future research is to automatically learn how to parse text describing scenes into formal representations by using more advanced semantic parsing methods.

We can also improve the representation used for spatial priors of objects in scenes. For instance, in this paper we represented support surfaces by their orientation. We can improve the representation by modeling whether a surface is an interior or exterior surface.

Another interesting line of future work would be to explore the influence of object identity in determining when people use ego-centric or object-centric spatial reference models, and to improve resolution of spatial terms that have different interpretations (e.g., "the chair to the left of John" vs "the chair to the left of the table").

Finally, a promising line of research is to explore using spatial priors for resolving ambiguities during parsing. For example, the attachment of "next to" in "Put a lamp on the table next to the book" can be readily disambiguated with spatial priors such as the ones we presented.

Acknowledgments

We thank the anonymous reviewers for their thoughtful comments. We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*.

Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yeh, and Christopher D. Manning. 2007. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Bob Coyne and Richard Sproat. 2001. WordsEye: an automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*.

Bob Coyne, Richard Sproat, and Julia Hirschberg. 2010. Spatial relations in text-to-scene conversion. In *Computational Models of Spatial Language Interpretation, Workshop at Spatial Cognition*.

Bob Coyne, Alexander Klapheke, Masoud Rouhizadeh, Richard Sproat, and Daniel Bauer. 2012. Annotation tools and knowledge representation for a text-to-scene system. *Proceedings of COLING 2012: Technical Papers*.

Desmond Elliott and Frank Keller. 2013. Image description using visual dependency representations. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. 2012. Example-based synthesis of 3D object arrangements. *ACM Transactions on Graphics (TOG)*.

Nicholas FitzGerald, Yoav Artzi, and Luke Zettlemoyer. 2013. Learning distributions over logical forms for referring expression generation. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Alan D. Kalvin and Russell H. Taylor. 1996. Superfaces: Polygonal mesh simplification with bounded error. *Computer Graphics and Applications, IEEE*.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. 2011. Baby talk: Understanding and generating simple image descriptions. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Cynthia Matuszek, Nicholas Fitzgerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *International Conference on Machine Learning (ICML)*.

Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. 2011. Interactive furniture layout using interior design guidelines. In *ACM Transactions on Graphics (TOG)*.

George A. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*.

Margaret Mitchell, Xufeng Han, Jesse Dodge, Alyssa Mensch, Amit Goyal, Alex Berg, Kota Yamaguchi, Tamara Berg, Karl Stratos, and Hal Daumé III. 2012. Midge: Generating image descriptions from computer vision detections. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*.

Manolis Savva, Angel X. Chang, Gilbert Bernstein, Christopher D. Manning, and Pat Hanrahan. 2014. On being the right scale: Sizing large collections of 3D models. *Stanford University Technical Report CSTR 2014-03*.

Adam Vogel and Dan Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of ACL*.

Adam Vogel, Christopher Potts, and Dan Jurafsky. 2013. Implicatures and nested beliefs in approximate Decentralized-POMDPs. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Terry Winograd. 1972. Understanding natural language. *Cognitive psychology*.

C. Lawrence Zitnick, Devi Parikh, and Lucy Vanderwende. 2013. Learning the visual interpretation of sentences. In *IEEE Intenational Conference on Computer Vision (ICCV)*.

A Model of Coherence Based on Distributed Sentence Representation

Jiwei Li[1] **and Eduard Hovy**[3]
[1]Computer Science Department, Stanford University, Stanford, CA 94305, USA
[3]Language Technology Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
jiweil@stanford.edu ehovy@andrew.cmu.edu

Abstract

Coherence is what makes a multi-sentence text meaningful, both logically and syntactically. To solve the challenge of ordering a set of sentences into coherent order, existing approaches focus mostly on defining and using sophisticated features to capture the cross-sentence argumentation logic and syntactic relationships. But both argumentation semantics and cross-sentence syntax (such as coreference and tense rules) are very hard to formalize. In this paper, we introduce a neural network model for the coherence task based on distributed sentence representation. The proposed approach learns a syntactico-semantic representation for sentences automatically, using either recurrent or recursive neural networks. The architecture obviated the need for feature engineering, and learns sentence representations, which are to some extent able to capture the 'rules' governing coherent sentence structure. The proposed approach outperforms existing baselines and generates the state-of-art performance in standard coherence evaluation tasks[1].

1 Introduction

Coherence is a central aspect in natural language processing of multi-sentence texts. It is essential in generating readable text that the text planner compute which ordering of clauses (or sentences; we use them interchangeably in this paper) is likely to support understanding and avoid confusion. As Mann and Thompson (1988) define it,

> A text is coherent when it can be explained what role each clause plays with regard to the whole.

Several researchers in the 1980s and 1990s addressed the problem, the most influential of which include: Rhetorical Structure Theory (RST; (Mann and Thompson, 1988)), which defined about 25 relations that govern clause interdependencies and ordering and give rise to text tree structures; the stepwise assembly of semantic graphs to support adductive inference toward the best explanation (Hobbs et al., 1988); Discourse Representation Theory (DRT; (Lascarides and Asher, 1991)), a formal semantic model of discourse contexts that constrain coreference and quantification scoping; the model of intention-oriented conversation blocks and their stack-based queueing to model attention flow (Grosz and Sidner, 1986), and more recently an inventory of a hundred or so binary inter-clause relations and associated annotated corpus (Penn Discourse Treebank. Work in text planning implemented some of these models, especially operationalized RST (Hovy, 1988) and explanation relations (Moore and Paris, 1989) to govern the planning of coherent paragraphs. Other computational work defined so called schemas (McKeown, 1985), frames with fixed sequences of clause types to achieve stereotypical communicative intentions.

Little of this work survives. Modern research tries simply to order a collection of clauses or sentences without giving an account of which order(s) is/are coherent or what the overall text structure is. The research focuses on identifying and defining a set of increasingly sophisticated features by which algorithms can be trained to propose orderings. Features being explored include the clause entities, organized into a grid (Lapata and Barzilay, 2005; Barzilay and Lapata, 2008), coreference clues to ordering (Elsner and Charniak, 2008), named-entity categories (Eisner and Charniak, 2011), syntactic features (Louis and Nenkova, 2012), and others. Besides being time-intensive (feature engineering usually requites considerable

[1]Code available at `stanford.edu/~jiweil/` or by request from the first author.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2039–2048,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

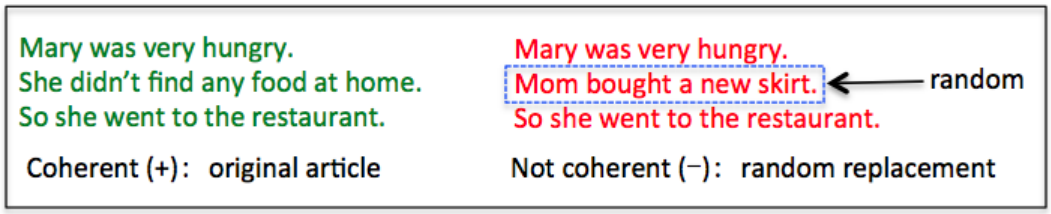


Figure 1: Illustrations of coherent (positive) vs not-coherent (negative) training examples.

effort and can depend greatly on upstream feature extraction algorithms), it is not immediately apparent which aspects of a clause or a coherent text to consider when deciding on ordering. More importantly, the features developed to date are still incapable of fully specifying the acceptable ordering(s) within a context, let alone describe why they are coherent.

Recently, deep architectures, have been applied to various natural language processing tasks (see Section 2). Such deep connectionist architectures learn a dense, low-dimensional representation of their problem in a hierarchical way that is capable of capturing both semantic and syntactic aspects of tokens (e.g., (Bengio et al., 2006)), entities, N-grams (Wang and Manning, 2012), or phrases (Socher et al., 2013). More recent researches have begun looking at higher level distributed representations that transcend the token level, such as sentence-level (Le and Mikolov, 2014) or even discourse-level (Kalchbrenner and Blunsom, 2013) aspects. Just as words combine to form meaningful sentences, can we take advantage of distributional semantic representations to explore the composition of sentences to form coherent meanings in paragraphs?

In this paper, we demonstrate that it is feasible to discover the coherent structure of a text using distributed sentence representations learned in a deep learning framework. Specifically, we consider a WINDOW approach for sentences, as shown in Figure 1, where positive examples are windows of sentences selected from original articles generated by humans, and negatives examples are generated by random replacements[2]. The semantic representations for terms and sentences are obtained through optimizing the neural network framework based on these positive vs negative examples and the proposed model produces state-of-art performance in multiple standard evaluations for coherence models (Barzilay and Lee, 2004).

The rest of this paper is organized as follows: We describe related work in Section 2, then describe how to obtain a distributed representation for sentences in Section 3, and the window composition in Section 4. Experimental results are shown in Section 5, followed by a conclusion.

2 Related Work

Coherence In addition to the early computational work discussed above, local coherence was extensively studied within the modeling framework of Centering Theory (Grosz et al., 1995; Walker et al., 1998; Strube and Hahn, 1999; Poesio et al., 2004), which provides principles to form a coherence metric (Miltsakaki and Kukich, 2000; Hasler, 2004). Centering approaches suffer from a severe dependence on manually annotated input.

A recent popular approach is the entity grid model introduced by Barzilay and Lapata (2008) , in which sentences are represented by a vector of discourse entities along with their grammatical roles (e.g., subject or object). Probabilities of transitions between adjacent sentences are derived from entity features and then concatenated to a document vector representation, which is used as input to machine learning classifiers such as SVM. Many frameworks have extended the entity approach, for example, by pre-grouping entities based on semantic relatedness (Filippova and Strube, 2007) or adding more useful types of features such as coreference (Elsner and Charniak, 2008), named entities (Eisner and Charniak, 2011), and discourse relations (Lin et al., 2011).

Other systems include the global graph model (Guinaudeau and Strube, 2013) which projects entities into a global graph. Louis and Nenkova (2012) introduced an HMM system in which the coherence between adjacent sentences is modeled by a hidden Markov framework captured by the

[2]Our approach is inspired by Collobert et al.'s idea (2011) that a word and its context form a positive training sample while a random word in that same context gives a negative training sample, when training word embeddings in the deep learning framework.

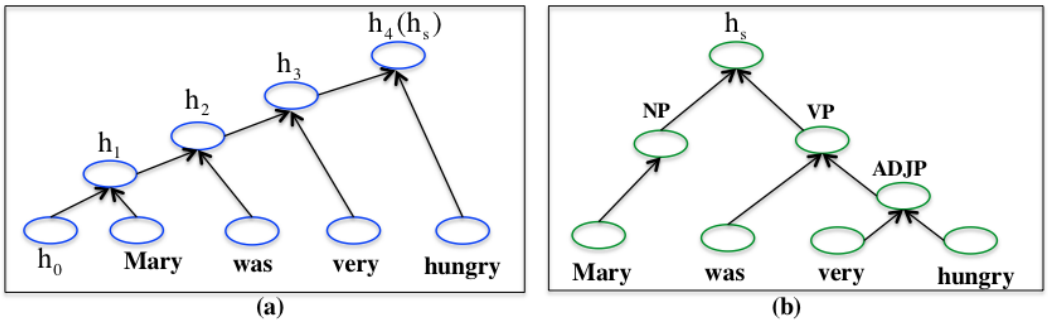


Figure 2: Sentential compositionality obtained from (a) recurrent / (b) recursive neural network. The bottom layer represents word vectors in the sentence. The top layer h_s denotes the resulting sentence vector.

transition rules of different topics.

Recurrent and Recursive Neural Networks In the context of NLP, recurrent neural networks view a sentence as a sequence of tokens and incorporate information from the past (i.e., preceding tokens) (Schuster and Paliwal, 1997; Sutskever et al., 2011) for acquisition of the current output. At each step, the recurrent network takes as input both the output of previous steps and the current token, convolutes the inputs, and forwards the result to the next step. It has been successfully applied to tasks such as language modeling (Mikolov et al., 2010) and spoken language understanding (Mesnil et al., 2013). The advantage of recurrent network is that it does not depend on external deeper structure (e.g., parse tree) and is easy to implement. However, in the recurrent framework, long-distance dependencies are difficult to capture due to the vanishing gradient problem (Bengio et al., 1994); two tokens may be structurally close to each other, even though they are far away in word sequence[3].

Recursive neural networks comprise another class of architecture, one that relies and operates on structured inputs (e.g., parse trees). It computes the representation for each parent based on its children iteratively in a bottom-up fashion. A series of variations have been proposed, each tailored to different task-specific requirements, such as Matrix-Vector RNN (Socher et al., 2012) that represents every word as both a vector and a matrix, or Recursive Neural Tensor Networks (Socher et al., 2013) that allow the model to have greater interactions between the input vectors. Many tasks have benefited from this recursive framework, including parsing (Socher et al., 2011b), sentiment analysis (Socher et al., 2013), and paraphrase detection (Socher et al., 2011a).

[3]For example, a verb and its corresponding direct object can be far away in terms of tokens if many adjectives lies in between, but they are adjacent in the parse tree (Irsoy and Cardie, 2013).

2.1 Distributed Representations

Both recurrent and recursive networks require a vector representation of each input token. Distributed representations for words were first proposed in (Rumelhart et al., 1988) and have been successful for statistical language modeling (Elman, 1990). Various deep learning architectures have been explored to learn these embeddings in an unsupervised manner from a large corpus (Bengio et al., 2006; Collobert and Weston, 2008; Mnih and Hinton, 2007; Mikolov et al., 2013), which might have different generalization capabilities and are able to capture the semantic meanings depending on the specific task at hand. These vector representations can to some extent capture interesting semantic relationships, such as $King - man \approx Queue - woman$ (Mikolov et al., 2010), and recently have been successfully used in various NLP applications, including named entity recognition, tagging, segmentation (Wang et al., 2013), and machine translation (e.g.,(Collobert and Weston, 2008; Zou et al., 2013)).

3 Sentence Model

In this section, we demonstrate the strategy adopted to compute a vector for a sentence given the sequence of its words and their embeddings. We implemented two approaches, Recurrent and Recursive neural networks, following the descriptions in for example (Mikolov et al., 2010; Sutskever et al., 2011; Socher et al., 2013). As

the details of both approaches can be readily found there, we make this section brief and omit the details for brevity.

Let s denote a sentence, comprised of a sequence of words $s = \{w_1, w_2, ..., w_{n_s}\}$, where n_s denotes the number of words within sentence s. Each word w is associated with a specific vector embedding $\mathbf{e_w} = \{e_w^1, e_w^2, ..., e_w^K\}$, where K denotes the dimension of the word embedding. We wish to compute the vector representation for current sentence $h_s = \{h_s^1, h_s^2, ..., h_s^K\}$.

Recurrent Sentence Representation (Recurrent) The recurrent network captures certain general considerations regarding sentential compositionality. As shown in Figure 2 (a), for sentence s, recurrent network successively takes word w_i at step i, combines its vector representation e_w^t with former input h_{i-1} from step $i-1$, calculates the resulting current embedding h_t, and passes it to the next step. The standard recurrent network calculates h_t as follows:

$$h_t = f(V_{Recurrent} \cdot h_{t-1} + W_{Recurrent} \cdot e_w^t + b_{Recurrent}) \quad (1)$$

where $W_{Recurrent}$ and $V_{Recurrent}$ are $K \times K$ matrixes. $b_{Recurrent}$ denotes $K \times 1$ bias vector and $f = tanh$ is a standard element-wise nonlinearity.

Note that calculation for representation at time $t = 1$ is given by:

$$h_1 = f(V_{Recurrent} \cdot h_0 + W_{Recurrent} \cdot e_w^1 + b_{Recurrent}) \quad (2)$$

where h_0 denotes the global sentence starting vector.

Recursive Sentence Representation (Recursive) Recursive sentence representation relies on the structure of parse trees, where each leaf node of the tree corresponds to a word from the original sentence. It computes a representation for each parent node based on its immediate children recursively in a bottom-up fashion until reaching the root of the tree. Concretely, for a given parent p in the tree and its two children c_1 (associated with vector representation h_{c_1}) and c_2 (associated with vector representation h_{c_2}), standard recursive networks calculates h_p for p as follows:

$$h_p = f(W_{Recursive} \cdot [h_{c_1}, h_{c_2}] + b_{Recursive}) \quad (3)$$

where $[h_{c_1}, h_{c_2}]$ denotes the concatenating vector for children vector representation h_{c_1} and h_{c_2}. $W_{Recursive}$ is a $K \times 2K$ matrix and $b_{Recursive}$ is the $1 \times K$ bias vector. $f(\cdot)$ is tanh function.

Recursive neural models compute parent vectors iteratively until the root node's representation is obtained, and use the root embedding to represent the whole sentence, as shown in Figure 2 (b).

4 Coherence Model

The proposed coherence model adopts a window approach (Collobert et al., 2011), in which we train a three-layer neural network based on a sliding windows of L sentences.

4.1 Sentence Convolution

We treat a window of sentences as a clique C and associate each clique with a tag y_C that takes the value 1 if coherent, and 0 otherwise[4]. As shown in Figure 1, cliques taken from original articles are treated as coherent and those with sentences randomly replaced are used as negative examples. .

The sentence convolution algorithm adopted in this paper is defined by a three-layer neural network, i.e., sentence-level input layer, hidden layer, and overall output layer as shown in Figure 3. Formally, each clique C takes as input a $(L \times K) \times 1$ vector h_C by concatenating the embeddings of all its contained sentences, denoted as $h_C = [h_{s_1}, h_{s_2}, ..., h_{s_L}]$. (Note that if we wish to classify the first and last sentences and include their context, we require special beginning and ending sentence vectors, which are defined as $h_{<S>}$ for s_{start} and $h_{</S>}$ for s_{end} respectively.)

Let H denote the number of neurons in the hidden (second) layer. Then each of the hidden layers takes as input h_C and performs the convolution using a non-linear tanh function, parametrized by W_{sen} and b_{sen}. The concatenating output vector for hidden layers, defined as q_C, can therefore be rewritten as:

$$q_C = f(W_{sen} \times h_C + b_{sen}) \quad (4)$$

where W_{sen} is a $H \times (L \times K)$ dimensional matrix and b_{sen} is a $H \times 1$ dimensional bias vector.

[4] instead of a binary classification (correct/incorrect), another commonly used approach is the contrastive approach that minimizes the score function $max(0, 1 - s + s_c)$ (Collobert et al., 2011; Smith and Eisner, 2005). s denotes the score of a true (coherent) window and s_c the score of a corrupt (containing incoherence) one) in an attempt to make the score of true windows larger and corrupt windows smaller. We tried the contrastive one for both recurrent and recursive networks but the binary approach constantly outperformed the contrastive one in this task.

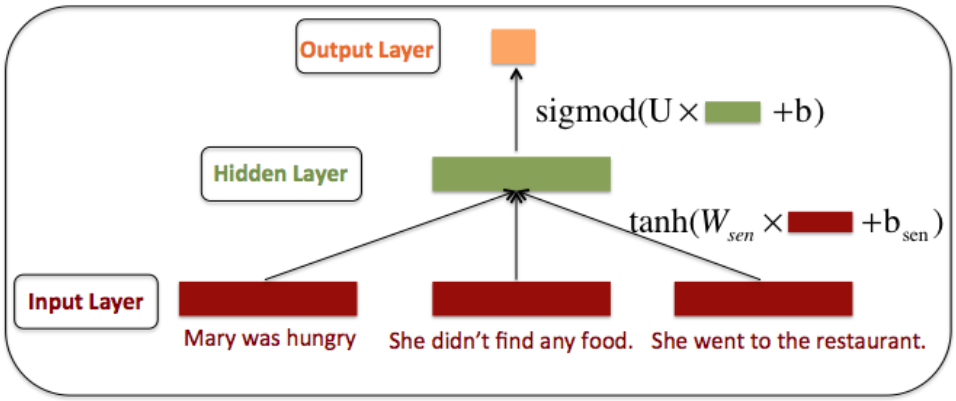


Figure 3: An example of coherence model based on a window of sentences (clique).

The output layer takes as input q_C and generates a scalar using linear function $U^T q_C + b$. A sigmod function is then adopted to project the value to a [0,1] probability space, which can be interpreted as the probability of whether one clique is coherent or not. The execution at the output layer can be summarized as:

$$p(y_C = 1) = \text{sigmod}(U^T q_C + b) \tag{5}$$

where U is an $H \times 1$ vector and b denotes the bias.

4.2 Training

In the proposed framework, suppose we have M training samples, the cost function for recurrent neural network with regularization on the training set is given by:

$$J(\Theta) = \frac{1}{M} \sum_{C \in trainset} \{-y_C \log[p(y_C = 1)] - (1 - y_C) \log[1 - p(y_C = 1)]\} + \frac{Q}{2M} \sum_{\theta \in \Theta} \theta^2 \tag{6}$$

where

$$\Theta = [W_{Recurrent}, W_{sen}, U_{sen}]$$

The regularization part is paralyzed by Q to avoid overfitting. A similar loss function is applied to the recursive network with only minor parameter altering that is excluded for brevity.

To minimize the objective $J(\Theta)$, we use the diagonal variant of AdaGrad (Duchi et al., 2011) with minibatches, which is widely applied in deep learning literature (e.g.,(Socher et al., 2011a; Pei et al., 2014)). The learning rate in AdaGrad is adapting differently for different parameters at different steps. Concretely, for parameter updates, let g_τ^i denote the subgradient at time step for parameter θ_i, which is obtained from backpropagation[5], the parameter update at time step t is given by:

$$\theta_\tau = \theta_{\tau-1} - \frac{\alpha}{\sum_{t=0}^{\tau} \sqrt{g_\tau^{i2}}} g_\tau^i \tag{7}$$

where α denotes the learning rate and is set to 0.01 in our approach. Optimal performance is achieved when batch size is set between 20 and 30.

4.3 Initialization

Elements in W_{sen} are initialized by randomly drawing from the uniform distribution $[-\epsilon, \epsilon]$, where $\epsilon = \frac{\sqrt{6}}{\sqrt{H+K \times L}}$ as suggested in (Collobert et al., 2011). $W_{recurrent}$, $V_{recurrent}$, $W_{recursive}$ and h_0 are initialized by randomly sampling from a uniform distribution $U(-0.2, 0.2)$. All bias vectors are initialized with 0. Hidden layer number H is set to 100.

Word embeddings $\{e\}$ are borrowed from Senna (Collobert et al., 2011; Collobert, 2011). The dimension for these embeddings is 50.

5 Experiments

We evaluate the proposed coherence model on two common evaluation approaches adopted in existing work (Barzilay and Lapata, 2008; Louis and Nenkova, 2012; Elsner et al., 2007; Lin et al., 2011): Sentence Ordering and Readability Assessment.

5.1 Sentence Ordering

We follow (Barzilay and Lapata, 2008; Louis and Nenkova, 2012; Elsner et al., 2007; Lin et al.,

[5]For more details on backpropagation through RNNs, see Socher et al. (2010).

2011) that all use pairs of articles, one containing the original document order and the other a random permutation of the sentences from the same document. The pairwise approach is predicated on the assumption that the original article is always more coherent than a random permutation; this assumption has been verified in Lin et al.'s work (2011).

We need to define the coherence score S_d for a given document d, where d is comprised of a series of sentences, $d = \{s_1, s_2, .., s_{N_d}\}$, and N_d denotes the number of sentences within d. Based on our clique definition, document d is comprised of N_d cliques. Taking window size $L = 3$ as example, cliques generated from document d appear as follows:

$$< s_{start}, s_1, s_2 >, < s_1, s_2, s_3 >, ..., \\ < s_{N_d-2}, s_{N_d-1}, s_{N_d} >, < s_{N_d-1}, s_{N_d}, s_{end} >$$

The coherence score for a given document S_d is the probability that all cliques within d are coherent, which is given by:

$$S_d = \prod_{C \in d} p(y_C = 1) \quad (8)$$

For document pair $< d_1, d_2 >$ in our task, we would say document d_1 is more coherent than d_2 if

$$S_{d_1} > S_{d_2} \quad (9)$$

5.1.1 Dataset

We use two corpora that are widely employed for coherence prediction (Barzilay and Lee, 2004; Barzilay and Lapata, 2008; Elsner et al., 2007). One contains reports on airplane accidents from the National Transportation Safety Board and the other contains reports about earthquakes from the Associated Press. These articles are about 10 sentences long and usually exhibit clear sentence structure. For preprocessing, we only lowercase the capital letters to match with tokens in Senna word embeddings. In the recursive network, sentences are parsed using the Stanford Parser[6] and then transformed into binary trees. The accident corpus ends up with a vocabulary size of 4758 and an average of 10.6 sentences per document. The earthquake corpus contains 3287 distinct terms and an average of 11.5 sentences per document.

For each of the two corpora, we have 100 articles for training and 100 (accidents) and 99 (earthquakes) for testing. A maximum of 20 random permutations were generated for each test article to create the pairwise data (total of 1986 test pairs for the accident corpus and 1956 for earthquakes)[7].

Positive cliques are taken from original training documents. For easy training, rather than creating negative examples by replacing centered sentences randomly, the negative dataset contains cliques where centered sentences are replaced only by other sentences within the same document.

5.1.2 Training and Testing

Despite the numerous parameters in the deep learning framework, we tune only two principal ones for each setting: window size L (tried on $\{3, 5, 7\}$) and regularization parameter Q (tried on $\{0.01, 0.1, 0.25, 0.5, 1.0, 1.25, 2.0, 2.5, 5.0\}$). We trained parameters using 10-fold cross-validation on the training data. Concretely, in each setting, 90 documents were used for training and evaluation was done on the remaining articles, following (Louis and Nenkova, 2012). After tuning, the final model was tested on the testing set.

5.1.3 Model Comparison

We report performance of recursive and recurrent networks. We also report results from some popular approaches in the literature, including:

Entity Grid Model : Grid model (Barzilay and Lapata, 2008) obtains the best performance when coreference resolution, expressive syntactic information, and salience-based features are incorporated. Entity grid models represent each sentence as a column of a grid of features and apply machine learning methods (e.g., SVM) to identify the coherent transitions based on entity features (for details of entity models see (Barzilay and Lapata, 2008)). Results are directly taken from Barzilay and Lapata's paper (2008).

HMM : Hidden-Markov approach proposed by Louis and Nenkova (2012) to model the state (cluster) transition probability in the coherent context using syntactic features. Sentences need to be clustered in advance where the number of clusters is tuned as a parameter. We directly take

[6] http://nlp.stanford.edu/software/lex-parser.shtml

[7] Permutations are downloaded from http://people.csail.mit.edu/regina/coherence/CLsubmission/.

	Acci	Earthquake	Average
Recursive	0.864	0.976	0.920
Recurrent	0.840	0.951	0.895
Entity Grid	0.904	0.872	0.888
HMM	0.822	0.938	0.880
HMM+Entity	0.842	0.911	0.877
HMM+Content	0.742	0.953	0.848
Graph	0.846	0.635	0.740

Table 1: Comparison of Different Coherence Frameworks. Reported baseline results are among the best performance regarding each approach is reprinted from prior work from (Barzilay and Lapata, 2008; Louis and Nenkova, 2012; Guinaudeau and Strube, 2013).

the results from Louis and Nenkova's paper and report the best results among different combinations of parameter and feature settings[8]. We also report performances of models from Louis and Nenkova's work that combine HMM and entity/content models in a unified framework.

Graph Based Approach : Guinaudeau and Strube (2013) extended the entity grid model to a bipartite graph representing the text, where the entity transition information needed for local coherence computation is embedded in the bipartite graph. The Graph Based Approach outperforms the original entity approach in some of feature settings (Guinaudeau and Strube, 2013).

As can be seen in Table 1, the proposed frameworks (both recurrent and recursive) obtain state-of-art performance and outperform all existing baselines by a large margin. One interpretation is that the abstract sentence vector representations computed by the deep learning framework is more powerful in capturing exactly the relevant the semantic/logical/syntactic features in coherent contexts than features or other representations developed by human feature engineering are.

Another good quality of the deep learning framework is that it can be trained easily and makes unnecessary the effort required of feature engineering. In contrast, almost all existing baselines and other coherence methods require sophisticated feature selection processes and greatly rely on external feature extraction algorithm.

The recurrent network is easier to implement than the recursive network and does not rely on external resources (i.e., parse trees), but the recursive network obtains better performance by building the convolution on parse trees rather than simply piling up terms within the sentence, which is in line with common expectation.

Both recurrent and recursive models obtain better performance on the Earthquake than the Accident dataset. Scrutiny of the corpus reveals that articles reporting earthquakes exhibit a more consistent structure: earthquake outbreak, describing the center and intensity of the earthquake, injuries and rescue operations, etc., while accident articles usually exhibit more diverse scenarios.

5.2 Readability Assessment

Barzilay and Lapata (2008) proposed a readability assessment task for stylistic judgments about the difficulty of reading a document. Their approach combines a coherence system with Schwarm and Ostendorf's (2005) readability features to classify documents into two categories, more readable (coherent) documents and less readable ones. The evaluation accesses the ability to differentiate "easy to read" documents from difficult ones of each model.

5.2.1 Dataset

Barzilay and Lapata's (2008) data corpus is from the $Encyclopedia\ Britannica$ and the $Britannica\ Elementary$, the latter being a new version targeted at children. Both versions contain 107 articles. The $Encyclopedia\ Britannica$ corpus contains an average of 83.1 sentences per document and the $Britannica\ Elementary$ contains 36.6. The encyclopedia lemmas are written by different authors and consequently vary considerably in structure and vocabulary choice. Early researchers assumed that the children version ($Britannica\ Elementary$) is easier to read, hence more coherent than documents in $Encyclopedia\ Britannica$. This is a somewhat questionable assumption that needs further investigation.

5.2.2 Training and Testing

Existing coherence approaches again apply a pairwise ranking strategy and the article associated with the higher score is considered to be the more readable. As the replacement strategy for generating negative example is apparently not well fitted to this task, we adopted the following training framework: we use all sliding windows of sentences from coherent documents (documents from $Britannica\ Elementary$) as positive examples,

[8]The details for information about parameter and feature of best setting can be found in (Louis and Nenkova, 2012).

Approach	Accuracy
Recurrent	0.803
Recursive	0.828
Graph Approach	0.786
Entity	0.509
S&O	0.786
Entity+S&O	0.888

Table 2: Comparison of Different Coherence Frameworks on Readability Assessment. Reported baselines results are are taken from (Barzilay and Lapata, 2008; Guinaudeau and Strube, 2013). S&O: Schwarm and Ostendorf (2005).

and cliques from $Encyclopedia\ Britannica$ as negative examples, and again apply Eq. 6 for training and optimization. During testing, we turn to Equations 8 and 9 for pairwise comparison. We adopted five-fold cross-validation in the same way as in (Barzilay and Lapata, 2008; Guinaudeau and Strube, 2013) for fair comparison. Parameters were tuned within each training set also using five-fold cross-validation. Parameters to tune included window size L and regularization parameter Q.

5.3 Results

We report results of the proposed approaches in the work along with entity model (Barzilay and Lapata, 2008) and graph based approach (Elsner and Charniak, 2008) in Table 2. The tabs shows that deep learning approaches again significantly outperform Entry and Global Approach baselines and are nearly comparable to the combination of entity and S&O features. Again, the recursive network outperforms the recurrent network in this task.

6 Conclusion

In this paper, we apply two neural network approaches to the sentence-ordering (coherence) task, using compositional sentence representations learned by recurrent and recursive composition. The proposed approach obtains state-of-art performance on the standard coherence evaluation tasks.

Acknowledgements

The authors want to thank Richard Socher and Pradeep Dasigi for the clarification of deep learning techniques. We also thank the three anonymous EMNLP reviewers for helpful comments.

References

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL*, pages 113–120.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

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

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics*, number EPFL-CONF-192374.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Micha Eisner and Eugene Charniak. 2011. Extending the entity grid with entity-specific features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 125–129. Association for Computational Linguistics.

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

Micha Elsner and Eugene Charniak. 2008. Coreference-inspired coherence modeling. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 41–44. Association for Computational Linguistics.

Micha Elsner, Joseph L Austerweil, and Eugene Charniak. 2007. A unified local and global model for discourse coherence. In *HLT-NAACL*, pages 436–443.

Katja Filippova and Michael Strube. 2007. Extending the entity-grid coherence model to semantically related entities. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, pages 139–142. Association for Computational Linguistics.

Barbara J Grosz and Candace L Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational linguistics*, 12(3):175–204.

Barbara J Grosz, Scott Weinstein, and Aravind K Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational linguistics*, 21(2):203–225.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 93–103.

Laura Hasler. 2004. An investigation into the use of centering transitions for summarisation. In *Proceedings of the 7th Annual CLUK Research Colloquium*, pages 100–107.

Jerry R Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards. 1988. Interpretation as abduction. In *Proceedings of the 26th annual meeting on Association for Computational Linguistics*, pages 95–103. Association for Computational Linguistics.

Eduard H Hovy. 1988. Planning coherent multisentential text. In *Proceedings of the 26th annual meeting on Association for Computational Linguistics*, pages 163–169. Association for Computational Linguistics.

Ozan Irsoy and Claire Cardie. 2013. Bidirectional recursive neural networks for token-level labeling with structure. *arXiv preprint arXiv:1312.0493*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *arXiv preprint arXiv:1306.3584*.

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In *IJCAI*, volume 5, pages 1085–1090.

Alex Lascarides and Nicholas Asher. 1991. Discourse relations and defeasible knowledge. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, pages 55–62. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 997–1006. Association for Computational Linguistics.

Annie Louis and Ani Nenkova. 2012. A coherence model based on syntactic patterns. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1157–1168. Association for Computational Linguistics.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Kathleen R McKeown. 1985. Discourse strategies for generating natural-language text. *Artificial Intelligence*, 27(1):1–41.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. Interspeech.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

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

Eleni Miltsakaki and Karen Kukich. 2000. The role of centering theory's rough-shift in the teaching and evaluation of writing skills. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 408–415. Association for Computational Linguistics.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

Johanna D Moore and Cecile L Paris. 1989. Planning text for advisory dialogues. In *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, pages 203–211. Association for Computational Linguistics.

Wenzhe Pei, Tao Ge, and Chang Baobao. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of ACL*.

Massimo Poesio, Rosemary Stevenson, Barbara Di Eugenio, and Janet Hitzeman. 2004. Centering: A parametric theory and its instantiations. *Computational linguistics*, 30(3):309–363.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. *Learning representations by back-propagating errors*. MIT Press, Cambridge, MA, USA.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681.

Sarah E Schwarm and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics.

Noah A Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362. Association for Computational Linguistics.

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, volume 24, pages 801–809.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011b. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 129–136.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

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

Michael Strube and Udo Hahn. 1999. Functional centering: Grounding referential coherence in information structure. *Computational linguistics*, 25(3):309–344.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Marilyn A Walker, Aravind Krishna Joshi, and Ellen Friedman Prince. 1998. *Centering theory in discourse*. Oxford University Press.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Houfeng Wang, Longkai Zhang, Li Li, He Zhengyan, and Ni Sun. 2013. Improving chinese word segmentation on micro-blog using rich punctuations.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*.

Discriminative Reranking of Discourse Parses Using Tree Kernels

Shafiq Joty and **Alessandro Moschitti**
ALT Research Group
Qatar Computing Research Institute
{sjoty,amoschitti}@qf.org.qa

Abstract

In this paper, we present a discriminative approach for reranking discourse trees generated by an existing probabilistic discourse parser. The reranker relies on tree kernels (TKs) to capture the global dependencies between discourse units in a tree. In particular, we design new computational structures of discourse trees, which combined with standard TKs, originate novel discourse TKs. The empirical evaluation shows that our reranker can improve the state-of-the-art sentence-level parsing accuracy from 79.77% to 82.15%, a relative error reduction of 11.8%, which in turn pushes the state-of-the-art document-level accuracy from 55.8% to 57.3%.

1 Introduction

Clauses and sentences in a well-written text are interrelated and exhibit a coherence structure. Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) represents the coherence structure of a text by a labeled tree, called discourse tree (DT) as shown in Figure 1. The leaves correspond to contiguous clause-like units called elementary discourse units (EDUs). Adjacent EDUs and larger discourse units are hierarchically connected by coherence relations (e.g., ELABORATION, CAUSE). Discourse units connected by a relation are further distinguished depending on their relative importance: *nuclei* are the core parts of the relation while *satellites* are the supportive ones.

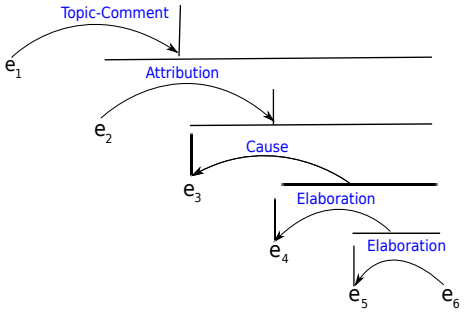
Conventionally, discourse analysis in RST involves two subtasks: (*i*) *discourse segmentation:* breaking the text into a sequence of EDUs, and (*ii*) *discourse parsing:* linking the discourse units to form a labeled tree. Despite the fact that discourse analysis is central to many NLP applications, the state-of-the-art document-level discourse parser (Joty et al., 2013) has an f-score of only 55.83% using manual discourse segmentation on the RST Discourse Treebank (RST-DT).

Although recent work has proposed rich linguistic features (Feng and Hirst, 2012) and powerful parsing models (Joty et al., 2012), discourse parsing remains a hard task, partly because these approaches do not consider global features and long range structural dependencies between DT constituents. For example, consider the human-annotated DT (Figure 1a) and the DT generated by the discourse parser of Joty et al. (2013) (Figure 1b) for the same text. The parser makes a mistake in finding the right structure: it considers only e_3 as the text to be attributed to e_2, where all the text spans from e_3 to e_6 (linked by CAUSE and ELABORATION) compose the statement to be attributed. Such errors occur because existing systems do not encode long range dependencies between DT constituents such as those between e_3 and e_{4-6}.

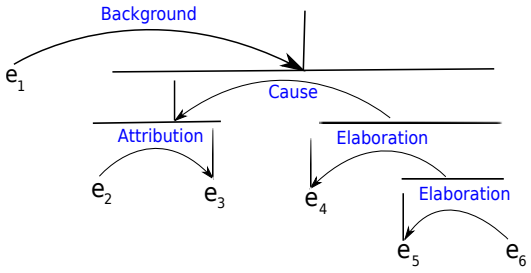
Reranking models can make the global structural information available to the system as follows: first, a base parser produces several DT hypotheses; and then a classifier exploits the entire information in each hypothesis, e.g., the complete DT with its dependencies, for selecting the best DT. Designing features capturing such global properties is however not trivial as it requires the selection of important DT fragments. This means selecting subtree patterns from an exponential feature space. An alternative approach is to implicitly generate the whole feature space using tree kernels (TKs) (Collins and Duffy, 2002; Moschitti, 2006).

In this paper, we present reranking models for discourse parsing based on Support Vector Machines (SVMs) and TKs. The latter allows us to represent structured data using the substructure space thus capturing structural dependencies between DT constituents, which is essential for effective discourse parsing. Specifically, we made the following contributions. First, we extend the

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2049–2060,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics



(a) A human-annotated discourse tree.



(b) A discourse tree generated by Joty et al. (2013).

Figure 1: Example of human-annotated and system-generated discourse trees for the text *[what's more,]$_{e_1}$ [he believes]$_{e_2}$ [seasonal swings in the auto industry this year aren't occurring at the same time in the past,]$_{e_3}$ [because of production and pricing differences]$_{e_4}$ [that are curbing the accuracy of seasonal adjustments]$_{e_5}$] [built into the employment data.]$_{e_6}$* Horizontal lines indicate text segments; satellites are connected to their nuclei by curved arrows.

existing discourse parser[1] (Joty et al., 2013) to produce a list of k most probable parses for each input text, with associated probabilities that define the initial ranking.

Second, we define a set of discourse tree kernels (DISCTK) based on the functional composition of standard TKs with structures representing the properties of DTs. DISCTK can be used for any classification task involving discourse trees.

Third, we use DISCTK to define kernels for reranking and use them in SVMs. Our rerankers can exploit the complete DT structure using TKs. They can ascertain if portions of a DT are compatible, incompatible or simply not likely to coexist, since each substructure is an exploitable feature. In other words, problematic DTs are expected to be ranked lower by our reranker.

Finally, we investigate the potential of our approach by computing the oracle f-scores for both document- and sentence-level discourse parsing. However, as demonstrated later in Section 6, for *document-level* parsing, the top k parses often miss the best parse. For example, the oracle f-scores for 5- and 20-best document-level parsing are only 56.91% and 57.65%, respectively. Thus the scope of improvement for the reranker is rather narrow at the document level. On the other hand, the oracle f-score for 5-best *sentence-level* discourse parsing is 88.09%, where the base parser (i.e., 1-best) has an oracle f-score of 79.77%. Therefore, in this paper we address the following two questions: (*i*) how far can a reranker improve the parsing accuracy at the sentence level? and (*ii*) how far can this improvement, if at all, push the (combined) document-level parsing accuracy?

To this end, our comparative experiments on RST-DT show that the sentence-level reranker can improve the f-score of the state-of-the-art from 79.77% to 82.15%, corresponding to a relative error reduction of 11.8%, which in turn pushes the state-of-the-art document-level f-score from 55.8% to 57.3%, an error reduction of 3.4%.

In the rest of the paper, after introducing the TK technology in Section 2, we illustrate our novel structures, and how they lead to the design of novel DISCTKs in Section 3. We present the k-best discourse parser in Section 4. In Section 5, we describe our reranking approach using DISCTKs. We report our experiments in Section 6. We briefly overview the related work in Section 7, and finally, we summarize our contributions in Section 8.

2 Kernels for Structural Representation

Tree kernels (Collins and Duffy, 2002; Shawe-Taylor and Cristianini, 2004; Moschitti, 2006) are a viable alternative for representing arbitrary subtree structures in learning algorithms. Their basic idea is that kernel-based learning algorithms, e.g., SVMs or perceptron, only need the scalar product between the feature vectors representing the data instances to learn and classify; and kernel functions compute such scalar products in an efficient way. In the following subsections, we briefly describe the kernel machines and three types of tree kernels (TKs), which efficiently compute the scalar product in the subtree space, where the vector components are all possible substructures of the corresponding trees.

2.1 Kernel Machines

Kernel Machines (Cortes and Vapnik, 1995), e.g., SVMs, perform binary classification by learning a hyperplane $H(\vec{x}) = \vec{w} \cdot \vec{x} + b = 0$, where

[1] Available from http://alt.qcri.org/tools/

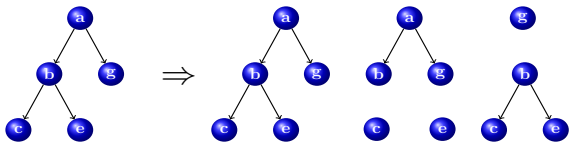


Figure 2: A tree with its STK subtrees; STK$_b$ also includes leaves as features.

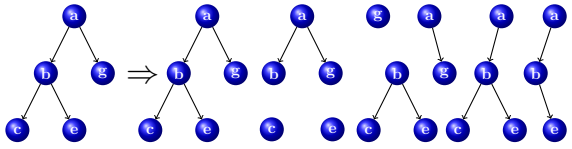


Figure 3: A tree with its PTK fragments.

$\vec{x} \in \mathbb{R}^n$ is the feature vector representation of an object $o \in \mathcal{O}$ to be classified and $\vec{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ are parameters learned from the training data. One can train such machines in the dual space by rewriting the model parameter $\vec{w}$ as a linear combination of training examples, i.e., $\vec{w} = \sum_{i=1..l} y_i \alpha_i \vec{x}_i$, where y_i is equal to 1 for positive examples and -1 for negative examples, $\alpha_i \in \mathbb{R}_+$ and $\vec{x}_i \forall i \in \{1, .., l\}$ are the training instances. Then, we can use the data object $o_i \in \mathcal{O}$ directly in the hyperplane equation considering their mapping function $\phi : \mathcal{O} \rightarrow \mathbb{R}^n$, as follows: $H(o) = \sum_{i=1..l} y_i \alpha_i \vec{x}_i \cdot \vec{x} + b = \sum_{i=1..l} y_i \alpha_i \phi(o_i) \cdot \phi(o) + b = \sum_{i=1..l} y_i \alpha_i K(o_i, o) + b$, where the product $K(o_i, o) = \langle \phi(o_i) \cdot \phi(o) \rangle$ is the kernel function (e.g., TK) associated with the mapping ϕ.

2.2 Tree Kernels

Convolution TKs compute the number of common tree fragments between two trees T_1 and T_2 without explicitly considering the whole fragment space. A TK function over T_1 and T_2 is defined as: $TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$, where N_{T_1} and N_{T_2} are the sets of the nodes of T_1 and T_2, respectively, and $\Delta(n_1, n_2)$ is equal to the number of common fragments rooted in the n_1 and n_2 nodes.[2] The computation of Δ function depends on the shape of fragments, conversely, a different Δ determines the richness of the kernel space and thus different tree kernels. In the following, we briefly describe two existing and well-known tree kernels. Please see several tutorials on kernels (Moschitti, 2013; Moschitti, 2012; Moschitti, 2010) for more details.[3]

Syntactic Tree Kernels (STK) produce fragments such that each of their nodes includes all or none of its children. Figure 2 shows a tree T and its three fragments (do not consider the single nodes) in the STK space on the left and right of the arrow, respectively. STK(T,T) counts the number of common fragments, which in this case is the number of subtrees of T, i.e., three. In the figure, we also show three single nodes, c, e, and g, i.e., the leaves of T, which are computed by a variant of the kernel, that we call STK$_b$. The computational complexity of STK is $O(|N_{T_1}||N_{T_2}|)$, but the average running time tends to be linear (i.e. $O(|N_{T_1}| + |N_{T_2}|)$) for syntactic trees (Moschitti, 2006).

Partial Tree Kernel (PTK) generates a richer set of tree fragments. Given a target tree T, PTK can generate any subset of connected nodes of T, whose edges are in T. For example, Figure 3 shows a tree with its nine fragments including all single nodes (i.e., the leaves of T). PTK is more general than STK as its fragments can include any subsequence of children of a target node. The time complexity of PTK is $O(p\rho^2|N_{T_1}||N_{T_2}|)$, where p is the largest subsequence of children that one wants to consider and ρ is the maximal out-degree observed in the two trees. However, the average running time again tends to be linear for syntactic trees (Moschitti, 2006).

3 Discourse Tree Kernels (DISCTK)

Engineering features that can capture the dependencies between DT constituents is a difficult task. In principle, any dependency between words, relations and structures (see Figure 1) can be an important feature for discourse parsing. This may lead to an exponential number of features, which makes the feature engineering process very hard.

The standard TKs described in the previous section serve as a viable option to get useful subtree features automatically. However, the definition of the input to a TK, i.e., the tree representing a training instance, is extremely important as it implicitly affects the subtree space generated by the TK, where the target learning task is carried out. This can be shown as follows. Let $\phi_M()$ be a mapping from linguistic objects o_i, e.g., a discourse parse, to a *meaningful* tree T_i, and let $\phi_{TK}()$ be a mapping into a tree kernel space us-

[2] To get a similarity score between 0 and 1, it is common to apply a normalization in the kernel space, i.e. $\frac{TK(T_1,T_2)}{\sqrt{TK(T_1,T_1) \times TK(T_2,T_2)}}$.

[3] Tutorials notes available at `http://disi.unitn.it/moschitti/`

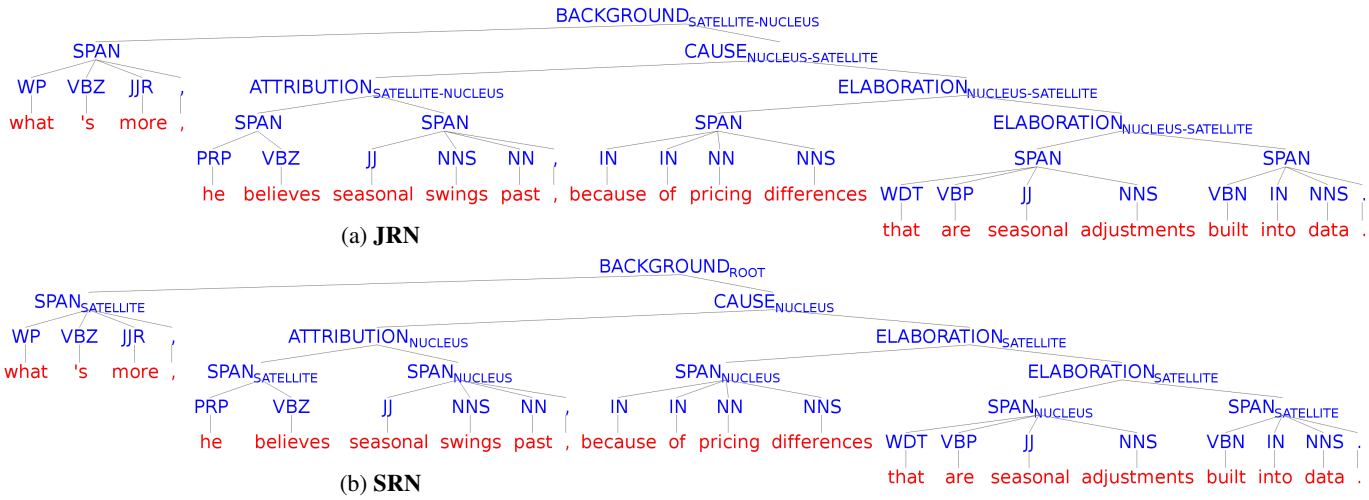


Figure 4: DISCTK trees: (a) Joint Relation-Nucleus (JRN), and (b) Split Relation Nucleus (SRN).

ing one of the TKs described in Section 2.2, i.e., $TK(T_1, T_2) = \phi_{TK}(T_1) \cdot \phi_{TK}(T_2)$. If we apply TK to the objects o_i transformed by $\phi_M()$, we obtain $TK(\phi_M(o_1), \phi_M(o_2)) = \phi_{TK}(\phi_M(o_1)) \cdot \phi_{TK}(\phi_M(o_2)) = \big(\phi_{TK} \circ \phi_M\big)(o_1) \cdot \big(\phi_{TK} \circ \phi_M\big)(o_2) = DiscTK(o_1, o_2)$, which is a new kernel[4] induced by the mapping $\phi_{DiscTK} = \big(\phi_{TK} \circ \phi_M\big)$.

We define two different mappings ϕ_M to transform the discourse parses generated by the base parser into two different tree structures: (*i*) the Joint Relation-Nucleus tree (JRN), and (*ii*) the Split Relation Nucleus tree (SRN).

3.1 Joint Relation-Nucleus Tree (JRN)

As shown in Figure 4a, JRN is a direct mapping of the parser output, where the nuclearity statuses (i.e., satellite or nucleus) of the connecting nodes are attached to the relation labels.[5] For example, the root BACKGROUND$_{Satellite-Nucleus}$ in Figure 4a denotes a *Background* relation between a *satellite* discourse unit on the left and a *nucleus* unit on the right. Text spans (i.e., EDUs) are represented as sequences of Part-of-Speech (POS) tags connected to the associated words, and are grouped under dummy SPAN nodes. We experiment with two lexical variations of the trees: (*i*) *All* includes all the words in the EDU, and (*ii*) *Bigram* includes only the first and last two words in the EDU.

When JRN is used with the STK kernel, an exponential number of fragments are generated. For example, the upper row of Figure 5 shows two smallest (atomic) fragments and one subtree composed of two atomic fragments. Note that much larger structures encoding long range dependencies are also part of the feature space. These fragments can reveal if the discourse units are organized in a compatible way, and help the reranker to detect the kind of errors shown earlier in Figure 1b. However, one problem with JRN representation is that since the relation nodes are composed of three different labels, the generated subtrees tend to be sparse. In the following, we describe SRN that attempts to solve this issue.

3.2 Split Relation Nucleus Tree (SRN)

SRN is not very different from JRN as shown in Figure 4b. The only difference is that instead of attaching the nuclearity statuses to the relation labels, in this representation we assign them to their respective discourse units. When STK kernel is applied to SRN it again produces an exponential number of fragments. For example, the lower row of Figure 5 shows two atomic fragments and one subtree composed of two atomic fragments. Comparing the two examples in Figure 5, it is easy to understand that the space of subtrees extracted from SRN is less sparse than that of JRN.

Note that, as described in Secion 2.2, when the PTK kernel is applied to JRN and SRN trees, it can generate a richer feature space, e.g., features that are paths containing relation labels (e.g., BACKGROUND - CAUSE - ELABORATION or ATTRIBUTION - CAUSE - ELABORATION).

4 Generation of k-best Discourse Parses

In this section we describe the 1-best discourse parser of Joty et al. (2013), and how we extend

[4] People interested in algorithms may like it more designing a complex algorithm to compute $\big(\phi_{TK} \circ \phi_M\big)$. However, the design of ϕ_M is conceptually equivalent and more effective from an engineering viewpoint.

[5] This is a common standard followed by the parsers.

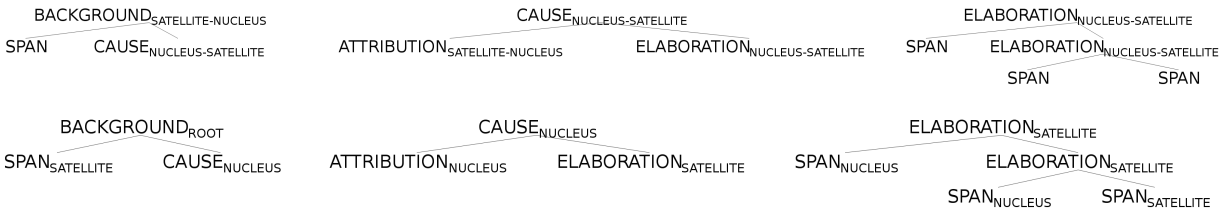


Figure 5: Fragments from JRN in Figure 4a (upper row) and SRN in Figure 4b (lower row).

it to k-best discourse parsing.

Joty et al. (2013) decompose the problem of document-level discourse parsing into two stages as shown in Figure 6. In the first stage, the *intra-sentential* discourse parser produces discourse subtrees for the individual sentences in a document. Then the *multi-sentential* parser combines the sentence-level subtrees and produces a DT for the document. Both parsers have the same two components: a *parsing model* and a *parsing algorithm*. The parsing model explores the search space of possible DTs and assigns a probability to every possible DT. Then the parsing algorithm selects the most probable DT(s). While two separate parsing models are employed for intra- and multi-sentential parsing, the same parsing algorithm is used in both parsing conditions. The two-stage parsing exploits the fact that sentence boundaries correlate very well with discourse boundaries. For example, more than 95% of the sentences in RST-DT have a well-formed discourse subtree in the full document-level discourse tree.

The choice of using two separate models for intra- and multi-sentential parsing is well justified for the following two reasons: (*i*) it has been observed that discourse relations have different distributions in the two parsing scenarios, and (*ii*) the models could independently pick their own informative feature sets. The parsing model used for intra-sentential parsing is a Dynamic Conditional Random Field (DCRF) (Sutton et al., 2007) shown in Figure 7. The observed nodes U_j at the bottom layer represent the discourse units at a certain level of the DT; the binary nodes S_j at the middle layer predict whether two adjacent units U_{j-1} and U_j should be connected or not; and the multi-class nodes R_j at the top layer predict the discourse relation between U_{j-1} and U_j. Notice that the model represents the structure and the label of a DT constituent jointly, and captures the sequential dependencies between the DT constituents. Since the chain-structured DCRF model does not scale up to multi-sentential parsing of long documents,

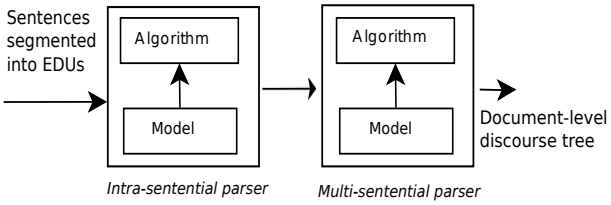


Figure 6: The two-stage document-level discourse parser proposed by Joty et al. (2013).

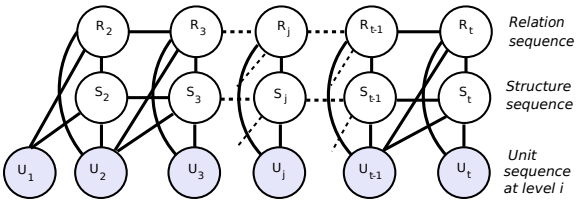


Figure 7: The intra-sentential parsing model.

the multi-sentential parsing model is a CRF which breaks the chain structure of the DCRF model.

The parsing models are applied recursively at different levels of the DT in their respective parsing scenarios (i.e., intra- and multi-sentential), and the probabilities of all possible DT constituents are obtained by computing the posterior marginals over the relation-structure pairs (i.e., $P(R_j, S_j{=}1|U_1, \cdots, U_t, \Theta)$, where Θ are model parameters). These probabilities are then used in a CKY-like probabilistic parsing algorithm to find the globally optimal DT for the given text.

Let U_x^b and U_x^e denote the beginning and end EDU Ids of a discourse unit U_x, and $R[U_i^b, U_m^e, U_j^e]$ refers to a coherence relation R that holds between the discourse unit containing EDUs U_i^b through U_m^e and the unit containing EDUs $U_m^e{+}1$ through U_j^e. Given n discourse units, the parsing algorithm uses the upper-triangular portion of the $n{\times}n$ dynamic programming table A, where cell $A[i,j]$ (for $i < j$) stores:

$$A[i,j] = P(r^*[U_i^b, U_{m^*}^e, U_j^e]), \text{ where}$$

$$(m^*, r^*) = \underset{i \le m < j \ ; \ R}{\operatorname{argmax}} P(R[U_i^b, U_m^e, U_j^e]) \times A[i,m] \times A[m+1,j] \quad (1)$$

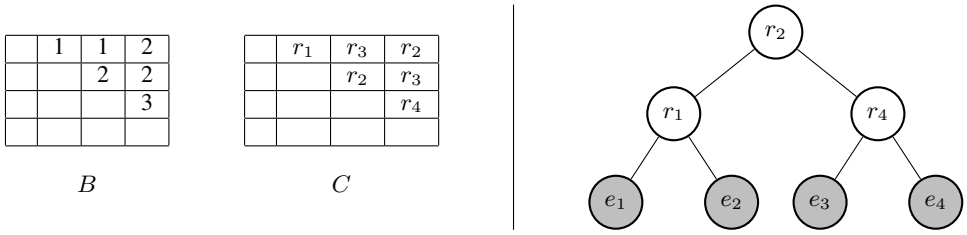


Figure 8: The B and C dynamic programming tables (left), and the corresponding discourse tree (right).

In addition to A, which stores the *probability* of the most probable constituents of a DT, the parsing algorithm also simultaneously maintains two other tables B and C for storing the best structure (i.e., $U^e_{m^*}$) and the relations (i.e., r^*) of the corresponding DT constituents, respectively. For example, given 4 EDUs $e_1 \cdots e_4$, the B and C tables at the left side in Figure 8 together represent the DT shown at the right. More specifically, to generate the DT, we first look at the top-right entries in the two tables, and find $B[1,4] = 2$ and $C[1,4] = r_2$, which specify that the two discourse units $e_{1:2}$ and $e_{3:4}$ should be connected by the relation r_2 (the root in the DT). Then, we see how EDUs e_1 and e_2 should be connected by looking at the entries $B[1,2]$ and $C[1,2]$, and find $B[1,2] = 1$ and $C[1,2] = r_1$, which indicates that these two units should be connected by the relation r_1 (the left pre-terminal). Finally, to see how EDUs e_3 and e_4 should be linked, we look at the entries $B[3,4]$ and $C[3,4]$, which tell us that they should be linked by the relation r_4 (the right pre-terminal).

It is straight-forward to generalize the above algorithm to produce k most probable DTs. When filling up the dynamic programming tables, rather than storing a single best parse for each subtree, we store and keep track of k-best candidates simultaneously. More specifically, each cell in the dynamic programming tables (i.e., A, B and C) should now contain k entries (sorted by their probabilities), and for each such entry there should be a back-pointer that keeps track of the decoding path.

The algorithm works in polynomial time. For n discourse units and M number of relations, the 1-best parsing algorithm has a time complexity of $O(n^3M)$ and a space complexity of $O(n^2)$, where the k-best version has a time and space complexities of $O(n^3Mk^2 \log k)$ and $O(n^2k)$, respectively. There are cleverer ways to reduce the complexity (e.g., see (Huang and Chiang, 2005) for three such ways). However, since the efficiency of the algorithm did not limit us to produce k-best parses for larger k, it was not a priority in this work.

5 Kernels for Reranking Discourse Trees

In Section 3, we described DISCTK, which essentially can be used for any classification task involving discourse trees. For example, given a DT, we can use DISCTK to classify it as correct vs. incorrect. However, such classification is not completely aligned to our purpose, since our goal is to select the best (i.e., the most correct) DT from k candidate DTs; i.e., a ranking task. We adopt a preference reranking technique as described in (Moschitti et al., 2006; Dinarelli et al., 2011).

5.1 Preference Reranker

Preference reranking (PR) uses a classifier $\mathcal{C}$ of pairs of hypotheses $\langle h_i, h_j \rangle$, which decides if h_i (i.e., a candidate DT in our case) is better than h_j. We generate positive and negative examples to train the classifier using the following approach. The pairs $\langle h_1, h_i \rangle$ constitute positive examples, where h_1 has the highest f-score accuracy on the Relation metric (to be described in Section 6) with respect to the gold standard among the candidate hypotheses, and vice versa, $\langle h_i, h_1 \rangle$ are considered as negative examples. At test time, $\mathcal{C}$ classifies all pairs $\langle h_i, h_j \rangle$ generated from the k-best hypotheses. A positive decision is a vote for h_i, and a negative decision is a vote for h_j. Also, the classifier score can be used as a weighted vote. Hypotheses are then ranked according to the number (sum) of the (weighted) votes they get.[6]

We build our reranker using simple SVMs.[7]

[6] As shown by Collins and Duffy (2002), only the classification of k hypotheses (paired with the empty one) is needed in practice, thus the complexity is only $O(k)$.

[7] Structural kernels, e.g., TKs, cannot be used in more advanced algorithms working in structured output spaces, e.g., SVM^{struct}. Indeed, to our knowledge, no one could successfully find a general and exact solution for the $argmax$ equation, typically part of such advanced models, when structural kernels are used. Some approximate solutions for simple kernels, e.g., polynomial or gaussian kernels, are given in (Joachims and Yu, 2009), whereas (Severyn and Moschitti, 2011; Severyn and Moschitti, 2012) provide solutions for using the cutting-plane algorithm (which requires $argmax$ computation) with structural kernels but in binary SVMs.

Since in our problem a pair of hypotheses $\langle h_i, h_j \rangle$ constitutes a data instance, we now need to define the kernel between the pairs. However, notice that DISCTK only works on a single pair.

Considering that our task is to decide whether h_i is better than h_j, it can be convenient to represent the pairs in terms of differences between the vectors of the two hypotheses, i.e., $\phi_K(h_i) - \phi_K(h_j)$, where K (i.e., DISCTK) is defined between two hypotheses (not on two pairs of hypotheses). More specifically, to compute this difference implicitly, we can use the following kernel summation: $PK(\langle h_1, h_2 \rangle, \langle h'_1, h'_2 \rangle) = (\phi_K(h_1) - \phi_K(h_2)) \circ (\phi_K(h'_1) - \phi_K(h'_2)) = K(h_1, h'_1) + K(h_2, h'_2) - K(h_1, h'_2) - K(h_2, h'_1)$.

In general, Preference Kernel (PK) works well because it removes many identical features by taking differences between two huge implicit TK-vectors. In our reranking framework, we also include traditional feature vectors in addition to the trees. Therefore, each hypothesis h is represented as a tuple $\langle T, \vec{v} \rangle$ composed of a tree T and a feature vector $\vec{v}$. We then define a structural kernel (i.e., similarity) between two hypotheses h and h' as follows: $K(h, h') = DiscTK(T, T') + FV(\vec{v}, \vec{v}')$, where DISCTK maps the DTs T and T' to JRN or SRN and then applies STK, STK$_b$ or PTK defined in Sections 2.2 and 3, and FV is a standard kernel, e.g., linear, polynomial, gaussian, etc., over feature vectors (see next section).

5.2 Feature Vectors

We also investigate the impact of traditional (i.e., not subtree) features for reranking discourse parses. Our feature vector comprises two types of features that capture global properties of the DTs.

Basic Features. This set includes eight global features. The first two are the probability and the (inverse) rank of the DT given by the base parser. These two features are expected to help the reranker to perform at least as good as the base parser. The other six features encode the structural properties of the DT, which include depth of the DT, number of nodes connecting two EDUs (i.e., SPANs in Figure 4), number of nodes connecting two relational nodes, number of nodes connecting a relational node and an EDU, number of nodes that connects a relational node as left child and an EDU as right child, and vice versa.

Relation Features. We encode the relations in the DT as bag-of-relations (i.e., frequency count). This will allow us to assess the impact of a flat representation of the DT. Note that more important relational features would be the subtree patterns extracted from the DT. However, they are already generated by TKs in a simpler way. See (Pighin and Moschitti, 2009; Pighin and Moschitti, 2010) for a way to extract the most relevant features from a model learned in the kernel space.

6 Experiments

Our experiments aim to show that reranking of discourse parses is a promising research direction, which can improve the state-of-the-art. To achieve this, we (*i*) compute the oracle accuracy of the k-best parser, (*ii*) test different kernels for reranking discourse parses by applying standard kernels to our new structures, (*iii*) show the reranking performance using the best kernel for different number of hypotheses, and (*iv*) show the relative importance of features coming from different sources.

6.1 Experimental Setup

Data. We use the standard RST-DT corpus (Carlson et al., 2002), which comes with discourse annotations for 385 articles (347 for training and 38 for testing) from the Wall Street Journal. We extracted sentence-level DTs from a document-level DT by finding the subtrees that exactly span over the sentences. This gives 7321 and 951 sentences in the training and test sets, respectively. Following previous work, we use the same 18 coarser relations defined by Carlson and Marcu (2001).

We create the training data for the reranker in a 5-fold cross-validation fashion.[8] Specifically, we split the training set into 5 equal-sized folds, and train the parsing model on 4 folds and apply to the rest to produce k most probable DTs for each text. Then we generate and label the pairs (by comparing with the gold) from the k most probable trees as described in Section 5.1. Finally, we merge the 5 labeled folds to create the full training data.

SVM Reranker. We use SVM-light-TK to train our reranking models,[9] which enables the use of tree kernels (Moschitti, 2006) in SVM-light (Joachims, 1999). We build our new kernels for reranking exploiting the standard built-in TK functions, such as STK, STK$_b$ and PTK. We applied

[8] Note that our earlier experiments with a 2-fold cross validation process yielded only 50% of our current improvement.

[9] http://disi.unitn.it/moschitti/Tree-Kernel.htm

a linear kernel to standard feature vectors as it showed to be the best on our development set.

Metrics. The standard procedure to evaluate discourse parsing performance is to compute Precision, Recall and f-score of the unlabeled and labeled metrics proposed by Marcu (2000b).[10] Specifically, the unlabeled metric *Span* measures how accurate the parser is in finding the right structure (i.e., skeleton) of the DT, while the labeled metrics *Nuclearity* and *Relation* measure the parser's ability to find the right labels (nuclearity and relation) in addition to the right structure. Optimization of the Relation metric is considered to be the hardest and the most desirable goal in discourse parsing since it gives aggregated evaluation on tree structure and relation labels. Therefore, we measure the *oracle* accuracy of the k-best discourse parser based on the f-scores of the *Relation* metric, and our reranking framework aims to optimize the Relation metric.[11] Specifically, the oracle accuracy for k-best parsing is measured as follows: $\text{ORACLE} = \frac{\sum_{i=1}^{N} \max_{j=1}^{k} \text{f-score}_r(g_i, h_i^j)}{N}$, where N is the total number of texts (sentences or documents) evaluated, g_i is the gold DT annotation for text i, h_i^j is the j^{th} parse hypothesis generated by the k-best parser for text i, and $f\text{-}score_r(g_i, h_i^j)$ is the f-score accuracy of hypothesis h_i^j on the Relation metric. In all our experiments we report the f-scores of the Relation metric.

6.2 Oracle Accuracy

Table 1 presents the oracle scores of the k-best intra-sentential parser PAR-S on the standard RST-DT test set. The 1-best result corresponds to the accuracy of the base parser (i.e., 79.77%). The 2-best shows dramatic oracle-rate improvement (i.e., 4.65% absolute), suggesting that the base parser often generates the best tree in its top 2 outputs. 5-best increases the oracle score to 88.09%. Afterwards, the increase in accuracy slows down, achieving, e.g., 90.37% and 92.57% at 10-best and 20-best, respectively.

The results are quite different at the document level as Table 2 shows the oracle scores of the k-best document-level parser PAR-D.[12] The results suggest that the best tree is often missing in the top k parses, and the improvement in oracle-rate is very little as compared to the sentence-level parsing. The 2-best and the 5-best improve over the base accuracy by only 0.7% and 1.0%, respectively. The improvement becomes even lower for larger k. For example, the gain from 20-best to 30-best parsing is only 0.09%. This is not surprising because generally document-level DTs are big with many constituents, and only a very few of them change from k-best to k+1-best parsing. These small changes do not contribute much to the overall f-score accuracy.[13] In summary, the results in Tables 1 and 2 demonstrate that a k-best reranker can potentially improve the parsing accuracy at the sentence level, but may not be a suitable option for improving parsing at the document level. In the following, we report our results for reranking sentence-level discourse parses.

k	1	2	5	10	15	20
PAR-S	79.77	84.42	88.09	90.37	91.74	92.57

Table 1: Oracle scores as a function of k of k-best sentence-level parses on RST-DT test set.

k	1	2	5	10	15	20
PAR-D	55.83	56.52	56.91	57.23	57.54	57.65

Table 2: Oracle scores as a function of k of k-best document-level parses on RST-DT test set.

6.3 Performance of Different DISCTKs

Section 3 has pointed out that different DISCTKs can be obtained by specifying the TK type (e.g., STK, STK$_b$, PTK) and the mapping ϕ_M (i.e., JRN, SRN) in the overall kernel function $\big(\phi_{TK} \circ \phi_M\big)(o_1) \cdot \big(\phi_{TK} \circ \phi_M\big)(o_2)$. Table 3 reports the performance of such model compositions using the 5-best hypotheses on the RST-DT test set. Additionally, it also reports the accuracy for the two versions of JRN and SRN, i.e., Bigram and All. From these results, we can note the following.

Firstly, the kernels generally perform better on Bigram than All lexicalization. This suggests that using all the words from the text spans (i.e., EDUs) produces sparse models.

[10] Precision, Recall and f-score are the same when the discourse parser uses manual discourse segmentation. Since all our experiments in this paper are based on manual discourse segmentation, we only report the f-scores.

[11] It is important to note that optimizing Relation metric may also result in improved Nuclearity scores.

[12] For document-level parsing, Joty et al. (2013) propose two approaches to combine intra- and multi-sentential parsers, namely 1S-1S (1 Sentence-1 Subtree) and Sliding window. In this work we extend 1S-1S to k-best document-level parser PAR-D since it is not only time efficient but it also achieves better results on the Relation metric.

[13] Note that Joty et al. (2012; 2013) report lower f-scores both at the sentence level (i.e., 77.1% as opposed to our 79.77%) and at the document level (i.e., 55.73% as opposed to our 55.83%). We fixed a crucial bug in their (1-best) parsing algorithm, which accounts for the improved performance.

$\phi_{TK} \circ \phi_M$	JRN		SRN	
	Bigram	All	Bigram	All
STK	81.28	80.04	82.15	80.04
STK_b	81.35	80.28	82.18	80.25
PTK	81.63	78.50	81.42	78.25

Table 3: Reranking performance of different discourse tree kernels on different representations.

Secondly, while the tree kernels perform similarly on the JRN representation, STK performs significantly better (p-value < 0.01) than PTK on SRN.[14] This result is interesting as it provides indications of the type of DT fragments useful for improving parsing accuracy. As pointed out in Section 2.2, PTK includes all features generated by STK, and additionally, it includes fragments whose nodes can have any subsets of the children they have in the original DT. Since this does not improve the accuracy, we speculate that complete fragments, e.g., [CAUSE [ATTRIBUTION][ELABORATION]] are more meaningful than the partial ones, e.g., [CAUSE [ATTRIBUTION]] and [CAUSE [ELABORATION]], which may add too much uncertainty on the signature of the relations contained in the DT. We verified this hypothesis by running an experiment with PTK constraining it to only generate fragments whose nodes preserve all or none of their children. The accuracy of such fragments approached the ones of STK, suggesting that relation information should be used as a whole for engineering features.

Finally, STK_b is slightly (but not significantly) better than STK suggesting that the lexical information is already captured by the base parser.

Note that the results in Table 3 confirms many other experiments we carried out on several development sets. For any run: (*i*) STK always performs as well as STK_b, (*ii*) STK is always better than PTK, and (*iii*) SRN is always better than JRN. In what follows, we show the reranking performance based on STK applied to SRN with Bigram.

6.4 Insights on DISCTK-based Reranking

Table 4 reports the performance of our reranker (RR) in comparison with the oracle (OR) accuracy for different values of k, where we also show the corresponding relative error rate reduction (ERR) with respect to the baseline. To assess the generality of our approach, we evaluated our reranker on both the standard test set and the entire training set using 5-fold cross validation.[15]

[14] Statistical significance is verified using paired t-test.

[15] The reranker was trained on 4 folds and tested on the rest

Baseline	Basic feat.	+ Rel. feat.	+ Tree
79.77	79.84	79.81	82.15

Table 5: Comparison of features from different sources for 5-best discourse reranking.

	(Joty et al., 2013)	With Reranker
PAR-D	55.8	**57.3**

Table 6: Document-level parsing results with 5-best sentence-level discourse reranker.

We note that: (*i*) the best result on the standard test set is 82.15% for $k = 4$ and 5, which gives an ERR of 11.76%, and significantly (p-value < 0.01) outperforms the baseline, (*ii*) the improvement is consistent when we move from standard test set to 5-folds, (*iii*) the best result on the 5-folds is 80.86 for $k = 6$, which is significantly (p-value < 0.01) better than the baseline 78.57, and gives an ERR of 11.32%. We also experimented with other values of k in both training and test sets (also increasing k only in the test set), but we could not improve over our best result. This suggests that outperforming the baseline (which in our case is the state of the art) is rather difficult.[16]

In this respect, we also investigated the impact of traditional ranking methods based on feature vectors, and compared it with our TK-based model. Table 5 shows the 5-best reranking accuracy for different feature subsets. The *Basic features* (Section 5.2) alone do not significantly improve over the *Baseline*. The only relevant features are the probability and the rank of each hypothesis, which condense all the information of the local model (TKs models always used them).

Similarly, adding the relations as bag-of-relations in the vector (*Rel. feat.*) does not provide any gain, whereas the relations encoded in the tree fragments (Tree) gives improvement. This shows the importance of using structural dependencies for reranking discourse parses.

Finally, Table 6 shows that if we use our sentence-level reranker in the document-level parser of Joty et al. (2013), the accuracy of the latter increases from 55.8% to 57.3%, which is a significant improvement ($p < 0.01$), and establishes a new state-of-the-art for document-level parsing.

6.5 Error Analysis

We looked at some examples where our reranker failed to identify the best DT. Unsurprisingly, it

[16] The human agreement on sentence-level parsing is 83%.

	Standard test set						5-folds (average)					
	k=1	k=2	k=3	k=4	k=5	k=6	k=1	k=2	k=3	k=4	k=5	k=6
RR	79.77	81.08	81.56	**82.15**	**82.15**	82.11	78.57	79.76	80.28	80.68	80.80	**80.86**
ERR	-	6.48	8.85	**11.76**	**11.76**	11.57	-	5.88	8.45	10.43	11.02	**11.32**
OR	79.77	84.42	86.55	87.68	88.09	88.75	78.57	83.20	85.13	86.49	87.35	88.03

Table 4: Reranking performance (RR) in comparison with oracle (OR) accuracy for different values of k on the standard testset and 5-folds of RST-DT. Second row shows the relative error rate reduction (ERR).

happens many times for small DTs containing only two or three EDUs, especially when the relations are semantically similar. Figure 9 presents such a case, where the reranker fails to rank the DT with *Summary* ahead of the DT with *Elaboration*. Although we understand that the reranker lacks enough structural context to distinguish the two relations in this example, we expected that including the lexical items (e.g., *(CFD)*) in our DT representation could help. However, similar short parenthesized texts are also used to elaborate as in *Senate Majority Leader George Mitchell (D., Maine)*, where the text *(D., Maine)* (i.e., Democrat from state Maine) elaborates its preceding text. This confuses our reranker. We also found error examples where the reranker failed to distinguish between *Background* and *Elaboration*, and between *Cause* and *Elaboration*. This suggests that we need rich semantic representation of the text to improve our reranker further.

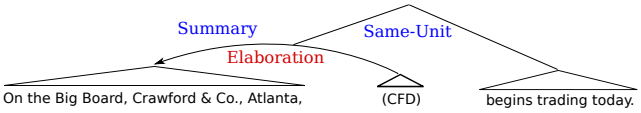


Figure 9: An error made by our reranker.

7 Related Work

Early work on discourse parsing applied hand-coded rules based on discourse cues and surface patterns (Marcu, 2000a). Supervised learning was first attempted by Marcu (2000b) to build a shift-reduce discourse parser. This work was then considerably improved by Soricut and Marcu (2003). They presented probabilistic generative models for *sentence-level* discourse parsing based on lexico-syntactic patterns. Sporleder and Lapata (2005) investigated the necessity of syntax in discourse analysis. More recently, Hernault et al. (2010) presented the HILDA discourse parser that iteratively employs two SVM classifiers in pipeline to build a DT in a greedy way. Feng and Hirst (2012) improved the HILDA parser by incorporating rich linguistic features, which include lexical semantics and discourse production rules.

Joty et al. (2013) achieved the best prior results by (*i*) jointly modeling the structure and the label of a DT constituent, (*ii*) performing optimal rather than greedy decoding, and (*iii*) discriminating between intra- and multi-sentential discourse parsing. However, their model does not consider long range dependencies between DT constituents, which are encoded by our kernels. Regarding the latter, our work is surely inspired by (Collins and Duffy, 2002), which uses TK for syntactic parsing reranking or in general discriminative reranking, e.g., (Collins and Koo, 2005; Charniak and Johnson, 2005; Dinarelli et al., 2011). However, such excellent studies do not regard discourse parsing, and in absolute they achieved lower improvements than our methods.

8 Conclusions and Future Work

In this paper, we have presented a discriminative approach for reranking discourse trees generated by an existing discourse parser. Our reranker uses tree kernels in SVM preference ranking framework to effectively capture the long range structural dependencies between the constituents of a discourse tree. We have shown the reranking performance for sentence-level discourse parsing using the standard tree kernels (i.e., STK and PTK) on two different representations (i.e., JRN and SRN) of the discourse tree, and compare it with the traditional feature vector-based approach. Our results show that: (*i*) the reranker improves only when it considers subtree features computed by the tree kernels, (*ii*) SRN is a better representation than JRN, (*iii*) STK performs better than PTK for reranking discourse trees, and (*iv*) our best result outperforms the state-of-the-art significantly.

In the future, we would like to apply our reranker to the document-level parses. However, this will require a better hypotheses generator.

Acknowledgments

This research is part of the Interactive sYstems for Answer Search (Iyas) project, conducted by the Arabic Language Technologies (ALT) group at Qatar Computing Research Institute (QCRI) within the Qatar Foundation.

References

Lynn Carlson and Daniel Marcu. 2001. Discourse Tagging Reference Manual. Technical Report ISI-TR-545, University of Southern California Information Sciences Institute.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2002. RST Discourse Treebank (RST-DT) LDC2002T07. *Linguistic Data Consortium, Philadelphia*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, ACL'05, pages 173–180, NJ, USA. ACL.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *ACL*.

Michael Collins and Terry Koo. 2005. Discriminative Reranking for Natural Language Parsing. *Comput. Linguist.*, 31(1):25–70, March.

Corinna Cortes and Vladimir Vapnik. 1995. Support Vector Networks. *Machine Learning*, 20:273–297.

Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. 2011. Discriminative Reranking for Spoken Language Understanding. *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, 20:526539.

Vanessa Feng and Graeme Hirst. 2012. Text-level Discourse Parsing with Rich Linguistic Features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, ACL '12, pages 60–68, Jeju Island, Korea. ACL.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010. HILDA: A Discourse Parser Using Support Vector Machine Classification. *Dialogue and Discourse*, 1(3):1—33.

Liang Huang and David Chiang. 2005. Better K-best Parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing '05, pages 53–64, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thorsten Joachims and Chun-Nam John Yu. 2009. Sparse Kernel SVMs via Cutting-Plane Training. *Machine Learning*, 76(2-3):179–193. ECML.

Thorsten Joachims. 1999. Making large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vector Learning*.

Shafiq Joty, Giuseppe Carenini, and Raymond T. Ng. 2012. A Novel Discriminative Framework for Sentence-Level Discourse Analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 904–915, Jeju Island, Korea. ACL.

Shafiq Joty, Giuseppe Carenini, Raymond T. Ng, and Yashar Mehdad. 2013. Combining Intra- and Multi-sentential Rhetorical Parsing for Document-level Discourse Analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, Sofia, Bulgaria. ACL.

William Mann and Sandra Thompson. 1988. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(3):243–281.

Daniel Marcu. 2000a. The Rhetorical Parsing of Unrestricted Texts: A Surface-based Approach. *Computational Linguistics*, 26:395–448.

Daniel Marcu. 2000b. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press, Cambridge, MA, USA.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2006. Semantic Role Labeling via Tree Kernel Joint Inference. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 61–68, New York City, June. Association for Computational Linguistics.

Alessandro Moschitti. 2006. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In *17th European Conference on Machine Learning*, pages 318–329. Springer.

Alessandro Moschitti. 2010. Kernel Engineering for Fast and Easy Design of Natural Language Applications. In *COLING (Tutorials)*, pages 1–91.

Alessandro Moschitti. 2012. State-of-the-Art Kernels for Natural Language Processing. In *Tutorial Abstracts of ACL 2012*, page 2, Jeju Island, Korea, July. Association for Computational Linguistics.

Alessandro Moschitti. 2013. Kernel-based Learning to Rank with Syntactic and Semantic Structures. In *SIGIR*, page 1128.

Daniele Pighin and Alessandro Moschitti. 2009. Reverse Engineering of Tree Kernel Feature Spaces. In *EMNLP*, pages 111–120.

Daniele Pighin and Alessandro Moschitti. 2010. On Reverse Feature Engineering of Syntactic Tree Kernels. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 223–233, Uppsala, Sweden, July. Association for Computational Linguistics.

Aliaksei Severyn and Alessandro Moschitti. 2011. Fast Support Vector Machines for Structural Kernels. In *ECML/PKDD (3)*, pages 175–190.

Aliaksei Severyn and Alessandro Moschitti. 2012. Fast Support Vector Machines for Convolution Tree Kernels. *Data Min. Knowl. Discov.*, 25(2):325–357.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Radu Soricut and Daniel Marcu. 2003. Sentence Level Discourse Parsing Using Syntactic and Lexical Information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL'03, pages 149–156, Edmonton, Canada. ACL.

Caroline Sporleder and Mirella Lapata. 2005. Discourse Chunking and its Application to Sentence Compression. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT-EMNLP'05, pages 257–264, Vancouver, British Columbia, Canada. ACL.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labeling and Segmenting Sequence Data. *Journal of Machine Learning Research (JMLR)*, 8:693–723.

Recursive Deep Models for Discourse Parsing

Jiwei Li[1], Rumeng Li[2] and Eduard Hovy[3]
[1]Computer Science Department, Stanford University, Stanford, CA 94305, USA
[2]School of EECS, Peking University, Beijing 100871, P.R. China
[3]Language Technology Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
jiweil@stanford.edu alicerumeng@foxmail.com ehovy@andrew.cmu.edu

Abstract

Text-level discourse parsing remains a challenge: most approaches employ features that fail to capture the intentional, semantic, and syntactic aspects that govern discourse coherence. In this paper, we propose a recursive model for discourse parsing that jointly models distributed representations for clauses, sentences, and entire discourses. The learned representations can to some extent learn the semantic and intentional import of words and larger discourse units automatically,. The proposed framework obtains comparable performance regarding standard discoursing parsing evaluations when compared against current state-of-art systems.

1 Introduction

In a coherent text, units (clauses, sentences, and larger multi-clause groupings) are tightly connected semantically, syntactically, and logically. Mann and Thompson (1988) define a text to be coherent when it is possible to describe clearly the role that each discourse unit (at any level of grouping) plays with respect to the whole. In a coherent text, no unit is completely isolated. Discourse parsing tries to identify how the units are connected with each other and thereby uncover the hierarchical structure of the text, from which multiple NLP tasks can benefit, including text summarization (Louis et al., 2010), sentence compression (Sporleder and Lapata, 2005) or question-answering (Verberne et al., 2007).

Despite recent progress in automatic discourse segmentation and sentence-level parsing (e.g., (Fisher and Roark, 2007; Joty et al., 2012; Soricut and Marcu, 2003), document-level discourse parsing remains a significant challenge. Recent attempts (e.g., (Hernault et al., 2010b; Feng and Hirst, 2012; Joty et al., 2013)) are still considerably inferior when compared to human gold-standard discourse analysis. The challenge stems from the fact that compared with sentence-level dependency parsing, the set of relations between discourse units is less straightforward to define. Because there are no clause-level 'parts of discourse' analogous to word-level parts of speech, there is no discourse-level grammar analogous to sentence-level grammar. To understand how discourse units are connected, one has to understand the communicative function of each unit, and the role it plays within the context that encapsulates it, taken recursively all the way up for the entire text. Manually developed features relating to words and other syntax-related cues, used in most of the recent prevailing approaches (e.g., (Feng and Hirst, 2012; Hernault et al., 2010b)), are insufficient for capturing such nested intentionality.

Recently, deep learning architectures have been applied to various natural language processing tasks (for details see Section 2) and have shown the advantages to capture the relevant semantic and syntactic aspects of units in context. As word distributions are composed to form the meanings of clauses, the goal is to extend distributed clause-level representations to the single- and multi-sentence (discourse) levels, and produce the hierarchical structure of entire texts.

Inspired by this idea, we introduce in this paper a deep learning approach for discourse parsing. The proposed parsing algorithm relies on a recursive neural network to decide (1) whether two discourse units are connected and if so (2) by what relation they are connected. Concretely, the parsing algorithm takes as input a document of any length, and first obtains the distributed representation for each of its sentences using recursive convolution based on the sentence parse tree. It then proceeds bottom-up, applying a binary classifier to determine the probability of two adjacent

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2061–2069,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

discourse units being merged to form a new subtree followed by a multi-class classifier to select the appropriate discourse relation label, and calculates the distributed representation for the subtree so formed, gradually unifying subtrees until a single overall tree spans the entire sentence. The compositional distributed representation enables the parser to make accurate parsing decisions and capture relations between different sentences and units. The binary and multi-class classifiers, along with parameters involved in convolution, are jointly trained from a collection of gold-standard discourse structures.

The rest of this paper is organized as follows. We present related work in Section 2 and describe the RST Discourse Treebank in Section 3. The sentence convolution approach is illustrated in Section 4 and the discourse parser model in Section 5. We report experimental results in Section 6 and conclude in Section 7.

2 Related Work

2.1 Discourse Analysis and Parsing

The basis of discourse structure lies in the recognition that discourse units (minimally, clauses) are related to one another in principled ways, and that the juxtaposition of two units creates a joint meaning larger than either unit's meaning alone. In a coherent text this juxtaposition is never random, but serves the speaker's communicative goals.

Considerable work on linguistic and computational discourse processing in the 1970s and 80s led to the development of several proposals for relations that combine units; for a compilation see (Hovy and Maier, 1997). Of these the most influential is Rhetorical Structure Theory RST (Mann and Thompson, 1988) that defines about 25 relations, each containing semantic constraints on its component parts plus a description of the overall functional/semantic effect produced as a unit when the parts have been appropriately connected in the text. For example, the SOLUTIONHOOD relation connects one unit describing a problem situation with another describing its solution, using phrases such as "the answer is"; in successful communication the reader will understand that a problem is described and its solution is given.

Since there is no syntactic definition of a problem or solution (they can each be stated in a single clause, a paragraph, or an entire text), one has to characterize discourse units by their communicative (rhetorical) function. The functions are reflected in text as signals of the author's intentions, and take various forms (including expressions such as "therefore", "for example", "the answer is", and so on; patterns of tense or pronoun usage; syntactic forms; etc.). The signals govern discourse blocks ranging from a clause to an entire text , each one associated with some discourse relation.

In order to build a text's hierarchical structure, a discourse parser needs to recognize these signals and use them to appropriately compose the relationship and nesting. Early approaches (Marcu, 2000a; LeThanh et al., 2004) rely mainly on overt discourse markers (or cue words) and use hand-coded rules to build text structure trees, bottom-up from clauses to sentences to paragraphs.... Since a hierarchical discourse tree structure is analogous to a constituency based syntactic tree, modern research explored syntactic parsing techniques (e.g., CKY) for discourse parsing based on multiple text-level or sentence-level features (Soricut and Marcu, 2003; Reitter, 2003; Baldridge and Lascarides, 2005; Subba and Di Eugenio, 2009; Lin et al., 2009; Luong et al., 2014).

A recent prevailing idea for discourse parsing is to train two classifiers, namely a binary structure classifier for determining whether two adjacent text units should be merged to form a new subtree, followed by a multi-class relation classifier for determining which discourse relation label should be assigned to the new subtree. The idea is proposed by Hernault and his colleagues (Duverle and Prendinger, 2009; Hernault et al., 2010a) and followed by other work using more sophisticated features (Feng and Hirst, 2012; Hernault et al., 2010b). Current state-of-art performance for relation identification is achieved by the recent representation learning approach proposed by (Ji and Eisenstein, 2014). The proposed framework presented in this paper is similar to (Ji and Eisenstein, 2014) for transforming the discourse units to the abstract representations.

2.2 Recursive Deep Learning

Recursive neural networks constitute one type of deep learning frameworks which was first proposed in (Goller and Kuchler, 1996). The recursive framework relies and operates on structured inputs (e.g., a parse tree) and computes the representation for each parent based on its children

iteratively in a bottom-up fashion. A series of variations of RNN has been proposed to tailor different task-specific requirements, including Matrix-Vector RNN (Socher et al., 2012) that represents every word as both a vector and a matrix, or Recursive Neural Tensor Network (Socher et al., 2013) that allows the model to have greater interactions between the input vectors. Many tasks have benefited from the recursive framework, including parsing (Socher et al., 2011b), sentiment analysis (Socher et al., 2013), textual entailment (Bowman, 2013), segmentation (Wang and Mansur, 2013; Houfeng et al., 2013), and paraphrase detection (Socher et al., 2011a).

3 The RST Discourse Treebank

There are today two primary alternative discourse treebanks suitable for training data: the Rhetorical Structure Theory Discourse Treebank RST-DT (Carlson et al., 2003) and the Penn Discourse Treebank (Prasad et al., 2008). In this paper, we select the former. In RST (Mann and Thompson, 1988), a coherent context or a document is represented as a hierarchical tree structure, the leaves of which are clause-sized units called Elementary Discourse Units (EDUs). Adjacent nodes (siblings in the tree) are linked with discourse relations that are either binary (hypotactic) or multi-child (paratactic). One child of each hypotactic relation is always more salient (called the NUCLEUS); its sibling (the SATELLITE) is less salient compared and may be omitted in summarization. Multi-nuclear relations (e.g., CONJUNCTION) exhibit no distinction of salience between the units.

The RST Discourse Treebank contains 385 annotated documents (347 for training and 38 for testing) from the Wall Street Journal. A total of 110 fine-grained relations defined in (Marcu, 2000b) are used for tagging relations in RST-DT. They are subtypes of 18 original high-level RST categories. For fair comparison with existing systems, we use in this work the 18 coarse-grained relation classes, which with nuclearity attached form a set of 41 distinct relations. Non-binary relations are converted into a cascade of right-branching binary relations.

Conventionally, discourse parsing in RST-DT involves the following sub-tasks: (1) EDU segmentation to segment the raw text into EDUs, (2) tree-building. Since the segmentation task is essentially clause delimitation and hence relatively easy (with state-of-art accuracy at most 95%), we focus on the latter problem. We assume that the gold-standard EDU segmentations are already given, as assumed in other past work (Feng and Hirst, 2012).

4 EDU Model

In this section, we describe how we compute the distributed representation for a given sentence based on its parse tree structure and contained words. Our implementation is based on (Socher et al., 2013). As the details can easily be found there, we omit them for brevity.

Let s denote any given sentence, comprised of a sequence of tokens $s = \{w_1, w_2, ..., w_{n_s}\}$, where n_s denotes the number of tokens in s. Each token w is associated with a specific vector embedding $\mathbf{e_w} = \{e_w^1, e_w^2, ..., e_w^K\}$, where K denotes the dimension of the word embedding. We wish to compute the vector representation h_s for current sentence, where $h_s = \{h_s^1, h_s^2, ..., h_s^K\}$.

Parse trees are obtained using the Stanford Parser[1], and each clause is treated as an EDU. For a given parent p in the tree and its two children c_1 (associated with vector representation h_{c_1}) and c_2 (associated with vector representation h_{c_2}), standard recursive networks calculate the vector for parent p as follows:

$$h_p = f(W \cdot [h_{c_1}, h_{c_2}] + b) \quad (1)$$

where $[h_{c_1}, h_{c_2}]$ denotes the concatenating vector for children representations h_{c_1} and h_{c_2}; W is a $K \times 2K$ matrix and b is the $1 \times K$ bias vector; and $f(\cdot)$ is the function $tanh$. Recursive neural models compute parent vectors iteratively until the root node's representation is obtained, and use the root embedding to represent the whole sentence.

5 Discourse Parsing

Since recent work (Feng and Hirst, 2012; Hernault et al., 2010b) has demonstrated the advantage of combining the binary structure classifier (determining whether two adjacent text units should be merged to form a new subtree) with the multi-class classifier (determining which discourse relation label to assign to the new subtree) over the older single multi-class classifier with the additional label NO-REL, our approach follows the modern

[1] http://nlp.stanford.edu/software/lex-parser.shtml

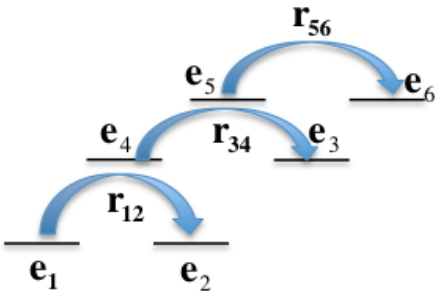


Figure 1: RST Discourse Tree Structure.

strategy but trains binary and multi-class classifiers jointly based on the discourse structure tree.

Figure 2 illustrates the structure of a discourse parse tree. Each node e in the tree is associated with a distributed vector h_e. e_1, e_2, e_3 and e_6 constitute the leaves of trees, the distributed vector representations of which are assumed to be already obtained from convolution in Section 4. Let N_r denote the number of relations and we have $N_r = 41$.

5.1 Binary (Structure) Classification

In this subsection, we train a binary (structure) classifier, which aims to decide whether two EDUs or spans should be merged during discourse tree reconstruction.

Let $t_{\text{binary}}(e_i, e_j)$ be the binary valued variable indicating whether e_i and e_j are related, or in other words, whether a certain type of discourse relations holds between e_i and e_j. According to Figure 2, the following pairs constitute the training data for binary classification:

$$t_{\text{binary}}(e_1, e_2) = 1, \;\; t_{\text{binary}}(e_3, e_4) = 1,$$
$$t_{\text{binary}}(e_2, e_3) = 0, \;\; t_{\text{binary}}(e_3, e_6) = 0,$$
$$t_{\text{binary}}(e_5, e_6) = 1$$

To train the binary classifier, we adopt a three-layer neural network structure, i.e., input layer, hidden layer, and output layer. Let $H = [h_{e_i}, h_{e_j}]$ denote the concatenating vector for two spans e_i and e_j. We first project the concatenating vector H to the hidden layer with N_{binary} hidden neurons. The hidden layer convolutes the input with non-linear tanh function as follows:

$$L^{\text{binary}}_{(e_i,e_j)} = f(G_{\text{binary}} * [h_{e_i}, h_{e_j}] + b_{\text{binary}})$$

where G_{binary} is an $N_{\text{binary}} * 2K$ convolution matrix and b_{binary} denotes the bias vector.

The output layer takes as input $L^{\text{binary}}_{(e_i,e_j)}$ and generates a scalar using the linear function $U_{binary} \cdot L^{\text{binary}}_{(e_i,e_j)} + b$. A $sigmod$ function is then adopted to project the value to a [0,1] probability space. The execution at the output layer can be summarized as:

$$p[t_{\text{binary}}(e_i, e_j) = 1] = g(U_{binary} \cdot L^{\text{binary}}_{(e_i,e_j)} + b^*_{\text{binary}}) \quad (2)$$

where U_{binary} is an $N_{\text{binary}} \times 1$ vector and b^*_{binary} denotes the bias. $g(\cdot)$ is the sigmod function.

5.2 Multi-class Relation Classification

If $t_{binary}(e_i, e_j)$ is determined to be 1, we next use variable $r(e_i, e_j)$ to denote the index of relation that holds between e_i and e_j. A multi-class classifier is train based on a three-layer neural network, in the similar way as binary classification in Section 5.1. Concretely, a matrix G_{Multi} and bias vector b_{Multi} are first adopted to convolute the concatenating node vectors to the hidden layer vector $L^{\text{multi}}_{(e_i,e_j)}$:

$$L^{\text{multi}}_{(e_i,e_j)} = f(G_{\text{multi}} * [h_{e_i}, h_{e_j}] + b_{\text{multi}}) \quad (3)$$

We then compute the posterior probability over labels given the hidden layer vector L using the softmax and obtain the N_r dimensional probability vector $P_{(e_1,e_2)}$ for each EDU pair as follows:

$$S_{(e_i,e_j)} = U_{\text{multi}} \cdot L^{\text{multi}}_{(e_i,e_j)} \quad (4)$$

$$P_{(e_1,e_2)}(i) = \frac{\exp(S_{(e_1,e_2)}(i))}{\sum_k \exp(S_{(e_1,e_2)})(k)} \quad (5)$$

where U_{multi} is the $N_r \times 2K$ matrix. The i^{th} element in $P_{(e_1,e_2)}$ denotes the probability that i^th relation holds between e_i and e_j. To note, binary and multi-class classifiers are trained independently.

5.3 Distributed Vector for Spans

What is missing in the previous two subsections are the distributed vectors for non-leaf nodes (i.e., e_4 and e_5 in Figure 1), which serve as structure and relation classification. Again, we turn to recursive deep learning network to obtain the distributed vector for each node in the tree in a bottom-up fashion.

Similar as for sentence parse-tree level compositionally, we extend a standard recursive neural network by associating each type of relations r with one specific $K \times 2K$ convolution matrix W_r.

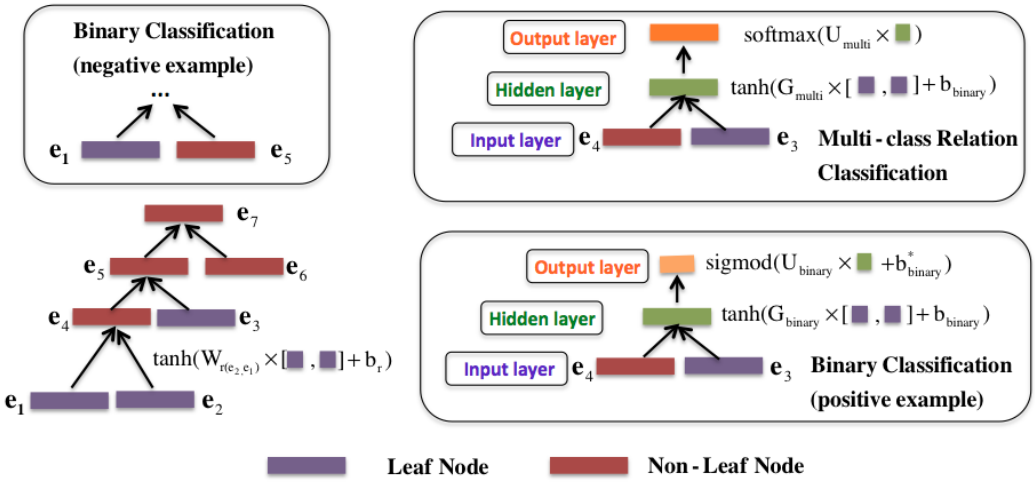


Figure 2: System Overview.

The representation for each node within the tree is calculated based on the representations for its children in a bottom-up fashion. Concretely, for a parent node p, given the distributed representation h_{e_i} for left child, h_{e_j} for right child, and the relation $r(e_1, e_2)$, its distributed vector h_p is calculated as follows:

$$h_p = f(W_{r(e_1,e_2)} \cdot [h_{e_i}, h_{e_j}] + b_{r(e_1,e_2)}) \quad (6)$$

where $b_{r(e_1,e_2)}$ is the bias vector and $f(\cdot)$ is the non-linear tanh function.

To note, our approach does not make any distinction between within-sentence text spans and cross-sentence text spans, different from (Feng and Hirst, 2012; Joty et al., 2013)

5.4 Cost Function

The parameters to optimize include sentence-level convolution parameters $[W, b]$, discourse-level convolution parameters $[\{W_r\}, \{b_r\}]$, binary classification parameters $[G_{\text{binary}}, b_{\text{binary}}, U_{\text{binary}}, b^*_{\text{binary}}]$, and multi-class parameters $[G_{\text{multi}}, b_{\text{multi}}, U_{\text{multi}}]$.

Suppose we have M_1 binary training samples and M_2 multi-class training examples (M_2 equals the number of positive examples in M_1, which is also the non-leaf nodes within the training discourse trees). The cost function for our framework with regularization on the training set is given by:

$$J(\Theta_{\text{binary}}) = \sum_{(e_i,e_j)\in\{\text{binary}\}} J_{\text{binary}}(e_i, e_j) + Q_{\text{binary}} \cdot \sum_{\theta\in\Theta_{\text{binary}}} \theta^2 \quad (7)$$

$$J(\Theta_{\text{multi}}) = \sum_{(e_i,e_j)\in\{\text{multi}\}} J_{\text{multi}}(e_i, e_j) + Q_{\text{multi}} \cdot \sum_{\theta\in\Theta_{\text{multi}}} \theta^2 \quad (8)$$

where

$$\begin{aligned} J_{\text{binary}}(e_i, e_j) &= -t(e_i, e_j) \log p(t(e_i, e_j) = 1) \\ &\quad - (1 - t(e_i, e_j)) \log[1 - p(t(e_i, e_j) = 1)] \\ J_{\text{multi}}(e_i, e_j) &= -\log[p(r(e_i, e_j) = r)] \end{aligned} \quad (9)$$

5.5 Backward Propagation

The derivative for parameters involved is computed through backward propagation. Here we illustrate how we compute the derivative of $J_{\text{multi}}(e_i, e_j)$ with respect to different parameters.

For each pair of nodes $(e_i, e_j) \in$ multi, we associate it with a N_r dimensional binary vector $R(e_i, e_j)$, which denotes the ground truth vector with a 1 at the correct label $r(e_i, e_j)$ and all other entries 0. Integrating softmax error vector, for any parameter θ, the derivative of $J_{\text{multi}}(e_i, e_j)$ with respect to θ is given by:

$$\frac{\partial J_{\text{multi}}(e_i, e_j)}{\partial \theta} = [P_{(e_i,e_j)} - R_{(e_i,e_j)}] \otimes \frac{\partial S_{(e_i,e_j)}}{\partial \theta} \quad (10)$$

where $\otimes$ denotes the Hadamard product between the two vectors. Each training pair recursively backpropagates its error to some node in the discourse tree through $[\{W_r\}, \{b_r\}]$, and then to nodes in sentence parse tree through $[W, b]$, and the derivatives can be obtained according to standard backpropagation (Goller and Kuchler, 1996; Socher et al., 2010).

5.6 Additional Features

When determining the structure/multi relation between individual EDUs, additional features are also considered, the usefulness of which has been illustrated in a bunch of existing work (Feng and Hirst, 2012; Hernault et al., 2010b; Joty et al., 2012). We consider the following simple text-level features:

- Tokens at the beginning and end of the EDUs.
- POS at the beginning and end of the EDUs.
- Whether two EDUs are in the same sentence.

5.7 Optimization

We use the diagonal variant of AdaGrad (Duchi et al., 2011) with minibatches, which is widely applied in deep learning literature (e.g.,(Socher et al., 2011a; Pei et al., 2014)). The learning rate in AdaGrad is adapted differently for different parameters at different steps. Concretely, let g_τ^i denote the subgradient at time step t for parameter θ_i obtained from backpropagation, the parameter update at time step t is given by:

$$\theta_\tau = \theta_{\tau-1} - \frac{\alpha}{\sum_{t=0}^{\tau}\sqrt{g_\tau^{i2}}} g_\tau^i \tag{11}$$

where α denotes the learning rate and is set to 0.01 in our approach.

Elements in $\{W_r\}$, W, G_{binary}, G_{multi}, U_{binary}, U_{multi} are initialized by randomly drawing from the uniform distribution $[-\epsilon, \epsilon]$, where ϵ is calculated as suggested in (Collobert et al., 2011). All bias vectors are initialized with 0. Word embeddings $\{e\}$ are borrowed from Senna (Collobert et al., 2011; Collobert, 2011).

5.8 Inference

For inference, the goal is to find the most probable discourse tree given the EDUs within the document. Existing inference approach basically include the approach adopted in (Feng and Hirst, 2012; Hernault et al., 2010b) that merges the most likely spans at each step and SPADE (Fisher and Roark, 2007) that first finds the tree structure that is globally optimal, then assigns the most probable relations to the internal nodes.

In this paper, we implement a probabilistic CKY-like bottom-up algorithm for computing the most likely parse tree using dynamic programming as are adopted in (Joty et al., 2012; Joty et al., 2013; Jurafsky and Martin, 2000) for the search of global optimum. For a document with n EDUs, as different relations are characterized with different compositions (thus leading to different vectors), we use a $N_r \times n \times n$ dynamic programming table Pr, the cell $Pr[r, i, j]$ of which represents the span contained EDUs from i to j and stores the probability that relation r holds between the two spans within i to j. $Pr[r, i, j]$ is computed as follows:

$$\begin{aligned} Pr[r, i, j] =& \max_{r_1, r_2, k} Pr[r_1, i, k] \cdot Pr[r_2, k, j] \\ & \times P(t_{\text{binary}}(e_{[i,k]}, e_{[k,j]}) = 1) \\ & \times P(r(e_{[i,k]}, e_{[k,j]}) = 1) \end{aligned} \tag{12}$$

At each merging step, a distributed vector for the merged point is calculated according to Eq. 13 for different relations. The CKY-like algorithms finds the global optimal. To note, the worst-case running time of our inference algorithm is O($N_r^2 n^3$), where n denotes the number of sentences within the document, which is much slower than the greedy search. In this work, for simplification, we simplify the framework by maintaining the top 10 options at each step.

6 Experiments

A measure of the performance of the system is realized by comparing the structure and labeling of the RS-tree produced by our algorithm to gold-standard annotations.

Standard evaluation of discourse parsing output computes the ratio of the number of identical tree constituents shared in the generated RS-trees and the gold-standard trees against the total number of constituents in the generated discourse trees[2], which is further divided to three matrices: **Span** (on the blank tree structure), **nuclearity** (on the tree structure with nuclearity indication), and **relation** (on the tree structure with rhetorical relation indication but no nuclearity indication).

The **nuclearity** and **relation** decisions are made based on the multi-class output labels from the deep learning framework. As we do not consider nuclearity when classifying different discourse relations, the two labels *attribute[N][S]* and *attribute[S][N]* made by multi-class classifier will be treated as the same relation label ATTRIBUTE.

[2]Conventionally, evaluation matrices involve precision, recall and F-score in terms of the comparison between tree structures. But these are the same when manual segmentation is used (Marcu, 2000b).

Approach	Span	Nuclearity	Relation
HILDA	75.3	60.0	46.8
Joty et al.	82.5	68.4	55.7
Feng and Hirst	**85.7**	71.0	58.2
Ji and Eisenstein	82.1	**71.1**	**61.6**
Unified (with feature)	82.0	70.0	57.1
Ours (no feature)	82.4	69.2	56.8
Ours (with feature)	84.0	70.8	58.6
human	88.7	77.7	65.7

Table 1: Performances for different approaches. Performances for baselines are reprinted from (Joty et al., 2013; Feng and Hirst, 2014; Ji and Eisenstein, 2014).

Also, we do not train a separate classifier for NUCLEUS and SATELLITE identification. The nuclearity decision is made based on the relation type produced by the multi-class classifier.

6.1 Parameter Tuning

The regularization parameter Q constitutes the only parameter to tune in our framework. We tune it on the 347 training documents. Concretely, we employ a five-fold cross validation on the RST dataset and tune Q on 5 different values: 0.01, 0.1, 0.5, 1.5, 2.5. The final model was tested on the testing set after parameter tuning.

6.2 Baselines

We compare our model against the following currently prevailing discourse parsing baselines:

HILDA A discourse parser based on support vector machine classification introduced by Hernault et al. (Hernault et al., 2010b). HILDA uses the binary and multi-class classifier to reconstruct the tree structure in a greedy way, where the most likely nodes are merged at each step. The results for HILDA are obtained by running the system with default settings on the same inputs we provided to our system.

Joty et al The discourse parser introduced by Joty et al. (Joty et al., 2013). It relies on CRF and combines intra-sentential and multi-sentential parsers in two different ways. Joty et al. adopt the global optimal inference as in our work. We reported the performance from their paper (Joty et al., 2013).

Feng and Hirst The linear-time discourse parser introduced in (Feng and Hirst, 2014) which relies on two linear-chain CRFs to obtain a sequence of discourse constituents.

Ji and Eisenstein The shift-reduce discourse parser introduced in (Ji and Eisenstein, 2014) which parses document by relying on the distributed representations obtained from deep learning framework.

Additionally, we implemented a simplified version of our model called **unified** where we use a unified convolutional function with unified parameters $[W_{sen}, b_{sen}]$ for span vector computation. Concretely, for a parent node p, given the distributed representation h_{e_i} for left child, h_{e_j} for right child, and the relation $r(e_1, e_2)$, rather than taking the inter relation between two children, its distributed vector h_p is calculated:

$$h_p = f(W_{sen} \cdot [h_{e_i}, h_{e_j}] + b_{sen}) \qquad (13)$$

6.3 Performance

Performances for different models approaches reported in Table 1. And as we can observe, although the proposed framework obtains comparable result compared with existing state-of-state performances regarding all evaluating parameters for discourse parsing. Specifically, as for the three measures, no system achieves top performance on all three, though some systems outperform all others for one of the measures. The proposed system achieves high overall performance on all three, although it does not achieve top score on any measure. The system gets a little bit performance boost by considering text-level features illustrated in Section 5.6. The simplified version of the original model underperforms against the original approach due to lack of expressive power in convolution. Performance plummets when different relations are uniformly treated, which illustrates the importance of taking into consideration different types of relations in the span convolution procedure.

7 Conclusion

In this paper, we describe an RST-style text-level discourse parser based on a neural network model. The incorporation of sentence-level distributed vectors for discourse analysis obtains comparable performance compared with current state-of-art discourse parsing system.

Our future work will focus on extending discourse-level distributed presentations to related

tasks, such as implicit discourse relation identification or dialogue analysis. Further, once the tree structure for a document can be determined, the vector for the entire document can be obtained in bottom-up fashion, as in this paper. One can now investigate whether the discourse parse tree is useful for acquiring a single document-level vector representation, which would benefit multiple tasks, such as document classification or macro-sentiment analysis.

Acknowledgements

The authors want to thank Vanessa Wei Feng and Shafiq Joty for helpful discussions regarding RST dataset. We also want to thank Richard Socher, Zhengyan He and Pradeep Dasigi for the clarification of deep learning techniques.

References

Jason Baldridge and Alex Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 96–103. Association for Computational Linguistics.

Samuel R Bowman. 2013. Can recursive neural tensor networks learn logical reasoning? *arXiv preprint arXiv:1312.6192*.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. *Building a discourse-tagged corpus in the framework of rhetorical structure theory*. Springer.

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

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics*, number EPFL-CONF-192374.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

David A Duverle and Helmut Prendinger. 2009. A novel discourse parser based on support vector machine classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 665–673. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 60–68. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear time bottom-up discourse parser with constraints and post-editing. In *ACL*.

Seeger Fisher and Brian Roark. 2007. The utility of parse-derived features for automatic discourse segmentation. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 488.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE.

Hugo Hernault, Danushka Bollegala, and Mitsuru Ishizuka. 2010a. A semi-supervised approach to improve classification of infrequent discourse relations using feature vector extension. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 399–409. Association for Computational Linguistics.

Hugo Hernault, Helmut Prendinger, Mitsuru Ishizuka, et al. 2010b. Hilda: a discourse parser using support vector machine classification. *Dialogue & Discourse*, 1(3).

Wang Houfeng, Longkai Zhang, and Ni Sun. 2013. Improving chinese word segmentation on microblog using rich punctuations.

Eduard H Hovy and Elisabeth Maier. 1997. Parsimonious or profligate: How many and which discourse structure relations. *Discourse Processes*.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing.

Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. 2012. A novel discriminative framework for sentence-level discourse analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 904–915. Association for Computational Linguistics.

Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Yashar Mehdad. 2013. Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. In *Proceedings of the 51st annual meeting of the association for computational linguistics (ACL)*, pages 486–496.

Dan Jurafsky and James H Martin. 2000. *Speech & Language Processing*. Pearson Education India.

Huong LeThanh, Geetha Abeysinghe, and Christian Huyck. 2004. Generating discourse structures for written texts. In *Proceedings of the 20th international conference on Computational Linguistics*, page 329. Association for Computational Linguistics.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 343–351. Association for Computational Linguistics.

Annie Louis, Aravind Joshi, and Ani Nenkova. 2010. Discourse indicators for content selection in summarization. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 147–156. Association for Computational Linguistics.

Minh-Thang Luong, Michael C Frank, and Mark Johnson. 2014. Parsing entire discourses as very long strings: Capturing topic continuity in grounded language learning.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Daniel Marcu. 2000a. The rhetorical parsing of unrestricted texts: A surface-based approach. *Computational Linguistics*, 26(3):395–448.

Daniel Marcu. 2000b. *The theory and practice of discourse parsing and summarization*. MIT Press.

Wenzhe Pei, Tao Ge, and Chang Baobao. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of ACL*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*. Citeseer.

David Reitter. 2003. Simple signals for complex rhetorics: On rhetorical analysis with rich-feature support vector models. In *LDV Forum*, volume 18, pages 38–52.

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, volume 24, pages 801–809.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011b. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 129–136.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

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

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics.

Caroline Sporleder and Mirella Lapata. 2005. Discourse chunking and its application to sentence compression. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 257–264. Association for Computational Linguistics.

Rajen Subba and Barbara Di Eugenio. 2009. An effective discourse parser that uses rich linguistic information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 566–574. Association for Computational Linguistics.

Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2007. Evaluating discourse-based answer extraction for why-question answering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 735–736. ACM.

Longkai Zhang Houfeng Wang and Xu Sun Mairgup Mansur. 2013. Exploring representations from unlabeled data with co-training for chinese word segmentation.

Recall Error Analysis for Coreference Resolution

Sebastian Martschat and **Michael Strube**
Heidelberg Institute for Theoretical Studies gGmbH
Schloss-Wolfsbrunnenweg 35, 69118 Heidelberg, Germany
(sebastian.martschat|michael.strube)@h-its.org

Abstract

We present a novel method for coreference resolution error analysis which we apply to perform a recall error analysis of four state-of-the-art English coreference resolution systems. Our analysis highlights differences between the systems and identifies that the majority of recall errors for nouns and names are shared by all systems. We characterize this set of common challenging errors in terms of a broad range of lexical and semantic properties.

1 Introduction

Coreference resolution is the task of determining which mentions in a text refer to the same entity. State-of-the-art approaches include both learning-based (Fernandes et al., 2012; Björkelund and Farkas, 2012; Durrett and Klein, 2013) and deterministic models (Lee et al., 2013; Martschat, 2013). These approaches achieve state-of-the-art performance mainly relying on morphosyntactic and lexical factors. However, consider the following example.

> In order to improving the added value of oil products, the second phase project of **the Qinghai Petroleum Bureau's Ge'ermu oil refinery** has been put into production. This will further improve **the factory's** oil products structure.

Due to the lack of any string overlap, most state-of-the-art systems will miss the link between *the factory* and *the Qinghai Petroleum Bureau's Ge'ermu oil refinery*. The information that *factory* is a hypernym of *refinery*, however, may be useful to resolve such links.

The aim of this paper is to quantify and characterize such recall errors made by state-of-the-art coreference resolution systems. By doing so, we provide a solid foundation for work on employing knowledge sources for improving recall for coreference resolution (Ponzetto and Strube, 2006; Rahman and Ng, 2011; Ratinov and Roth, 2012; Bansal and Klein, 2012, inter alia). In particular, we make the following contributions:

We present a novel framework for coreference resolution error analysis. This yields a formal foundation for previous work on link-based error analysis (Uryupina, 2008; Martschat, 2013) and complements work on transformation-based error analysis (Kummerfeld and Klein, 2013).

We apply the method proposed in this paper to perform a recall error analysis of four state-of-the-art systems, encompassing deterministic and learning-based approaches. In particular, we identify and characterize a set of challenging errors common to all systems, and discuss strengths and weaknesses of each system regarding specific error types. We also present a brief precision error analysis.

A toolkit which implements the framework proposed in this paper is available for download.[1]

2 A Link-Based Analysis Framework

In this section we discuss challenges in coreference resolution error analysis and devise an error analysis framework to overcome these challenges.

2.1 Motivation

Suppose a document contains the entity BARACK OBAMA, which is referenced by four mentions in the following order: *Obama*, *he*, *the president* and *his*. A typical output of a current system not equipped with world knowledge will consist of two entities: {*Obama*, *he*} and {*the president*, *his*}

Obviously, the system made a recall error. But, due to the complex nature of the coreference resolution task, it is not clear how to represent the re-

[1] http://smartschat.de/software

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2070–2081,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

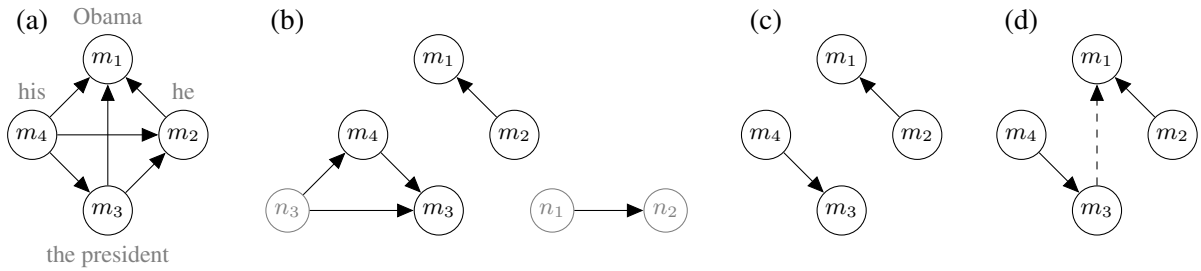


Figure 1: (a) a reference entity r, represented as a complete one-directional graph, (b) a set S of three system entities, (c) the partition r_S, (d) a spanning tree for r.

call error: is it missing the link between *the president* and *Obama*? Can the error be attributed to deficiencies in pronoun resolution?

Linguistically motivated error representations would facilitate both understanding of current challenges and make system development faster and easier. The aim of this section is to devise such representations.

2.2 Formalizing Coreference Resolution

To start with, we give a formal description of the coreference resolution task following the terminology used for the ACE (Mitchell et al., 2004) and OntoNotes (Weischedel et al., 2013) projects. A *mention* is a linguistic realization of a reference to an entity. Two mentions *corefer* if they refer to the same entity. Hence, coreference is reflexive, symmetric and transitive, and therefore an *equivalence relation*. The task of coreference resolution is to predict equivalence classes of mentions in a document according to the coreference relation.

In order to extract errors, we need to compare the *reference equivalence classes*, given by the annotation, with the *system equivalence classes* obtained from system output. The key question now is how we represent these equivalence classes of mentions. Adapting common terminology, we also refer to the equivalence classes as entities.

2.3 Representing Entities

The most straightforward entity representation ignores any structure and models an entity as a set of mentions. This representation was utilized for error analysis by Kummerfeld and Klein (2013), who extract errors by transforming reference into system entities. In this set-based representation, we can only extract whether two mentions corefer at all. More fine-grained information, for example about antecedent information, is not accessible.

We therefore propose to employ a *structured* entity representation, which explicitly models links established by the coreference relation between mentions. This leads to a link-based error representation which formalizes the methods presented in Uryupina (2008) and Martschat (2013).

We employ for representation a *complete one-directional graph*. That is, we represent an entity e over mentions $\{m_1, \ldots, m_n\}$ as a graph $e = (N, A)$, where $N = \{m_1, \ldots, m_n\}$ and $A = \{(m_k, m_j) \mid k > j\}$. The indices respect the mention ordering. Mentions earlier in the text have a lower index. An example graph for an entity over four mentions $m_1, \ldots, m_4$ (such as the BARACK OBAMA entity) is depicted in Figure 1a. In this graph, we express all coreference relations between all pairs of mentions.[2]

Using this representation, we can represent a set of entities as a set of graphs. In particular, given a document we consider the set of *reference entities* R given by the annotation, and the set of *system entities* S, given by the system output. In order to extract errors, we compare the graphs in R with the graphs in S.

In the following, we discuss how to compute recall errors for a reference entity $r \in R$ with respect to the system entities S. For computing precision errors, we just switch the roles of R and S.

2.4 Comparing Reference and System Entities

As we represent entities as sets of links between mentions, errors can be quantified as differences in the links. For example, if an edge (representing a link) from some reference entity $r \in R$ is missing

[2]We could also use an undirected instead of a one-directional graph, but using a one-directional graph conveniently models sequential information, which simplifies notation and the algorithms we will present.

in all system entities in S, this is a recall error.

In order to formalize this, we employ the notion of a *partition* of an entity. Let $r \in R$ be some reference entity, and let S be a set of system entities. The partition of r by S, written r_S, is obtained by taking all edges in r that also appear in S. r_S consists of all connected components of r (we will refer to these as *subentities*) that are also in S. All edges in r that are not in r_S are candidates for recall errors, as these were not in any entity in S.

Figure 1b shows a set S of three system entities: two consist of two mentions, one of three mentions. In our running example, this corresponds to the system output {*Obama*, *he*} and {*the president*, *his*} plus some spurious mentions, which are colored gray. The graph r_S for our example is shown in Figure 1c. The two edges correspond to the correctly recognized links (*he*, *Obama*) and (*his*, *the president*). All edges in r (Figure 1a) missing from this graph are candidates for errors.

2.5 Spanning Trees

However, taking all edges in r missing in r_S as errors leads to unintuitive results. In the BARACK OBAMA example, this would lead to four errors being extracted: (*the president*, *Obama*), (*his*, *Obama*), (*the president*, *he*) and (*his*, *he*). But, in order to correctly predict the BARACK OBAMA entity, a coreference resolution system only needs to predict three correct links, i.e. it has to provide a *spanning tree* of the entity's graph representation.

Therefore, to extract errors, we compute a spanning tree T_r of r, and take all edges in T_r that do not appear in r_S as errors. Figure 1d shows an example spanning tree for the running example entity r. The dashed edge, which corresponds to the link (*the president*, *Obama*), does not appear in r_S and is therefore extracted as an error.

The strategies for computing a spanning tree may differ for recall and precision errors. Hence, our extraction algorithm is parametrized by two procedures $\texttt{ST}_{\text{rec}}(e, P)$ and $\texttt{ST}_{\text{prec}}(e, P)$ which, given an entity e and a set of entities P, output a spanning tree T_e of e. The whole algorithm for error extraction is summarized in Algorithm 1.

Algorithm 1 Error Extraction from a Corpus

Input: A corpus $\mathcal{C}$, algorithms $\texttt{ST}_{\text{rec}}$, $\texttt{ST}_{\text{prec}}$ for computing spanning trees.

function ERRORS($\mathcal{C}$, $\texttt{ST}_{\text{rec}}$, $\texttt{ST}_{\text{prec}}$)

`recall_errors` $= [\,]$

`precision_errors` $= [\,]$

for $d \in \mathcal{C}$ **do**

Let R_d be the reference entities and S_d be the system entities of document d.

for $r \in R_d$ **do**

Add all edges in $\texttt{ST}_{\text{rec}}(r, S_d)$ not in r_{S_d} to `recall_errors`.

for $s \in S_d$ **do**

Add all edges in $\texttt{ST}_{\text{prec}}(s, R_d)$ not in s_{R_d} to `precision_errors`.

Output: `recall_errors`, `precision_errors`

3 Spanning Tree Algorithms

In the last section we presented a framework for link-based error analysis, which extracts errors by comparing entity spanning trees to entity partitions. Therefore we can accommodate different notions of errors by varying the algorithm for computing spanning trees. We now present some spanning tree algorithms for extracting recall and precision errors.

3.1 Recall Errors

We first observe that for error extraction, the structure of the spanning trees of the subentities appearing in r_S does not play a role. Edges present in r_S are not candidates for errors, since they appear in both the reference entity r and the system output S. Therefore, it does not matter which edges from the subentities are in the spanning tree.

Hence, to build the spanning tree, we first choose arbitrary spanning trees for the subentities in the partition. We choose the remaining edges according to the spanning tree algorithm.

Having settled on this, we only have to decide which edges to choose that connect the trees representing the subentities. There are many possible choices for this. For example, the graph in Figure 1c has four candidate edges which connect the trees for the subentities.

We can reduce the number of candidate edges by only considering the first mention (with respect to textual order) in a subentity as the source of an edge to be added. This makes sense since all other mentions in that subentity were correctly resolved to be coreferent with some preceding mention. We still have to decide on the target of the edge. In Figure 1c, we have two choices for edges: (m_3, m_1) and (m_3, m_2). We now present two methods for choosing edges.

Choosing Edges by Distance. The most straight-forward way to decide on an edge is to take the edge with smallest mention distance between source and target. This is the approach taken by Martschat (2013).

Choosing Edges by Accessibility. However, the distance-based approach may lead to unintuitive results. Let us consider again the BARACK OBAMA example from Figure 1. When choosing edges by distance, we would extract the error (*the president*, *he*). However, such links with a non-pronominal anaphor and a pronominal antecedent are difficult to process and considered unreliable (Ng and Cardie, 2002; Bengtson and Roth, 2008). On the other hand, the missed link (*the president*, *Obama*) constitutes a well-defined hyponymy relation which can be found in knowledge bases and is easily interpretable by humans.

Uryupina (Uryupina, 2007; Uryupina, 2008) presents a recall error analysis where she takes the "intuitively easiest" missing link to analyze (Uryupina, 2007, p. 196). How can we formalize such an intuition? We will employ a notion grounded in accessibility theory (Ariel, 1988). Names and nouns refer to less accessible entities than pronouns do. For such anaphors, we prefer descriptive (name/nominal) antecedents. Inspired by Ariel's degrees of accessibility, we choose a target for a given anaphor m_i as follows:

- If m_i is a pronoun, choose the closest preceding mention.
- If m_i is not a pronoun, choose the closest preceding proper name. If no such mention exists, choose the closest preceding common noun. If no such mention exists, choose the closest preceding mention.

Applied to the example from Figure 1, this algorithm extracts the error (*the president*, *Obama*).[3]

3.2 Precision Errors

Virtually all approaches to coreference resolution obtain entities by outputting pairs of anaphor and antecedent, subject to the constraint that one anaphor has at most one antecedent.

We use this information to build spanning trees for system entities: these spanning trees consist of exactly the edges which correspond to anaphor/antecedent pairs in the system output.

[3] A similar procedure was used by Ng and Cardie (2002) to extract meaningful antecedents when training a coreference resolution system.

4 Data and Systems

We now discuss data and coreference resolution systems which we will employ for our analysis.

4.1 Data

We analyze the errors of the systems on the English development data of the CoNLL'12 shared task on multilingual coreference resolution (Pradhan et al., 2012). This corpus contains 343 documents, spanning seven genres: bible texts, broadcast conversation, broadcast news, magazine texts, news wire, telephone conversations and web logs.

4.2 Systems

State-of-the-art approaches to coreference resolution encompass various paradigms, ranging from deterministic pairwise systems to learning-based structured prediction models. Hence, we want to conduct our analysis on a representative sample of the state of the art, which should be publicly available. Therefore, we decided on two deterministic and two learning-based systems:

- **StanfordSieve**[4] (Lee et al., 2013) was the winning system of the CoNLL'11 shared task. It employs a multi-sieve approach by making more confident decisions first.
- **Multigraph**[5] (Martschat, 2013) is a deterministic pairwise system which is based on Martschat et al. (2012), the second-ranking system in the English track of the CoNLL'12 shared task. It uses a subset of features as hard constraints and chooses an antecedent for a mention by summing up the remaining boolean features.
- **IMSCoref**[6] (Björkelund and Farkas, 2012) ranked second overall in the CoNLL'12 shared task (third for English). It stacks multiple decoders and relies on a combination of standard pairwise and lexicalized features.
- **BerkeleyCoref**[7] (Durrett and Klein, 2013) is a state-of-the-art system that uses mainly lexicalized features and a latent antecedent ranking architecture. It outperforms StanfordSieve and IMSCoref on the CoNLL'11 data.

[4] Part of Stanford CoreNLP, available at `http://nlp.stanford.edu/software/corenlp.shtml`. We use version 3.4.

[5] `http://smartschat.de/software`

[6] `http://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/IMSCoref.en.html` . We use the *CoNLL 2012 system*.

[7] `http://nlp.cs.berkeley.edu/berkeleycoref.shtml`

System	MUC	B^3	$CEAF_e$	Average
Fernandes et al.	69.46	57.83	54.43	60.57
Martschat	66.22	55.47	51.90	57.86
StanfordSieve	64.96	54.49	51.24	56.90
Multigraph	69.13	58.61	56.06	61.28
IMSCoref	67.15	55.19	50.94	57.76
BerkeleyCoref	70.27	59.29	56.11	61.89

Table 1: Comparison of the systems with Fernandes et al. (2012) and with Martschat (2013) on CoNLL'12 English development data.

For Multigraph, we modified the system described in Martschat (2013) slightly to allow for the incorporation of distance (similar to Cai and Strube (2010)). Inspired by Lappin and Leass (1994), we add salience weights for subjects and objects to the model to improve third-person pronoun resolution. We also extended the feature set by a substring feature. Furthermore, motivated by Chen and Ng (2012), we added a lexicalized feature for non-pronominal mentions that were coreferent in at least 50% of the cases in the training data.

StanfordSieve was run with its standard CoNLL shared task settings. The learning-based systems were trained on the CoNLL'12 training data. We trained IMSCoref with its standard settings, and trained BerkeleyCoref with the *final* feature set from Durrett and Klein (2013) for twenty iterations. We evaluate the systems on English CoNLL'12 development data and compare it with the winning system of the CoNLL'12 shared task (Fernandes et al., 2012) and with Martschat (2013) in Table 1, using the reference implementation v7 of the CoNLL scorer (Pradhan et al., 2014).

BerkeleyCoref performs best according to all metrics, followed by Multigraph. StanfordSieve is the worst performing system: the gap to BerkeleyCoref is five points in average score.

4.3 Discussion

Although we analyze recent systems on a recently published coreference data set, we believe that the results of our analysis will have implications for coreference in general. The data set is the largest and most genre-diverse coreference corpus so far. The systems we investigate represent major directions in coreference resolution model research, and make use of large and diverse feature sets proposed in the literature (Ng, 2010).

5 A Comparative Analysis

The coreference resolution systems presented in the previous section are a representative sample of the state of the art. Therefore, by analyzing the errors they make, we can learn about remaining challenges in coreference resolution and analyze the qualitative differences between the systems. The results of such an analysis will deepen our understanding of coreference resolution and will suggest promising directions for further research.

5.1 Experimental Settings

Previous studies identified the presence of recall errors as a main bottleneck for improving performance (Raghunathan et al., 2010; Durrett and Klein, 2013; Kummerfeld and Klein, 2013). This is also evidenced by the CoNLL shared tasks on coreference resolution (Pradhan et al., 2011; Pradhan et al., 2012), where most competitive systems had higher precision than recall. This indicates that an analysis of recall errors helps to understand and improve the state of the art. Hence, we focus on analyzing recall errors, and complement this by a brief analysis of precision errors.

We analyze errors of the four systems presented in the previous section on the CoNLL'12 English development data. To extract recall errors we employ the spanning tree algorithm which **chooses edges by accessibility**. We obtain precision errors from the pairwise output of the systems.

5.2 A Recall Error Analysis of StanfordSieve

Since StanfordSieve is currently the most-widely used coreference resolution system, it serves as a good starting point for our analysis. Remember that we represent each error as a pair of anaphor and antecedent. For an initial analysis, we categorize each error by mention type, distinguishing between proper name, common noun, pronoun, demonstrative pronoun and verb.[8]

StanfordSieve makes 5245 recall errors. To put this number into context, we compare it with the maximum number of recall errors a system can make. This count is obtained by extracting recall errors from the output of a system that puts each mention in its own entity, which yields 14609 errors. In Table 2 we present a detailed analysis. For each pair of mention type of anaphor and an-

[8]We obtain the type from the part-of-speech tag of the mention's head. Furthermore, we treat every mention whose head has a NER label in the data as a proper name.

	Name	Noun	Pron.	Dem.	Verb
Name					
Errors	1006	181	43	0	0
Maximum	3578	206	56	2	0
Noun					
Errors	517	1127	46	14	91
Maximum	742	2063	51	14	91
Pron.					
Errors	483	761	543	45	53
Maximum	1166	1535	4596	92	53
Dem.					
Errors	23	86	41	31	117
Maximum	27	93	43	46	117
Verb					
Errors	1	20	2	4	10
Maximum	1	20	2	4	11

Table 2: Number of StanfordSieve's recall errors according to mention type, compared to the maximum possible number of errors. Rows are anaphors, columns antecedents.

tecedent, the table displays the number of recall errors and of maximum errors possible.

StanfordSieve gets almost none of the links involving verbal or demonstrative mentions correct. This is due to the system not attempting to handle event coreference, and performing very poorly for demonstratives. On the other hand, recall for pronoun resolution is quite good, at least when considering non-verbal antecedents. While StanfordSieve makes 1885 recall errors when the anaphor is a pronoun, it successfully resolves most of such links present in the corpus. Finally, let us consider the links involving only proper names and common nouns. In total, these amount to 6589 links in the corpus (around 45% of all links). StanfordSieve misses 2831 of these links. Pairs of proper names seem to be easier to resolve than pairs of common nouns. Links between a common noun and a proper name are less frequent, but much more difficult: most of the links are missing.

5.3 Analysis of the Other Systems

In the previous section we identified various characteristics of the errors made by StanfordSieve: only (comparatively) few errors are made for pronoun resolution and name coreference, while other types of nominal anaphora and coreference of demonstrative/verbal mentions pose a challenge for the system. Do the other systems in our study also have these characteristics? In order to answer this question, we repeated the analysis for the three other systems described in Section 4. We summarize the results in Table 3. We only report numbers for pronoun resolution and name/noun coreference, as all systems do not resolve verbal mentions and perform poorly for demonstratives.

		Proportion	
System	**Total**	**Anaphor Pron.**	**Name/Noun**
StanfordSieve	5245	36%	54%
Multigraph	4630	32%	56%
IMSCoref	5220	32%	58%
BerkeleyCoref	4635	32%	56%

Table 3: Recall error numbers for all systems.

StanfordSieve makes the most recall errors, closely followed by IMSCoref. Multigraph and BerkeleyCoref make around 600 errors less. While the total number of errors differs between the systems, the distributions are similar. In particular, around 55% of recall errors made involve only proper names and common nouns. The number is a bit higher for IMSCoref. We conclude that, despite variations in performance, both deterministic and learning-based state-of-the-art systems have similar weaknesses regarding recall.

The results displayed in Table 3 suggest various opportunities for future research. In this paper, we will focus on analyzing name/noun recall errors, as these constitute a large fraction of all recall errors. Future work should address the pronoun resolution errors and a characterization of the verbal/demonstrative errors.

5.4 Analysis of the Name/Noun Recall Errors

We now turn towards a fine-grained analysis of the name/noun recall errors.

Table 4 displays the number of such recall errors made by each system, according to the mention types of anaphor and antecedent. We are interested in errors common to all systems, and in qualitative differences of errors between the systems.

5.4.1 Common Errors

Let us first analyze the errors common to all systems. Our analysis is driven by the question how these can be characterized, and which knowledge is missing to resolve such links. We discuss the errors depending on the mention types of anaphor and antecedent. The lower part of Table 4 displays the number of common errors for each category.

Description	Number of Recall Errors (Anaphor-Antecedent)			
	Name-Name	Noun-Name	Name-Noun	Noun-Noun
StanfordSieve	1006	517	181	1127
Multigraph	753	501	189	1152
IMSCoref	1082	500	188	1264
BerkeleyCoref	910	456	171	1072
Common errors	475	371	147	835
Correct boundaries idenfified	257	273	108	563
excl. IMSCoref	156	222	97	475

Table 4: Name/Noun recall errors for all systems.

Common		All	
Type	%	Type	%
ORG	25%	PERSON	26%
PERSON	19%	GPE	26%
GPE	16%	ORG	20%
DATE	14%	NONE	14%
NONE	9%	DATE	6%

Table 5: Distribution of top five named entity types of common name-name recall errors and all possible name-name recall errors.

Furthermore, in order to assess the impact of mention detection, the table shows the number of common errors where boundaries for both mentions were identified correctly by some system. We can see that boundary identification is a difficult problem, especially for proper name pairs: for 48% of such errors, no system found the correct boundaries of both mentions participating in the error. The number of errors where correct boundaries could be found drops significantly after excluding IMSCoref. This is due to the mention extraction strategy of IMSCoref: the other systems in our study discard the shorter mention when two mentions have the same head, IMSCoref keeps both mentions. Hence, the system is able to correctly identify some mentions even in the presence of parsing or preprocessing errors. However, as a result, IMSCoref has to process many spurious mentions, which makes learning more difficult.

We conclude that mention detection still constitutes a challenge. We now proceed to a detailed analysis of errors common to all systems. In passing we will discuss difficulties in mention detection with regard to specific error types.

Errors between Pairs of Proper Names. The systems share 475 recall errors between pairs of proper names. In Table 5, we compare the distribution of gold named entity types of these errors with the distribution of gold named entity types of all possible errors (obtained via a singleton system). We see that especially difficult classes of links are pairs with type ORG or DATE.

Let us now consider lexical features of the errors.[9] In 154 errors, the strings match completely, but the correct resolution was mostly prevented by annotation inconsistencies (e.g. *China* instead of *China's*) or propagated parsing and NER errors, which lead to deficiencies in mention extraction.

For 217 errors, at least one token appears in both mention strings, as in *the "Cole"* and *the "USS Cole"*. This shows the insufficiency of the features which hint to alias relations, may it be heuristics or learned lexical similarities (for 109 of the 217 errors, both mention boundaries were identified correctly by at least one system). Disambiguation with respect to knowledge bases could provide a principled way to identify name variations.

We classified the remaining 104 errors manually, see Table 6. For a couple of categories such as identifying acronyms, spelling variations and aliases, disambiguation could also help. Many errors happen for date mentions, which suggests the use of temporal tagging features.

Errors for Noun-Name Pairs. We now investigate the errors where the anaphor is a common noun and the antecedent is a proper name. 371 errors are common to all systems. The high fraction of common errors shows that this is an especially challenging category. We again start by investigating how the distribution of the named entity type

[9] When computing these, we ignored case and ignored all tokens with part-of-speech tag DT or POS.

Description	Occ.	Example
Acronyms	20	*National Ice Hockey League* and *NHL*
Alias	24	*Florida* and *the Sunshine State*
Annotation	2	Annotation errors (pronoun as name)
Context	2	*Paula Coccoz* and *juror number ten*
Date	29	*1989* and *last year's*
Metonymy	12	*South Afria* and *Pretoria*
Roles	8	*Al Gore* and *the Vice President*
Spelling	7	*Hsiachuotzu* and *Hsiachuotsu*

Table 6: Classification of common name-name recall errors without common tokens.

Common		All	
Type	%	Type	%
ORG	28%	ORG	27%
PERSON	22%	GPE	22%
GPE	19%	PERSON	18%
NONE	7%	NONE	11%
DATE	5%	DATE	5%

Table 7: Distribution of top five named entity types of common noun-name recall errors and all possible noun-name recall errors.

of the antecedent differs when we compare common errors to all possible errors. The results are shown in Table 7. Links with a proper name antecedent of type PERSON are especially difficult. They constitute 22% of the common errors, but only 18% of all possible errors.

Most mentions are in a hyponymy relation, like *the prime minister* and *Mr. Papandreou*. This confirms that harnessing such relations could improve coreference resolution (Rahman and Ng, 2011; Uryupina et al., 2011). For 65 of the errors (18%) there is lexical overlap: the head of the anaphor is contained in the proper name antecedent, as in *the entire park* and *the Ocean Park*.

When categorizing all common errors according to the head of the anaphor, we observe 204 different heads. 142 heads appear only once, but the top ten heads make up 88 of the 371 errors. The most frequent heads are *company* (15), *group* (12), *government*, *country* and *nation* (each 9). This suggests that even with few reliable hyponymy relations recall could be significantly improved.

We observe similar trends when the anaphor is a proper name and the antecedent is a noun.

	Reference System			
System	Stanford	MG	IMS	Berkeley
Stanford	-	51	47	61
MG	17	-	42	60
IMS	26	54	-	54
Berkeley	12	42	25	-

Table 8: Comparison of noun-name recall errors. Entries are errors made by the system in the row, while the participating mentions are coreferent according to the the system in the column.

Errors between Pairs of Common Nouns. 835 errors between pairs of common nouns are shared by all systems. For 174 of these, the anaphor is an indefinite noun phase, which makes resolution a lot harder, since most coreference resolution systems classify these as non-anaphoric and therefore do not attempt resolution.

For further analysis, we split all 835 errors in two categories, distinguishing whether the head matches between the mentions or not. In 341 cases the heads match. For many of these cases, parsing errors propagate and prevent the systems from recognizing the correct mention boundaries.

In order to get a better understanding of the errors for nouns with different heads, we randomly extracted 50 of the 494 pairs and investigated the relation that holds between the heads. In 23 cases, the heads were related via hyponymy. In 10 cases they were synonyms. The remaining 17 cases involve many different phenomena, for example meronymy. This confirms findings from previous research (Vieira and Poesio, 2000).

Hence, looking up lexical relations, especially hyponymy, might be helpful to solve these cases.

5.4.2 Differences between the Systems

In order to analyze differences between the systems, we compare the recall errors they make. The information how recall errors differ between systems will enable us to understand individual strengths and weaknesses.

Exemplarily, we will have a look at the differences in the errors when the anaphor is a common noun and the antecedent is a proper name. By system design and by the total error numbers (Table 4) we expect the learning-based systems to have a slight advantage over the deterministic systems.

In Table 8 we compare noun-name recall errors made by each system. Entries are errors made by

	Number and Proportion of Precision Errors (Anaphor-Antecedent)							
Description	**Name-Name**		**Noun-Name**		**Name-Noun**		**Noun-Noun**	
StanfordSieve	1038	31%	64	59%	65	72%	944	48%
Multigraph	1131	30%	76	51%	24	56%	743	42%
IMSCoref	834	26%	74	59%	46	64%	1050	54%
BerkeleyCoref	810	24%	191	67%	60	62%	1015	48%
Common errors	158		1		2		167	

Table 9: Name/Noun precision errors for all systems. The percentages are the proportion of precision errors with respect to all decision of the system in that category.

the system in the row, while the participating mentions are coreferent according to the the system in the column. The numbers confirm our hypothesis, but also show that the deterministic systems are able to recover a few links missed by the learning-based systems.

For example, BerkeleyCoref recovers 60 links that could not be found by Multigraph, including 34 links without any common token, such as *the airline* and *Pan Am*. Multigraph recovers only 42 links not found by BerkeleyCoref, 21 without any common token. Qualitatively, StanfordSieve and Multigraph are able to resolve a few links thanks to their engineered substring match, such as *the judge* and *Dallas District Judge Jack Hampton*.

We also conducted similar investigations for common noun and proper name pairs. For common nouns, the trends are similar: the learning-based systems have an advantage over the deterministic systems. However, only few relations between nouns with different heads are learned – compared to StanfordSieve, BerkeleyCoref recovers only 11 such pairs, such as *the man* and *an expert in the law*. Recall of the deterministic systems is further hampered by their strict checks for modifier agreement, which they employ to keep precision high. Both systems miss for example the link from the anaphor *the Milosevic regime* to *the regime*, since the nominal modifier of the anaphor does not appear in the antecedent.

For proper names, Multigraph employs sophisticated alias heuristics which help to resolve matches such as *Marshall Ye Ting's* and *his grandfather Ye Ting*. This explains the corresponding low number in Table 4. The lexicalized features of Multigraph, IMSCoref and BerkeleyCoref help to learn aliases when there is no string match, especially for the bible part of the corpus (resolving links such as *Jesus* and *the Son of Man*).

5.5 Precision Errors

In the above analysis we identified common name/noun recall errors and discussed strengths and weaknesses of each system. Let us complement this analysis by a brief discussion of corresponding precision errors.

Table 9 gives an overview. It displays the number of precision errors for each category, and the proportion of these errors compared to all decisions in that category. We can see some general trends from this table: first, more decisions lead to a higher proportion of errors. This shows the difficulty of balancing recall and precision. Second, proper name coreference seems much easier than common noun coreference. Coreference involving different mention types is a lot harder – the systems only attempt few decisions, most of them are wrong. This confirms findings from our recall error analysis. Third, the fraction of common errors is very low, which indicates that precisions errors stem from various sources, which are handled differently by each system.

6 Related Work

We now discuss related work in coreference resolution error analysis and in the related field of coreference resolution evaluation metrics.

Error Analysis. While many papers on coreference resolution briefly discuss errors made and resolved by the system under consideration, only few concentrate on error analysis. Uryupina (2008) presents a manual error analysis on the small MUC-7 test set; Martschat (2013) performs an automatic coarse-grained error classification on CoNLL data. By extending and formalizing the approach of Martschat (2013), we are able to perform a large-scale investigation of recall errors made by state-of-the-art systems.

Kummerfeld and Klein (2013) devise a method to extract errors from transformations of reference to system entities. They apply this method to a variety of systems and aggregate errors over these systems. By aggregating, they are not able to analyze differences. They furthermore focus on describing many different error classes, instead of closely investigating particular phenomena.

Evaluation Metrics. We extract recall and precision errors. How does our error analysis framework relate to coreference resolution evaluation metrics, which *quantify* recall and precision errors? We first observe a fundamental difference: evaluation metrics deal with *scoring* coreference chains, they provide no means of extracting recall or precision errors. Therefore our analysis complements insights obtained via evaluation metrics.

We follow Chen and Ng (2013) and distinguish between *linguistically agnostic metrics*, which do not employ linguistic information during scoring, and *linguistically informed metrics*, which employ linguistic information similar as we do when computing spanning trees.

We limit the discussion of linguistically agnostic metrics to the three most popular evaluation metrics whose average constitutes the official score in the CoNLL shared tasks on coreference resolution: MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998) and $CEAF_e$ (Luo, 2005).[10]

Our framework bears most similarities to the MUC metric, as both are based on the same link-based entity representation. In particular, when we divide the number of errors extracted from an entity by the size of a spanning tree for that entity, we obtain a score linearly related[11] to the MUC score for that entity (recall for reference entities, precision for system entities). B^3 and $CEAF_e$ are not founded on a link-based structure. B^3 computes recall by computing the relative overlap of reference and system entity for each reference mention, and then normalizes by the number of mentions. $CEAF_e$ computes an optimal entity alignment with respect to the relative overlap, and then normalizes by the number of entities. As the metrics are not link-based, they do not provide means to extract link-based errors. We leave determining whether the framework of these metrics exhibits a useful notion of errors to future work.

[10]These are linguistically agnostic since they do not differ between different mention or entity types when evaluating.

[11]via the transformation $x \mapsto 1 - x$

Recent work considered devising evaluation metrics which take linguistic information into account. Chen and Ng (2013) inject linguistic knowledge into existing evaluation metrics by weighting links in an entity representation graph. Tuggener (2014) devises scoring algorithms tailored for particular applications by redefining the notion of a correct link. While both of these works focus on scoring, they weight or explicitly define links in the reference and system entities, thereby they in principle allow error extraction. However, the authors do not attempt this and it is not clear whether the errors extracted that way are useful for analysis and system development.

7 Conclusions

We presented a novel link-based framework for coreference resolution error analysis, which extends and complements previous work. We applied the framework to analyze recall errors of four state-of-the-art systems on a large English benchmark dataset. Concentrating on errors involving only proper names and common nouns, we identified a core set of challenging errors common to all systems in our study.

We characterized the common errors among a broad range of properties. In particular, our analysis highlights and quantifies the usefulness of world knowledge. Furthermore, by comparing the recall errors made by each system, we identified individual strengths and weaknesses. A brief precision error analysis highlighted the hardness of resolving noun-name and noun-noun links.

The presented method and findings help to identify challenges in coreference resolution and to investigate ways to overcome these challenges.

Acknowledgments

This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author has been supported by a HITS Ph.D. scholarship.

References

Mira Ariel. 1988. Referring and accessibility. *Journal of Linguistics*, 24(1):65–87.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the 1st International Conference on Language Resources and Evaluation,* Granada, Spain, 28–30 May 1998, pages 563–566.

Mohit Bansal and Dan Klein. 2012. Coreference semantics from web features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Jeju Island, Korea, 8–14 July 2012, pages 389–398.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Waikiki, Honolulu, Hawaii, 25–27 October 2008, pages 294–303.

Anders Björkelund and Richárd Farkas. 2012. Data-driven multilingual coreference resolution using resolver stacking. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 49–55.

Jie Cai and Michael Strube. 2010. End-to-end coreference resolution via hypergraph partitioning. In *Proceedings of the 23rd International Conference on Computational Linguistics,* Beijing, China, 23–27 August 2010, pages 143–151.

Chen Chen and Vincent Ng. 2012. Combining the best of two worlds: A hybrid approach to multilingual coreference resolution. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 56–63.

Chen Chen and Vincent Ng. 2013. Linguistically aware coreference evaluation metrics. In *Proceedings of the 6th International Joint Conference on Natural Language Processing,* Nagoya, Japan, 14–18 October 2013, pages 1366–1374.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, Wash., 18–21 October 2013, pages 1971–1982.

Eraldo Fernandes, Cícero dos Santos, and Ruy Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 41–48.

Jonathan K. Kummerfeld and Dan Klein. 2013. Error-driven analysis of challenges in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, Wash., 18–21 October 2013, pages 265–277.

Shalom Lappin and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Human Language Technology Conference and the 2005 Conference on Empirical Methods in Natural Language Processing,* Vancouver, B.C., Canada, 6–8 October 2005, pages 25–32.

Sebastian Martschat, Jie Cai, Samuel Broscheit, Éva Mújdricza-Maydt, and Michael Strube. 2012. A multigraph model for coreference resolution. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 100–106.

Sebastian Martschat. 2013. Multigraph clustering for unsupervised coreference resolution. In *51st Annual Meeting of the Association for Computational Linguistics: Proceedings of the Student Research Workshop,* Sofia, Bulgaria, 5–7 August 2013, pages 81–88.

Alexis Mitchell, Stephanie Strassel, Shudong Huang, and Ramez Zakhary. 2004. ACE 2004 multilingual training corpus. LDC2005T09, Philadelphia, Penn.: Linguistic Data Consortium.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics,* Philadelphia, Penn., 7–12 July 2002, pages 104–111.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics,* Uppsala, Sweden, 11–16 July 2010, pages 1396–1411.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics,* New York, N.Y., 4–9 June 2006, pages 192–199.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. CoNLL-2011 Shared Task: Modeling unrestricted coreference in OntoNotes. In *Proceedings of the Shared Task of the 15th Conference on Computational Natural Language Learning,* Portland, Oreg., 23–24 June 2011, pages 1–27.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 Shared Task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings*

of the Shared Task of the 16th Conference on Computational Natural Language Learning, Jeju Island, Korea, 12–14 July 2012, pages 1–40.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* Baltimore, Md., 22–27 June 2014, pages 30–35.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing,* Cambridge, Mass., 9–11 October 2010, pages 492–501.

Altaf Rahman and Vincent Ng. 2011. Coreference resolution with world knowledge. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Portland, Oreg., 19–24 June 2011, pages 814–824.

Lev Ratinov and Dan Roth. 2012. Learning-based multi-sieve co-reference resolution with knowledge. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 1234–1244.

Don Tuggener. 2014. Coreference resolution evaluation for higher level applications. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics,* Gothenburg, Sweden, 26–30 April 2014, pages 231–235.

Olga Uryupina, Massimo Poesio, Claudio Giuliano, and Kateryna Tymoshenko. 2011. Disambiguation and filtering methods in using web knowledge for coreference resolution. In *Proceedings of the 24th International Florida Artificial Intelligence Research Society Conference,* Palm Beach, USA, 18–20 May 2011, pages 317–322.

Olga Uryupina. 2007. *Knowledge acquisition for coreference resolution*. Ph.D. thesis, Saarland University, Saarbrücken, Germany.

Olga Uryupina. 2008. Error analysis for learning-based coreference resolution. In *Proceedings of the 6th International Conference on Language Resources and Evaluation,* Marrakech, Morocco, 26 May – 1 June 2008, pages 1914–1919.

Renata Vieira and Massimo Poesio. 2000. An empirically-based system for processing definite descriptions. *Computational Linguistics*, 26(4):539–593.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Message Understanding Conference (MUC-6)*, pages 45–52, San Mateo, Cal. Morgan Kaufmann.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. OntoNotes release 5.0. LDC2013T19, Philadelphia, Penn.: Linguistic Data Consortium.

A Rule-Based System for Unrestricted Bridging Resolution: Recognizing Bridging Anaphora and Finding Links to Antecedents

Yufang Hou[1], **Katja Markert**[2], **Michael Strube**[1]
[1] Heidelberg Institute for Theoretical Studies gGmbH, Heidelberg, Germany
(yufang.hou|michael.strube)@h-its.org
[2]School of Computing, University of Leeds, UK
scskm@leeds.ac.uk

Abstract

Bridging resolution plays an important role in establishing (local) entity coherence. This paper proposes a rule-based approach for the challenging task of unrestricted bridging resolution, where bridging anaphors are not limited to definite NPs and semantic relations between anaphors and their antecedents are not restricted to meronymic relations. The system consists of eight rules which target different relations based on linguistic insights. Our rule-based system significantly outperforms a reimplementation of a previous rule-based system (Vieira and Poesio, 2000). Furthermore, it performs better than a learning-based approach which has access to the same knowledge resources as the rule-based system. Additionally, incorporating the rules and more features into the learning-based system yields a minor improvement over the rule-based system.

1 Introduction

Bridging resolution recovers the various non-identity relations between anaphora and antecedents. It plays an important role in establishing entity coherence in a text. In Example 1, the links between the bridging anaphors (**The five astronauts** and **touchdown**) and the antecedent (*The space shuttle Atlantis*) establish (local) entity coherence.[1]

(1) *The space shuttle Atlantis* landed at a desert air strip at Edwards Air Force Base, Calif., ending a five-day mission that dispatched the Jupiter-bound Galileo space probe. **The five astronauts** returned to Earth about three hours early because high winds had been predicted at the landing site. Fog shrouded the base before **touchdown**.

Bridging or *associative anaphora* has been widely discussed in the linguistic literature (Clark, 1975; Prince, 1981; Gundel et al., 1993; Löbner, 1998). Poesio and Vieira (1998) and Bunescu (2003) include cases where antecedent and anaphor are coreferent but do not share the same head noun (*different-head coreference*). We follow our previous work (Hou et al., 2013b) and restrict *bridging* to non-coreferential cases. We also exclude *comparative anaphora* (Modjeska et al., 2003).

Bridging resolution includes two subtasks: (1) recognizing bridging anaphors and (2) finding the correct antecedent among candidates. In recent empirical work, these two subtasks have been tackled separately: (Markert et al., 2012; Cahill and Riester, 2012; Rahman and Ng, 2012; Hou et al., 2013a) handle bridging recognition as part of information status (IS) classification, while (Poesio et al., 1997; Poesio et al., 2004; Markert et al., 2003; Lassalle and Denis, 2011; Hou et al., 2013b) concentrate on antecedent selection only, assuming that bridging recognition has already been performed. One exception is Vieira and Poesio (2000). They propose a rule-based system for processing definite NPs. However, they include *different-head coreference* into *bridging*. They report results for the whole anaphora resolution but do not report results for bridging resolution only. Another exception is Rösiger and Teufel (2014). They apply a coreference resolution system with several additional semantic features to find bridging links in scientific text where bridging anaphors are limited to definite NPs. They report preliminary results using the CoNLL scorer. However, we think the coreference resolution system and the evaluation metric for coreference resolution are

[1]Examples are from OntoNotes (Weischedel et al., 2011). Bridging anaphora are typed in boldface; antecedents in italics.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2082–2093,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

not suitable for bridging resolution since bridging is not a set problem.

Another vein of research for bridging resolution focuses on formal semantics. Asher and Lascarides (1998) and Cimiano (2006) model bridging by integrating discourse structure and semantics from a formal semantics viewpoint. However, the implementation of such a theoretical framework is beyond the current capabilities of NLP since it depends heavily on commonsense entailment.

In this paper, we propose a rule-based system for unrestricted bridging resolution. The system consists of eight rules which we carefully design based on linguistic intuitions, i.e., how the nature of bridging is reflected by various lexical, syntactic and semantic features. We evaluate our rule-based system on a corpus where bridging is reliably annotated. We find that our rule-based system significantly outperforms a reimplementation of a previous rule-based system (Vieira and Poesio, 2000). We further notice that our rule-based system performs better than a learning-based approach which has access to the same knowledge resources as the rule-based system. Surprisingly, incorporating the rules and more features into the learning-based approach only yields a minor improvement over the rule-based system. We observe that diverse bridging relations and relatively small-scale data for each type of relations make generalization difficult for the learning-based approach. This work is – to the best of our knowledge – the first system recognizing bridging anaphora and finding links to antecedents for unrestricted phenomenon where bridging anaphors are not limited to definite NPs and semantic relations between anaphors and their antecedents are not restricted to meronymic relations.

2 Data

All the data used throughout the paper come from the ISNotes corpus[2] released by Hou et al. (2013b). This corpus contains around 11,000 NPs annotated for information status including 663 bridging NPs and their antecedents in 50 texts taken from the WSJ portion of the OntoNotes corpus (Weischedel et al., 2011). ISNotes is reliably annotated for bridging: for bridging anaphor recognition, κ is over 60 for all three possible annotator pairings (κ is over 70 for two expert annotators); for selecting bridging antecedents, agreement is around 80% for all annotator pairings.

It is notable that bridging anaphors in ISNotes are not limited to definite NPs as in previous work (Poesio et al., 1997; Poesio et al., 2004; Lassalle and Denis, 2011). Table 1 shows the bridging anaphora distribution with regard to determiners in ISNotes: only around 38% of bridging anaphors are definite NPs (NPs modified by *the*); 15.4% of bridging anaphors are modified by determiners such as *a*, *an* or *one* which normally indicate indefinite NPs. Most bridging anaphors (43%) are not modified by any determiners, such as **touchdown** in Example 1. A small fraction of bridging anaphors (1.2%) are modified by other determiners, such as demonstratives.

Bridging Anaphors	663
Non-determiner	44.9%
Definite	38.5%
Indefinite	15.4%
Other-determiner	1.2%

Table 1: Bridging anaphora distribution w.r.t. determiners in ISNotes.

The semantic relations between anaphor and antecedent in the corpus are extremely diverse: only 14% of anaphors have a part-of/attribute-of relation with the antecedent (see Example 2) and only 7% of anaphors stand in a set relationship to the antecedent (see Example 3). 79% of anaphors have "other" relations with their antecedents (without further distinction), including encyclopedic relations such as *The space shuttle Atlantis*-**The five astronauts** (see Example 1) as well as context-specific relations such as *The space shuttle Atlantis*-**touchdown** (Example 1).

(2) At age eight, Josephine Baker was sent by her mother to *a white women's house* to do chores in exchange for meals and a place to sleep – a place in **the basement** with coal.

(3) This creates *several problems*. **One** is that there are not enough police to satisfy small businesses.

In ISNotes, bridging anaphora with distant antecedents are common when the antecedent is the global focus of a document. 29% of the anaphors in the corpus have antecedents that are three or more sentences away.

[2] http://www.h-its.org/english/research/nlp/download/isnotes.php

Bridging resolution is an extremely challenging task in ISNotes. In contrast with surface clues for coreference resolution, there are no clear surface clues for bridging resolution. In Example 4, the bridging anaphor **low-interest disaster loans** associates to the antecedent *the Carolinas and Caribbean*, whereas in Example 5 the NP loans is a generic use. In Example 6, the bridging anaphor **The opening show** associates to the antecedent *Mancuso FBI*, whereas the NP the show is coreferent with its antecedent *Mancuso FBI*.

(4) The $2.85 billion measure comes on top of $1.1 billion appropriated after Hugo stuck *the Carolinas and Caribbean* last month, and these totals don't reflect the additional benefit of **low-interest disaster loans**.

(5) Many states already have Enterprise Zones and legislation that combines tax incentives, loans, and grants to encourage investment in depressed areas.

(6) Over the first few weeks, *Mancuso FBI* has sprung straight from the headlines. **The opening show** featured a secretary of defense designate accused of womanizing (a la John Tower).
...
Most of all though, the show is redeemed by the character of Mancuso.

Our previous work on bridging resolution on this corpus only focuses on its subtasks. In Hou et al. (2013a) we model bridging anaphora recognition as a subtask of learning fine-grained information status. We report an F-measure of 0.42 for bridging anaphora recognition. In Hou et al. (2013b) we propose a joint inference framework for antecedent selection by exploring Markov logic networks. We report an accuracy of 0.41 for antecedent selection given gold bridging anaphora. In this paper, we aim to solve these two substasks together, i.e., recognizing bridging anaphora and finding links to antecedents.

3 Method

In this section, we describe our rule-based system for unrestricted bridging resolution. We choose ten documents randomly from the corpus as the development set. Then we carefully design rules for finding "bridging links" among all NPs in a document based on the generalizations of bridging in the linguistic literature as well as our inspections of bridging annotations in the development set. The system consists of two components: bridging link prediction and post processing.

3.1 Bridging Link Prediction

The bridging link prediction component consists of eight rules. Löbner (1985; 1998) interprets bridging anaphora as a particular kind of functional concept, which in a given situation assign a necessarily unique correlate to a (implicit) possessor argument. He distinguishes between relational nouns (e.g. parts terms, kinship terms, role terms) and sortal nouns and points out that relational nouns are more frequently used as bridging anaphora than sortal nouns. Rule1 to Rule4 in our system aim to resolve such relational nouns. We design Rule5 and Rule6 to capture set bridging. Finally, Rule7 and Rule8 are motivated by previous work on implicit semantic role labeling (Laparra and Rigau, 2013) which focuses on few predicates.

For all NPs in a document, each rule r is applied separately to predict a set of potential bridging links. Every rule has its own constraints on bridging anaphora and antecedents respectively. Bridging anaphors are diverse with regard to syntactic form and function: they can be modified by definite or indefinite determiners (Table 1), furthermore they can take the subject (e.g. Example 3 and Example 6) or other positions (e.g. Example 2 and Example 4) in sentences. The only frequent syntactic property shared is that bridging anaphors most often have a simple internal structure concerning modification. Therefore we first create an initial list of potential bridging anaphora A which excludes NPs which have a complex syntactic structure. An NP is added to A if it does not contain any other NPs and do not have modifications strongly indicating comparative NPs (such as *other symptoms*)[3]. Since head match is a strong indicator of coreference anaphora for definite NPs (Vieira and Poesio, 2000; Soon et al., 2001), we further exclude definite NPs from A if they have the same head as a previous NP. Then a set of potential bridging anaphors A_r is chosen from A based on r's constraints on bridging anaphora. Finally, for each potential bridging anaphor $ana \in$

[3] A small list of 10 markers such as *such, another* ... and the presence of adjectives or adverbs in the comparative form are used to predict comparative NPs.

A_r, a single best antecedent $ante$ from a list of candidate NPs (C_{ana}) is chosen if the rule's constraint on antecedents is applied successfully.

Every rule has its own scope to form the antecedent candidate set C_{ana}. Instead of using a static sentence window to construct the list of antecedent candidates like most previous work for resolving bridging anaphora (Poesio et al., 1997; Markert et al., 2003; Poesio et al., 2004; Lassalle and Denis, 2011), we use the development set to estimate the proper scope for each rule. The scope is influenced by the following factors: (1) the nature of the target bridging link (e.g., set bridging is a local coherence phenomenon where the antecedent often occurs in the same or up to two sentences prior to the anaphor); and (2) the strength of the rule's constraint to select the correct antecedent (e.g., in Rule8, the ability to select the correct antecedent decreases with increasing the scope to contain more antecedent candidates). In the following, we describe the motivation for each rule and their constraints in detail.

Rule1: building part NPs. To capture typical part-of bridging (see Example 2), we extract a list of 45 nouns which specify building parts (e.g. *room* or *roof*) from the General Inquirer lexicon (Stone et al., 1966). A common noun phrase from A is added to A_{r1} if: (1) its head appears in the building part list; and (2) it does not contain any nominal pre-modifications. Then for each potential bridging anaphor $ana \in A_{r1}$, the NP with the strongest semantic connectivity to the potential anaphor ana among all NPs preceding ana from the same sentence as well as from the previous two sentences is predicted to be the antecedent.

The semantic connectivity of an NP to a potential anaphor is measured via the hit counts of the preposition pattern query (*anaphor preposition NP*) in big corpora[4]. An initial effort to extract partOf relations using WordNet yields low recall on the development set. Therefore we use semantic connectivity expressed by prepositional patterns (e.g. *the basement of the house*) to capture underlying semantic relations. Such syntactic patterns are also explored in Poesio et al. (2004) to resolve meronymy bridging.

[4]We use Gigaword (Parker et al., 2011) with automatic POS tag and NP chunk information.

Rule2: relative person NPs. This rule is used to capture the bridging relation between a relative (e.g. *The husband*) and its antecedent (e.g. *She*). A list of 110 such relative nouns is extracted from WordNet. However, some relative nouns are frequently used generically instead of being bridging, such as *children*. To exclude such cases, we compute the argument taking ratio α for an NP using NomBank (Meyers et al., 2004). For each NP, α is calculated via its head frequency in the NomBank annotation divided by the head's total frequency in the WSJ corpus in which the NomBank annotation is conducted. The value of α reflects how likely an NP is to take arguments. For instance, the value of α is 0.90 for *husband* but 0.31 for *children*. To predict bridging anaphora more accurately, a conservative constraint is used. An NP from A is added to A_{r2} if: (1) its head appears in the relative person list; (2) its argument taking ratio α is bigger than 0.5; and (3) it does not contain any nominal or adjective pre-modifications. Then for each potential bridging anaphor $ana \in A_{r2}$, the closest non-relative person NP among all NPs preceding ana from the same sentence as well as from the previous two sentences is chosen as its antecedent.

Rule3: GPE job title NPs. In news articles, it is common that a globally salient geo-political entity (hence GPE, e.g. *Japan* or *U.S.*) is introduced in the beginning, then later a related job title NP (e.g. *officials* or *the prime minister*) is used directly without referring to this GPE explicitly. To resolve such bridging cases accurately, we compile a list of twelve job titles which are related to GPEs (e.g. *mayor* or *official*). An NP from A is added to A_{r3} if its head appears in this list and does not have a country pre-modification (e.g. *the Egyptian president*). Then for each potential bridging anaphor $ana \in A_{r3}$, the most salient GPE NP among all NPs preceding ana is predicted as its antecedent. We use the NP's frequency in the whole document to measure its salience throughout the paper. In case of a tie, the closest one is chosen to be the predicted antecedent.

Rule4: role NPs. Compared to Rule3, Rule4 is designed to resolve more general role NPs to their implicit possessor arguments. We extract a list containing around 100 nouns which specify professional roles from WordNet (e.g. *chairman*, *president* or *professor*). An NP from A is added to

A_{r4} if its head appears in this list. Then for each potential bridging anaphor $ana \in A_{r4}$, the most salient proper name NP which stands for an organization among all NPs preceding ana from the same sentence as well as from the previous four sentences is chosen as its antecedent (if such an NP exists). Recency is again used to break ties.

Rule5: percentage NPs. In set bridging as shown in Example 7, the anaphor (**Seventeen percent**) is indicated by a percentage expression from A, which is often in the subject position. The antecedent (*the firms*) is predicted to be the closest NP which modifies another percentage NP via the preposition "of" among all NPs occurring in the same or up to two sentences prior to the potential anaphor.

(7) 22% of *the firms* said employees or owners had been robbed on their way to or from work. **Seventeen percent** reported their customers being robbed.

Rule6: other set member NPs. In set bridging, apart from percentage expressions, numbers or indefinite pronouns are also good indicators for bridging anaphora. For such cases, the anaphor is predicted if it is: (1) a number expression (e.g. **One** in Example 3) or an indefinite pronoun(e.g. **some**, as shown in Example 8) from A; and (2) a subject NP. The antecedent is predicted to be the closest NP among all plural, subject NPs preceding the potential anaphor from the same sentence as well as from the previous two sentences (e.g. *Reds and yellows* in Example 8). If such an NP does not exist, the closest NP among all plural, object NPs preceding the potential anaphor from the same sentence as well as from the previous two sentences is chosen to be the predicted antecedent (e.g. *several problems* in Example 3).

(8) *Reds and yellows* went about their business with a kind of measured grimness. **Some** frantically dumped belongings into pillowcases.

Rule7: argument-taking NPs I. Laparra and Rigau (2013) found that different instances of the same predicate in a document likely maintain the same argument fillers. Here we follow this assumption but apply it to nouns and their nominal modifiers only: different instances of the same noun predicate likely maintain the same argument fillers indicated by nominal modifiers. First, a common noun phrase from A is added to A_{r7} if: (1) its argument taking ratio α is bigger than 0.5; (2) it does not contain any nominal or adjective pre-modifications; and (3) it is not modified by indefinite determiners[5] which usually introduce new discourse referents (Hawkins, 1978). Then for each potential bridging anaphor $ana \in A_{r7}$, we choose the antecedent by performing the following steps:

1. We take ana's head lemma form ana_h and collect all its syntactic modifications in the document. We consider nominal premodification, possessive modification as well as prepositional post-modification. All realizations of these modifications which precede ana form the antecedent candidates set C_{ana}.

2. We choose the most recent NP from C_{ana} as the predicted antecedent for the potential bridging anaphor ana.

In Example 9, we first predict the two occurrences of **residents** as bridging anaphors. Since in the text, other occurrences of the lemma "resident" are modified by "Marina" (supported by Marina residents) and "buildings" (supported by some residents of badly damaged buildings), we collect all NPs whose syntactic head is "Marina" or "buildings" in C_{ana} (i.e. *Marina*, *badly damaged buildings* and *buildings with substantial damage*). Then among all NPs in C_{ana}, the most recent NP is chosen to be the antecedent (i.e. *buildings with substantial damage*).

(9) She finds the response of Marina residents to the devastation of their homes "incredible".
...
Out on the streets, some residents of badly damaged buildings were allowed a 15 minute scavenger hunt through their possessions.
...
After being inspected, *buildings with substantial damage* were color - coded.
Green allowed **residents** to re-enter; red allowed **residents** one last entry to gather everything they could within 15 minutes.

Rule8: argument-taking NPs II. Prince (1992) found that discourse-old entities are more likely

[5]We compile a list of 17 such determiners, such as *a*, *an* or *one*.

to be represented by NPs in subject position. Although she could not draw a similar conclusion when collapsing *Inferrable* (= *bridging*) with *Discourse-old Nonpronominal*, we find that in the development set, an argument-taking NP in the subject position is a good indicator for bridging anaphora (e.g. **participants** in Example 10). A common noun phrase from A is collected in A_{r8} if: (1) its argument taking ratio α is bigger than 0.5; (2) it does not contain any nominal or adjective pre-modifications; and (3) it is in the subject position. Semantic connectivity again is used as the criteria to choose the antecedent: for each potential bridging anaphor $ana \in A_{r8}$, the NP with the strongest semantic connectivity to ana among all NPs preceding ana from the same sentence as well as from the previous two sentences is predicted to be the antecedent.

(10) Initial steps were taken at *Poland's first international environmental conference, which I attended last month*. ... While Polish data have been freely available since 1980, it was no accident that **participants** urged the free flow of information.

3.2 Post-processing

In the bridging link prediction component, each rule is applied separately. To resolve the conflicts between different rules (e.g., two rules predict different antecedents for the same potential anaphor), a post processing step is applied. We first order the rules according to their precision for predicting bridging pairs (i.e., recognizing bridging anaphors and finding links to antecedents) in the development set. When a conflict happens, the rule with the highest order has the priority to decide the antecedent. Table 2 summarizes the rules described in Section 3.1, the numbers in square brackets in the first column indicate the order of the rules. Table 3 shows the precisions of bridging anaphora recognition and bridging pairs prediction for each rule in the development set. Firing rate is the proportion of bridging links predicted by rule r among all predicted links.

4 Experiments and Results

4.1 Experimental Setup

We conduct all experiments on the ISNotes corpus. We use the OntoNotes named entity and syntactic annotations to extract features. Ten documents containing 113 bridging anaphors from the ISNotes corpus are set as the development set to estimate parameters for the rule-based system. The remaining 40 documents are used as the test set. In order to compare the results of different systems directly, we evaluate all systems on the test set.

4.2 Evaluation Metric

In ISNotes, bridging is annotated mostly between an NP (anaphor) and an entity (antecedent)[6], so that a bridging anaphor could have multiple links to different instantiations of the same entity (entity information is based on the Ontonotes coreference annotation). For bridging resolution, we use an evaluation metric based on bridging anaphors instead of all links between bridging anaphors and their antecedent instantiations. A link predicted by the system is counted as correct if it recognizes the bridging anaphor correctly and links the anaphor to any instantiation of the right antecedent entity preceding the anaphor.

In the evaluation metric, recall is calculated via the number of the correct links predicted by the system (one unique link per each predicted anaphor) divided by the total number of the gold bridging anaphors, precision is calculated via the number of the correct links predicted by the system divided by the total links predicted by the system.

4.3 A Learning-based Approach

To compare our rule-based system (hence *ruleSystem*, described in Section 3) with other approaches, we implement a learning-based system for unrestricted bridging resolution. We adapt the pairwise model which is widely used in coreference resolution (Soon et al., 2001). Similar to the rule-based system, we first create an initial list of possible bridging anaphora A_{ml} with one more constraint. The purpose is to exclude as many obvious non-bridging anaphoric NPs from the list as possible. An NP is added to A_{ml} if: (1) it does not contain any other NPs; (2) it is not modified by pre-modifications which strongly indicate comparative NPs; and (3) it is not a pronoun or a proper name. Then for each NP $a \in A_{ml}$, a list of antecedent candidates C_a is created by including all NPs preceding a from the same sentence

[6]There are a few cases where bridging is annotated between an NP and a non-NP antecedent (e.g. *verbs* or *clauses*).

rule	anaphor	antecedent	antecedent candidates scope
rule1 [2]	building part NPs	the NP with the strongest semantic connectivity to the potential anaphor	two
rule2 [5]	relative person NPs	the closest person NP which is not a relative NP	two
rule3 [6]	GPE job title NPs	the most salient GPE NP	all
rule4 [7]	role NPs	the most salient organization NP	four
rule5 [1]	percentage NPs	the closest NP which modifies another percentage NP via the preposition "of"	two
rule6 [3]	other set member NPs	the closest subject, plural NP; otherwise the closest object, plural NP	two
rule7 [4]	argument-taking NPs I	the closest NP whose head is an unfilled role of the potential anaphor (such a role is predicted via syntactic modifications of NPs which have the same head as the potential anaphor)	all
rule8 [8]	argument-taking NPs II	the NP with the strongest semantic connectivity to the potential anaphor	two

Table 2: Rules for unrestricted bridging resolution. Antecedent candidates scope are verified in the development set: "all" represents all NPs preceding the potential anaphor from the whole document, "four" NPs occurring in the same or up to four sentences prior to the potential anaphor, "two" NPs occurring in the same or up to two sentences prior to the potential anaphor.

rule	anaphora	anaphora recognition precision	bridging pairs prediction precision	firing rate
rule1 [2]	building part NPs	75.0%	50.0%	6.1%
rule2 [5]	relative person NPs	69.2%	46.2%	6.1%
rule3 [6]	GPE job title NPs	52.6%	44.7%	19.4%
rule4 [7]	role NPs	61.7%	32.1%	28.6%
rule5 [1]	percentage NPs	100.0%	100.0%	2.6%
rule6 [3]	other set member NPs	66.7%	46.7%	7.8%
rule7 [4]	argument-taking NPs I	53.8%	46.4%	6.1%
rule8 [8]	argument-taking NPs II	64.5%	25.0%	25.5%

Table 3: Precision of bridging anaphora recognition and bridging pairs prediction for each rule in the development set. The numbers in square brackets in the first column indicate the order of the rules.

as well as from the previous two sentences[7]. We create a pairwise instance (a, c) for every $c \in C_a$. We also add extra pairwise instances from the prediction of *ruleSystem* to the learning-based system. In the decoding stage, the *best first* strategy (Ng and Cardie, 2002) is used to predict the bridging links. Specifically, for each $a \in A_{ml}$, we predict the bridging link to be the most confident pair (a, c_{ante}) among all instances with the positive prediction. We use SVM^{light} to conduct the experiments[8]. All experiments are conducted via 10-fold cross-validation on the whole corpus[9].

mlSystem_ruleFeats We provide *mlSystem_ruleFeats* with the same knowledge resources as the rule-based system. All rules from the rule-based system are incorporated into *mlSystem_ruleFeats* as the features.

mlSystem_ruleFeats + atomFeats We augment *mlSystem_ruleFeats* with more features from our previous work (Markert et al., 2012; Hou et al., 2013a; Hou et al., 2013b) on bridging anaphora recognition and antecedent selection. Some of these features overlap with the atomic features used in the rule-based system.

Table 4 shows all the features we use for recognizing bridging anaphora. "$*$" indicates the resources are used in the rule-based system. We apply them to the first element a of a pairwise instance (a, c). Markert et al. (2012) and Hou et

[7]In ISNotes, 71% of NP antecedents occur in the same or up to two sentences prior to the anaphor. Initial experiments show that increasing the window size more than two sentences decreases the performance.

[8]To deal with data imbalance, the SVM^{light} parameter is set according to the ratio between positive and negative instances in the training set.

[9]To compare the learning-based approach to the rule-based system described in Section 3 directly, we report the results of learning-based approaches on the same test set as the rule-based system.

Markert et al. local feature set		
$f1$ FullPrevMention (b) $*$	$f2$ FullPreMentionTime (n)	$f3$ PartialPreMention (b)
$f4$ ContentWordPreMention (b)	$f5$ Determiner (n) $*$	$f6$ NPtype (n) $*$
$f7$ NPlength (int)	$f8$ GrammaticalRole (n) $*$	$f9$ NPNumber (n) $*$
$f10$ PreModByCompMarker (b) $*$		
Hou et al. local feature set		
features to identify bridging anaphora		
$f1$ IsCoherenceGap (b)	$f2$ IsSentFirstMention (b)	$f3$ IsDocFirstMention (b)
$f4$ IsWordNetRelationalNoun (b) $*$	$f5$ IsInquirerRoleNoun (b)	$f6$ IsBuildingPart (b) $*$
$f7$ IsSetElement (b) $*$	$f8$ PreModSpatialTemporal (b)	$f9$ IsYearExpression (b)
$f10$ PreModifiedByCountry (b) $*$	$f11$ AppearInIfClause (b)	$f12$ VerbPosTag (l)
$f13$ IsFrequentGenericNP (b)	$f14$ WorldKnowledgeNP (l)	$f15$ Unigrams (l)
$f16$ PreModByGeneralQuantifier (b)	$f17$ BridgingHeadNP (l)	$f18$ HasChildNP (b) $*$
features to identify function and worldKnowledge NPs		
$f20$ DependOnChangeVerb (b)	$f21$ IsFrequentProperName (b)	

Table 4: Features for bridging anaphora recognition from Markert et al. (2012) and Hou et al. (2013a). "b" indicates binary, "n" nominal, "l" lexical features, "$*$" resources used in the rule-based system.

Group	Feature	Value
semantic	$f1$ preposition pattern $*$	the normalized hit counts of the preposition pattern query *a prep. c* (e.g. *participants of the conference*) in big corpora
	$f2$ verb pattern	the normalized hit counts of the verb pattern query *c* $verb_a$ or $verb_a$ *c* in big corpora (for set bridging in Example 7, the pattern query is *the firms reported*)
	$f3$ WordNet partOf	whether a partOf relation holds between a and c in WordNet
	$f4$ semantic class $*$	16 classes, e.g. location, organization, GPE, rolePerson, relativePerson, product, date, money, percent
salience	$f5$ document span	the normalized value of the span of text in which c is mentioned
	$f6$ utterance distance	the sentence distance between a and c
	$f7$ local first mention	whether c is the first mention within the previous five sentences
	$f8$ global first mention	whether c is the first mention in the whole document
syntactic & lexical	$f9$ isSameHead	whether a and c share the same head (exclude coreferent antecedent candidates)
	$f10$ isWordOverlap	whether a is prenominally modified by the head of c (for bridging where the anaphor is a compound noun, such as *the mine*-**mine security**)
	$f11$ isCoArgument	whether subject c and object a are dependent on the same verb (the subject can not be the bridging antecedent of the object in the same clause)
	$f12$ WordNet distance	the inverse value of the shortest path length between a and c in WordNet

Table 5: Features for antecedent selection from Hou et al. (2013b). "$*$" indicates resources used in the rule-based system.

al. (2013a) classify eight fine-grained information status (IS) categories for NPs: *old*, *new* and 6 *mediated* categories (*syntactic*, *worldKnowledge*, *bridging*, *comparative*, *aggregate* and *function*). Features from Markert et al. (2012) work well to identify *old*, *new* and several *mediated* categories but fail to recognize most bridging anaphora. Hou et al. (2013a) remedy this by adding discourse structure features ($f1$-$f3$), semantic features ($f4$-$f10$) and features to detect generic nouns ($f11$-

Feature	Value
for anaphor candidate a	
$f1$ preModByNominal	whether a contains any nominal pre-modifications
$f2$ preModByAdj	whether a contains any adjective modifications
$f3$ isGPEJobTitle	whether a is a job title about GPE (e.g. *mayor* or *official*)
$f4$ isArgumentTakingNP	whether the argument taking ratio of a is bigger than 0.5
for antecedent candidate c	
$f5$ fullMentionTime	the normalized value of the frequency of c in the whole document
for pairwise instance (a, c)	
$f6$ word distance	the token distance between a and c

Table 6: Additional atomic features from the rule-based system.

$f14$ and $f16$).

Table 5 shows all features we use for selecting antecedents for bridging anaphora. "$*$" indicates the resources that are used in the rule-based system. These features are from Hou et al. (2013b)'s local pairwise model. They try to model: (1) the semantic relations between bridging anaphors and their antecedents ($f1$ to $f4$); (2) the salience of an antecedent from different perspectives ($f5$ to $f8$); and (3) the syntactic and lexical constraints between anaphor and antecedent ($f9$ to $f12$).

Apart from the features shown in Table 4 and Table 5, we further enrich *mlSystem_ruleFeats+atomFeats* with additional atomic features used in the rule-based system (Table 6).

mlSystem_atomFeats Based on *mlSystem_ruleFeats+atomFeats*, the rule features from the rule-based system are removed.

4.4 Baseline

We also reimplement the rule-based system from Vieira and Poesio (2000) as a baseline. The original algorithm focuses on processing definite NPs. It classifies four categories for the definite NPs: *discourse new*, *direct anaphora* (same-head coreferent anaphora), *lenient bridging* and *Unknown*. This algorithm also finds antecedents for NPs which belong to *direct anaphora* or *lenient bridging*.

Since Vieira and Poesio (2000) include *different-head coreference* into their *lenient bridging* category, we further divide their *lenient bridging* category into two subcategories: *different-head coreference* and *bridging*. Figure 1 shows the details of the division after failing to classify an NP as *discourse new* or *direct anaphora*. For more details about the whole system, see Vieira and Poesio (2000). We then apply this slightly revised algorithm to process all NPs in the initial list of potential bridging anaphora A from *ruleSystem* (described in Section 3.1).

4.5 Results and Discussion

Table 7 shows the results on the same test set of different approaches for unrestricted bridging resolution. The results reveal the difficulty of the task, when evaluating on a realistic scenario without constraints on types of bridging anaphora and bridging relations.

Both our rule-based system and all learning-based approaches significantly outperform the baseline at $p < 0.01$ (randomization test). The low recall in *baseline* is predictable, since it only considers meronymy bridging and compound noun anaphors whose head is prenominally modified by the antecedent head. (e.g. *the state–***state gasoline taxes**). Under the same features, the learning-based approach (*mlSystem_ruleFeats*) performs slightly worse than the rule-based system (*ruleSystem*) with regard to the F-score.

	R	P	F
baseline	2.9	13.3	4.8
ruleSystem	11.9	42.9	**18.6**
mlSystem_ruleFeats	12.1	35.0	18.0
mlSystem_ruleFeats+atomFeats	16.7	21.2	**18.7**
mlSystem_atomFeats	20.5	10.1	13.5

Table 7: Experimental results for the baseline, the rule-based system and the learning-based systems.

Surprisingly, incorporating rich features into the learning-based approach (*mlSystem_ruleFeats+atomFeats*) does not yield much improvement over the rule-based system (with an

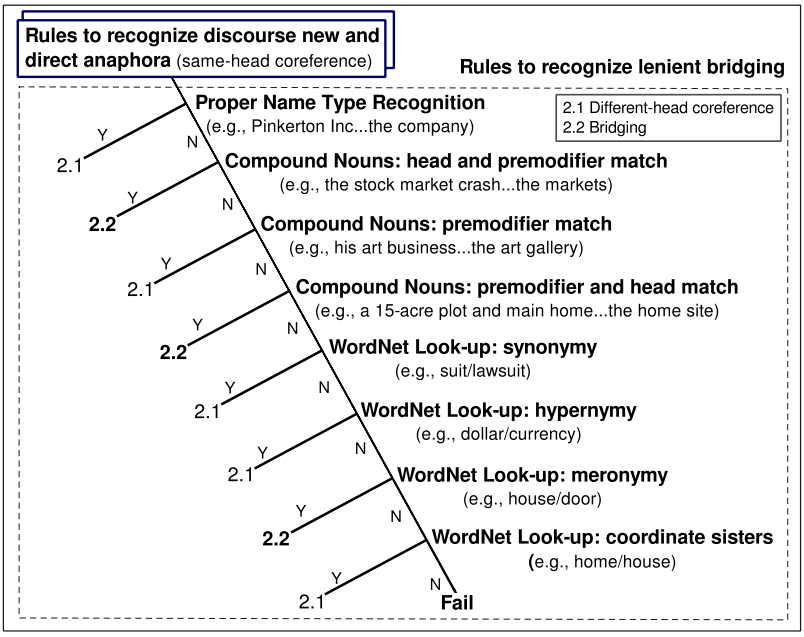


Figure 1: Vieria & Poesio's (2000) original algorithm for processing definite NPs. We further divide their lenient bridging category into two subcategories: *2.1 Different-head coreference* and *2.2 Bridging*.

F-score of 18.7 in *mlSystem_ruleFeats+atomFeats* compared to 18.6 in *ruleSystem*). We suppose that the learning-based system generalizes poorly with only atomic features in Table 4, Table 5 and Table 6. Results on *mlSystem_atomFeats* support our assumption: the F-score drops considerably after removing the rule features. Although ISNotes is a reasonably sized corpus for bridging compared to previous work, diverse bridging relations, especially lots of context specific relations such as *pachinko*-**devotees** or *palms*-**the thieves**, lead to relatively small-scale training data for each type of relation. Therefore it is difficult for the learning-based approach to learn effective rules to predict bridging links.

However, all learning-based systems tend to have higher recall but lower precision compared to the rule-based system. This suggests that the learning-based systems are "greedy" to predict bridging links. A close look at these links in *mlSystem_atomFeats* indicates that the learning-based system predicts more correct bridging anaphors but fails to find the correct antecedents. In fact, lots of those "half" correct links sound reasonable without the specific context, such as *the story-readers* (gold bridging link: *this novel*-**readers**) or *the executive director's office-the desks* (gold bridging link: *the fund's building*-**the desks**).

5 Conclusions

We proposed a rule-based approach for unrestricted bridging resolution where bridging anaphora are not limited to definite NPs and the relations between anaphor and antecedent are not restricted to meronymic relations. We designed eight rules to resolve bridging based on linguistic intuition. Our rule-based system performs better than a learning-based approach which has access to the same knowledge resources as the rule-based system. Particularly, the learning-based system enriched with more features does not yield much improvement over the rule-based system. We speculate that the learning-based system could benefit from more training data. Furthermore, better methods to model the semantics of the specific context need to be explored in the future.

This work is – to our knowledge – the first bridging resolution system that handles the unrestricted phenomenon in a realistic setting.

Acknowledgements

We thank Renata Vieira for excavating part of her source code for us. We also thank the reviewers for their helpful comments. Yufang Hou is funded by a PhD scholarship from the Research Training Group *Coherence in Language Processing* at Heidelberg University. This work has been partially funded by the Klaus Tschira Foundation.

References

Nicholas Asher and Alex Lascarides. 1998. Bridging. *Journal of Semantics*, 15:83–113.

Razvan Bunescu. 2003. Associative anaphora resolution: A Web-based approach. In *Proceedings of the EACL 2003 Workshop on The Computational Treatment of Anaphora,* Budapest, Hungary, 14 April, 2003, pages 47–52.

Aoife Cahill and Arndt Riester. 2012. Automatically acquiring fine-grained information status distinctions in German. In *Proceedings of the SIGdial 2012 Conference: The 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue,* Seoul, Korea, 5–6 July 2012, pages 232–236.

Philipp Cimiano. 2006. Ingredients of a first-order account of bridging. In *Proceedings of the 5th International Workshop on Inference in Computational Semantics,* Buxton, U.K., 20–21 April 2006, pages 139–144.

Herbert H. Clark. 1975. Bridging. In *Proceedings of the Conference on Theoretical Issues in Natural Language Processing,* Cambridge, Mass., June 1975, pages 169–174.

Jeanette K. Gundel, Nancy Hedberg, and Ron Zacharski. 1993. Cognitive status and the form of referring expressions in discourse. *Language*, 69:274–307.

John A. Hawkins. 1978. *Definiteness and indefiniteness: A study in reference and grammaticality prediction*. Humanities Press, Atlantic Highlands, N.J.

Yufang Hou, Katja Markert, and Michael Strube. 2013a. Cascading collective classification for bridging anaphora recognition using a rich linguistic feature set. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, Wash., 18–21 October 2013, pages 814–820.

Yufang Hou, Katja Markert, and Michael Strube. 2013b. Global inference for bridging anaphora resolution. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Atlanta, Georgia, 9–14 June 2013, pages 907–917.

Egoitz Laparra and German Rigau. 2013. ImpAr: A deterministic algorithm for implicit semantic role labelling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics,* Sofia, Bulgaria, 4–9 August 2013, pages 1180–1189.

Emmanuel Lassalle and Pascal Denis. 2011. Leveraging different meronym discovery methods for bridging resolution in French. In *Proceedings of the 8th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 2011),* Faro, Algarve, Portugal, 6–7 October 2011, pages 35–46.

Sebastian Löbner. 1985. Definites. *Journal of Semantics*, 4:279–326.

Sebastian Löbner. 1998. Definite associative anaphora. Unpublished Manuscript, Heinrich-Heine-Universität Düsseldorf.

Katja Markert, Malvina Nissim, and Natalia N. Modjeska. 2003. Using the web for nominal anaphora resolution. In *Proceedings of the EACL Workshop on the Computational Treatment of Anaphora.* Budapest, Hungary, 14 April 2003, pages 39–46.

Katja Markert, Yufang Hou, and Michael Strube. 2012. Collective classification for fine-grained information status. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics,* Jeju Island, Korea, 8–14 July 2012, pages 795–804.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishaman. 2004. Annotating noun argument structure for NomBank. In *Proceedings of the 4th International Conference on Language Resources and Evaluation,* Lisbon, Portugal, 26–28 May 2004, pages 803–806.

Natalia M. Modjeska, Katja Markert, and Malvina Nissim. 2003. Using the web in machine learning for other-anaphora resolution. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing,* Sapporo, Japan, 11–12 July 2003, pages 176–183.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics,* Philadelphia, Penn., 7–12 July 2002, pages 104–111.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition. LDC2011T07.

Massimo Poesio and Renata Vieira. 1998. A corpus-based investigation of definite description use. *Computational Linguistics*, 24(2):183–216.

Massimo Poesio, Renata Vieira, and Simone Teufel. 1997. Resolving bridging references in unrestricted text. In *Proceedings of the ACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Text, Madrid, Spain, July 1997*, pages 1–6.

Massimo Poesio, Rahul Mehta, Axel Maroudas, and Janet Hitzeman. 2004. Learning to resolve bridging references. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics,* Barcelona, Spain, 21–26 July 2004, pages 143–150.

Ellen F. Prince. 1981. Towards a taxonomy of given-new information. In P. Cole, editor, *Radical Pragmatics*, pages 223–255. Academic Press, New York, N.Y.

Ellen F. Prince. 1992. The ZPG letter: Subjects, definiteness, and information-status. In W.C. Mann and S.A. Thompson, editors, *Discourse Description. Diverse Linguistic Analyses of a Fund-Raising Text*, pages 295–325. John Benjamins, Amsterdam.

Altaf Rahman and Vincent Ng. 2012. Learning the fine-grained information status of discourse entities. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics,* Avignon, France, 23–27 April 2012, pages 798–807.

Ina Rösiger and Simone Teufel. 2014. Resolving coreference and associative noun phrases in scientific text. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics,* Gothenburg, Sweden, 26–30 April 2014, pages 44–55.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, Daniel M. Ogilvie, and Cambridge Computer Associates. 1966. *General Inquirer: A Computer Approach to Content Analysis*. MIT Press, Cambridge, Mass.

Renata Vieira and Massimo Poesio. 2000. An empirically-based system for processing definite descriptions. *Computational Linguistics*, 26(4):539–593.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2011. OntoNotes release 4.0. LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium.

Resolving Referring Expressions in Conversational Dialogs for Natural User Interfaces

Asli Celikyilmaz, Zhaleh Feizollahi, Dilek Hakkani-Tur, Ruhi Sarikaya
Microsoft
asli@ieee.org, zhalehf@microsoft.com
dilek@ieee.org, ruhi.sarikaya@microsoft.com

Abstract

Unlike traditional over-the-phone spoken dialog systems (SDSs), modern dialog systems tend to have visual rendering on the device screen as an additional modality to communicate the system's response to the user. Visual display of the system's response not only changes human behavior when interacting with devices, but also creates new research areas in SDSs. On-screen item identification and resolution in utterances is one critical problem to achieve a natural and accurate human-machine communication. We pose the problem as a classification task to correctly identify intended on-screen item(s) from user utterances. Using syntactic, semantic as well as context features from the display screen, our model can resolve different types of referring expressions with up to 90% accuracy. In the experiments we also show that the proposed model is robust to domain and screen layout changes.

1 Introduction

Todays natural user interfaces (NUI) for applications running on smart devices, e.g, phones (SIRI, Cortana, GoogleNow), consoles (Amazon FireTV, XBOX), tablet, etc., can handle not only simple spoken commands, but also natural conversational utterances. Unlike traditional over-the-phone spoken dialog systems (SDSs), user hears and sees the system's response displayed on the screen as an additional modality. Having visual access to the system's response and results changes human behavior when interacting with the machine, creating new and challenging problems in SDS.

[System]: How can i help you today ?
[User]: *Find non-fiction books by Chomsky.*
[System]: (Fetches the following books from database)

[User]: ***"show details for the oldest production"*** or
"details for the syntax book" or
"open the last one" or
"i want to see the one on linguistics" or
"bring me Jurafsky's text book"

Table 1: A sample multi-turn dialog. A list of second turn utterances referring to the last book (in bold) and a new search query (highlighted) are shown.

Consider a sample dialog in Table 1 between a user and a NUI in the books domain. After the system displays results on the screen, the user may choose one or more of the on-screen items with natural language utterances as shown in Table 1. Note that, there are multiple ways of referring to the same item, (e.g. the last book)[1]. To achieve a natural and accurate human to machine conversation, it is crucial to accurately identify and resolve referring expressions in utterances. As important as interpreting referring expressions (REs) is for modern NUI designs, relatively few studies have investigated withing the SDSs. Those that do focus on the impact of the input from multimodal interfaces such as gesture for understanding (Bolt, 1980; Heck et al., 2013; Johnston et al., 2002), touch for ASR error correction (Huggins-Daines and Rudnicky, 2008), or cues from the screen (Balchandran et al., 2008; Anastasiou et al., 2012). Most of these systems are engineered for a specific

[1] An item could be anything from a list, e.g. restaurants, games, contact list, organized in different lay-outs on the screen.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2094–2104,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

task, making it harder to generalize for different domains or SDSs. In this paper, we investigate a rather generic contextual model for resolving natural language REs for on-screen item selection to improve conversational understanding.

Our model, which we call FIS (Flexible Item Selection), is able to (1) detect if the user is referring to any item(s) on the screen, and (2) resolve REs to identify which items are referred to and score each item. FIS is a learning based system that uses information from pair of user utterance and candidate items on the screen to model association between them. We cast the task as a classification problem to determine whether there is a relation between the utterance and the item, representing each instance in the training dataset as relational features.

In a typical SDS, the spoken language understanding (SLU) engine maps user utterances into meaning representation by identifying user's intent and token level semantic slots via a semantic parser (Mori et al., 2008). The dialog manager uses the SLU components to decide on the correct system action. For on-screen item selection SLU alone may not be sufficient. To correctly associate the user's utterance with any of the on-screen items one would need to resolve the relational information between the utterance and the items. For instance, consider the dialog in Table 1. SLU engine can provide signals to the dialog model about the selected item, e.g., that "*linguistics*" is a book-genre or content, but may not be enough to indicate which book the user is referring. FIS module provides additional information for the dialog manager by augmenting SLU components.

In §3, we provide details about our data as well as data collection and annotation steps. In §4, we present various syntactic and semantic features to resolve different REs in utterances. In the experiments (§6), we evaluate the individual impact of each feature on the FIS model. We analyze the performance of the FIS model per each type of REs. Finally, we empirically investigate the robustness of the FIS model to domain and display screen changes. When tested on a domain that is unseen to the training data or on a device that has a different NUI design, the performance only slightly degrades proving its robustness to domain and design changes.

2 Related Work

Although the problems of modern NUIs on smart devices are fairly new, RE resolution in natural language has been studied by many in NLP community.

Multimodal systems provide a natural and effective way for users to interact with computers through multiple modalities such as speech, gesture, and gaze. Since the first appearance of the Put-That-There system (Bolt, 1980), a number of multimodal systems have been built, among which there are systems that combine speech, pointing (Neal, 1991), and gaze (Koons et al., 1993), systems that engage users in an intelligent conversation (Gustafson et al., 2000). Earlier studies have shown that multimodal interfaces enable users to interact with computers naturally and effectively (Schober and Clark, 1989; Oviatt et al., 1997). Considered as part of the situated interactive frameworks, many work focus on the problem of predicting how the user has resolved REs that is generated by the system, e.g., (Clark and Wilkes-Gibbs, ; Dale and Viethen, 2009; Gieselmann, 2004; Janarthanam and Lemon, 2010; Golland et al., 2010). In this work, focusing on smart devices, we investigate how the system resolves the REs in user utterances to take the next correct action.

In (Pfleger and J.Alexandersson, 2006) a reference resolution model is presented for a question-answering system on a mobile, multi-modal interface. Their system has several features to parse the posed question and keep history of the dialog to resolve co-reference issues. Their question-answering model uses gesture as features to resolve queries such as "*what's the name of that [pointing gesture] player*?", but they do not resolve locational referrals such as "*the middle one*" or "*the second harry potter movie*". Others such as (Funakoshi et al., 2012) resolve anaphoric ("*it*") or exophoric ("*this one*") types of expressions in user utterances to identify geometric objects. In this paper, we study several types of REs to build a natural and flexible interaction for the user.

(Heck et al., 2013) present an intent prediction model enriched with gesture detector to help disambiguate between different user intents related to the interface. In (Misu et al., 2014) a situated in-car dialog model is presented to answer drivers' spoken queries about their surroundings (no display screen). They integrate multi-modal inputs of

speech, geo-location and gaze. We investigate a variety of REs for visual interfaces, and analyze automatic resolution in a classification task introducing a wide range of syntactic, semantic and contextual features. We look at how REs change with screen layout comparing different devices. To the best of our knowledge, our work is first to analyze REs from these aspects.

3 Data

Crowdsourcing services, such as Amazon Mechanical Turk or CrowdFlower, have been extensively used for a variety of NLP tasks (Callison-Burch and Dredze, 2010). Here we explain how we collected the raw utterances from CrowdFlower platform (crowdflower.com).

For each HITApp (Human Intelligence Task Application), we provide judges with a written explanation about our Media App, a SDS built on a device with a large screen which displays items in a grid style layout, and what this particular system would do, namely search for books, music, tv and movies media result [2] Media App returns results based on the user query using an already implemented speech recognition, SLU and dialog engines. For each HIT, the users are shown a different screenshot showing the Media App's search results after a first-turn query is issued (e.g., "*find non-fiction books by Chomsky*" in Table 1). Users are asked to provide five different second turn text utterances for each screenshot. We launch several hitapps each with a different prompt to cover different REs.

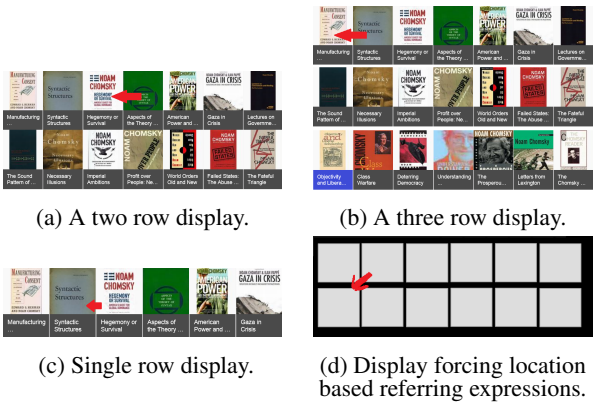
3.1 HITApp Types and Data Collection

A grid of media items is shown to the user with a red arrow pointing to the media result we want them to refer to (see Fig. 1). They can ask to play (an album or an audio book), select, or ask details about the particular media item. Each item in each grid layout becomes a different HIT or screenshot.

3.1.1 Item Layout and Screen Type Variation

The applications we consider have the following row×column layouts: 1×6, 2×6 and 3×6, as shown in Fig. 1 (columns vary depending on the returned item size). By varying the layout, we expect the referent of the last and the bottom layer items to change depending on how many rows, or

[2]Please e-mail the first author to inquire about the datasets.



(a) A two row display.

(b) A three row display.

(c) Single row display.

(d) Display forcing location based referring expressions.

Figure 1: Sketches of different HITApp Screens. The red arrows point to the media we want the annotators to refer.

columns exist in the grid. In addition, phrases like "*middle*", or *"center"* would not appear in the data when there are only one or two rows. Also, we expect that the distribution of types of utterances to vary. For example, in a grid of 1×6, "*the second one*" makes sense, but not so much on a 2×6 grid. We expect similar change based on the number of columns.

We use two kinds of screenshots to collect utterances with variations in REs. The first type of screenshots are aimed to bias the users to refer to items '*directly*' using (full/partial) titles or '*indirectly*' using other descriptors, or meta information such as year the movie is taken, or the author of the book. To collect utterances that indirectly referred to items, we need to show screen shots displaying system results with common titles, eventually forcing the user to use other descriptors for disambiguation. For example, given the first turn query "*find harry potter movies*", the system returns all the *Harry Potter* series, all of which contain the words *Harry Potter* in the title. The user can either refer in their utterance with the series number, the subtitle (e.g. *The prisoners of Azkaban*) or the location of the movie in the grid or by date, e.g., "*the new one*",

Because some media items have long titles, or contain foreign names that are not easy to pronounce, users may chose to refer these items by their location on the display, such as "*top right*", "*first album*", "*the movie on the bottom left*", etc. The second type of screen shots contains a template for each layout with no actual media item (Fig. 1(d)) which simply forces user to use locational references.

3.1.2 Interface Design Variation

In order to test our model's robustness to a different screen display on a new device, we employ an additional collection running another application named *places*, designed for hand-held devices. The places application can assist users in finding local businesses (restaurants, hotels, schools, etc.). and by nature of the device size can display fewer media items and arranges them in a list (one column). The number of items on the screen at any given time depends on the size of the hand-held device screen. The user can scroll down to see the rest of the results. Our collection displays the items in a 3, 4, and 5-rows per 1 column layout as shown in Fig. 2. We use the same variations in prompts as in §3.1. To generate the HitApp screens, we search for nearby places, in the top search engines (Google, Bing) and collect the results to the first turn natural language search queries (e.g.,"*find me sushi restaurants near me*").

Figure 2: A HitApp screen of places app. Items returned by the system regarding the first-turn utterance *"burger places near me?"*

3.2 Data Annotation

We collect text utterances using our media and places application. Using a similar HitApp we labeled each utterance with a domain, intent and segments in utterance with slot tags (see Table 2). The annotation agreement, Kappa measure (Cohen, 1960) is around 85%. Since we are building a relational model between utterances and each item on the screen, we ask the annotators to label each utterance-item as '0' or '1' indicating if the utterance is referring to that item or not. '1' means the item is the intended one. '0' indicates the item is not intended one or the utterance is not referring to any item on the screen, e.g., new search query. We also ask the annotators to label each utterance whether they contain locational (spatial) references.

Domain	Intents (I) & Slots
movie	I: find-movie/director/actor,buy-ticket Slots: name, mpaa-rating (*g-rated*), date,
books	I: find-book, buy-book, Slots: name, genre(*thriller*), author, publisher,
music	I: find-album, find-song, Slots: song-name, genre, album-type,...
tv	I: find-tvseries/play/add-to-queue.. Slots: name, type(*cartoon*), show-time....
places	I: find-place, select-item(*first one*).. Slots: place-type, rating, nearby(*closest*)....

Table 2: A sample of intents and semantic slot tags of utterance segments per domain. Examples for some slots values are presented in parenthesis as *italicized*.

3.3 Types of Observed Referring Expressions

We observe four main categories of REs in the utterances that are collected by varying the prompts and HITApp screens in crowd-sourcing:

Explicit Referential : Explicit mentions of whole or portions of the title of the item on the screen, and no other descriptors, e.g.,"*show me the details of star wars six*" (referring to the item with title "Star wars: Episode VI - Return of the Jedi").

Implicit Referential : The user refers to the item using distinguishing features other than the title, such as the release or publishing date, writers, actors, image content (describing the item image), genre, etc. "*how about the one with Kevin Spacey*".

Explicit Locational : The user refers to the item using the grid design, e.g., "*i want to purchase the e-book on the bottom right corner*".

Implicit Locational : Locational references in relation to other items on the screen, e.g., "*the second of Dan Brown's book*" (showing two of the Dan Brown's book on the same row).

4 Feature Extraction for FIS Model

Here, provide descriptions of each set of features of FIS model used to resolve each expression.

4.1 Similarity Features (SIM)

Similarity features represent the lexical overlap between the utterance and the item's title (that is displayed on the user's screen) and are mainly aimed to resolve *explicit REs*. We represent each utterance u_i and item-title t_k as sequence of words:

$$u_i=\{w_i(1),\ldots,w_i(n_i)\}$$
$$t_k=\{w_k(1),\ldots,w_k(m_k)\}$$

item bigrams	<bos> call five guys and fries <eos>
<bos> five	
five guys	●—●
guys burgers	
burgers and	
and fries	●—●
fries <eos>	●—●

Table 3: Bigram overlap between the item "*five guys burgers and fries*" and utterance"*five guys and fries*".

where $w_i(j)$ and $w_k(j)$ are the *j*th word in the sequence. Since inflectional morphology may make a word appear in an utterance in a different form than what occurs in the official title, we use both the word form as it appears in the utterance and in the item title. For example, *burger* and *burgers*, or *woman* and *women* are considered as four distinct words and all included in the bag-of-words. Using this representation we calculate four different similarity measures:

Jaccard Similarity: A common feature that can represent the ratio of the intersection to the union of unigrams. Consider, for instance, u_i="*call five guys and fries*" and the item t_k="*five guys burgers and fries*" in Fig 2. The Jaccard similarity *S*(*i*,*k*) is:

$$S(i,k)=1-\left(\, c(r_i \cap r_k)/c(r_i \cup r_k)\,\right)$$

where the r_i and r_k are unigrams of u_i and t_k respectively. $c(r_i \cap r_k)$ is the number of common words of u_i and t_k, $c(r_i \cup r_k)$ is the total unigram vocabulary size between them. In this case, the *S*(*i*,*k*)=0.66.

Orthographic Distance: Orthographic distance represent similarity of two text and can be as simple as an edit distance (Levenshtein distance) between their graphemes. The Levenshtein distance (Levenshtein, 1965) counts the insertion, deletion and substitution operations that are required to transform an utterance u_i into item's title t_k.

Word Order: This feature represents how similar are the order of words in two text. Sentences containing the same words but in different orders may result in different meanings. We extend Jaccard similarity by defining bigram word vectors r_i and r_k and look for overlapping bigrams as in Table 3. Among 6 bigrams between them, only 2 are overapping, hence the word-order similarity is *S*(*i*,*k*)=0.33.

Word Vector: This feature is the cosine similarity between the utterance u_i and the item-title t_k that measures the cosine of the angle between them. Here, we use the unigram word counts to represent the word vectors and the word vector similarity is defined as: $S(i,k)=(r_i \cdot r_k)/\|r_i\| \cdot \|r_k\|$.

4.2 Knowledge Graph Features

This binary feature is used to represent overlap between utterance and the meta information about the item and is mainly aimed to resolve *implicit REs*.

First, we obtain the meta information about the on-screen items using Freebase (Bollacker et al., 2008), the knowledge graph that contains knowledge about classes (books, movies, ...) and their attributes (title, publisher, year-released, ...). Knowledge is often represented as the attributes of the instances, along with values for those properties. Once we obtain the attribute values of the item from Freebase, we check if any attribute overlaps with part of the utterance. For instance, given an utterance "*how about the one with Kevin Spacey*", and the item-title "*House of Cards*", the knowledge graph attributes include **year**(*2013*), **cast**(*Kevin Spacey*), **director**(*James Foley*),... We turn the freebase feature 'on' since the actor attribute of that item is contained in the utterance. We also consider partial matches, e.g., last name of the actor attribute.

This feature is also used to resolve implicit REs, with item descriptions, such as "*the messenger boy with bicycle*" referring to the media item *Ride Like Hell*, a movie about a bike messenger. The synopsis feature in Freebase fires the freebase meta feature as the synopsis includes the following passage: "*... in which real messenger boys are used as stunts...* ".

4.3 Semantic Location Labeler (SLL) Feature

This feature set captures spatial cues in utterances and is mainly aimed to resolve *explicit locational* REs. Our goal is to capture the location indicating tokens in utterances and then resolve the referred location on the screen by using an indicator feature. We implement the SLL (Semantic Location Labeler), a sequence labeling model to tag locational cues in utterances using Conditional Random Fields (CRF) (Lafferty et al., 2001).

We sampled a set of locational utterances from each domain to be used as training set. We asked the annotators to label tokens with four different semantic tags that indicate a location.

The semantic tags include row and column indicator tags, referring to the position or pivotal reference. For instance, in "*second from the top*", "*second*" is the `column-position`, and "*top*" is the `row-pivot`, indicating the pivotal reference of the row in a multi-row grid display. Also in "*third from the last*", the "*third*" is the `column-position`, and the "*last*" is the `column-pivot`, the pivotal reference of the column in a multi-column grid display. The fourth tag, `row-position`, is used when the specific row is explicitly referred, such as in "*the Harry Potter movie in the first row*".

To train our CRF-based SLL model we use three types of features: the current word, window words e.g., previous-word, next-word, etc., using five-window around the current word, and syntactic features from the part-of-speech (POS) tagger using the Stanford's parser (Klein and Manning, 2003).

Row Indicator Feature: This feature sets the relationship between the n-gram in an utterance indicated by the `row-position` or `row-pivot` tag and the item's row number on the screen. For instance, given SSL output `row-pivot('top')` and item's location *row*=1, the value of the feature is set to '1'. If no row tag is found by SLL, this feature is set to '0'. We use regular expressions to parse the numerical indicators, e.g., '*top*'='1'.

Column Indicator Feature: Similarly, this feature indicates if a phrase in utterance indicated by the `column-position` or `column-pivot` tag matches the item's column number on the screen. If SLL model tags `column-pivot('on the left')`, then using the item's column number(=1), the value of this feature is set to '1'.

4.4 SLU Features

The SLU (Spoken Language Understanding) features are used to resolve *implicit* and *explicit REs*.

For our dialog system, we build one SLU model per each domain to extract two sets of semantic attributes from utterances: user's intent and semantic slots based on a predefined semantic schema (see examples in Table 2). We use the best intent hypothesis as a categorical feature in our FIS model. Although FIS is not an intent detection model, the intent from SLU is an effective semantic feature in resolving REs. Consider second turn utterance such as "*weather in seattle*", which is a '*find*' intent that is a new search or not related to any item on the screen. We map SLU intents such as *find-book* or *find-place*, to more specific ones, so that the intent feature would have values such as *find*, *filter*, *check-time*, not specific to a domain or device. The intent feature helps us to identify if user's utterance is related to any item on the screen. We also use the best slot hypothesis from the SLU slot model and search if there is full overlap of any recognized slot value with either the item-title or the item meta-information from freebase. In addition, we include the longest slot value n-gram match as an additional feature. We add a binary feature per domain, indicating whether there is a slot value match. Because we are using generic intents as categorical features instead of specific intents, and a slot value match feature instead of domain specific slot types as features, our models are rather domain independent.

5 GBDT Classifier

Among various classifier learning algorithms, we choose the GBDT (gradient boosted decision tree) (Friedman, 2001; Hastie et al., 2009), also known as MART (Multiple Additive Regression Trees). GBDT[3] is an efficient algorithm which learns an ensemble of trees. We find the main advantage of the decision tree classifier as opposed to other non-linear classifiers such as SVM (support vector machines) (Vapnik, 1995) or NN (neural networks) (Bishop, 1995) is the interpretability. Decision trees are "white boxes" in the sense that per-feature gain can be expressed by the magnitude of their weights, while SVM or NN's are generally black boxes, i.e. we cannot read the acquired knowledge in a comprehensible way. Additionally, decision trees can easily accept categorical and continuous valued features. We also present the results of the SVM models.

6 Experiments

We investigate several aspects of the SISI model including its robustness in resolving REs for domain or device variability. We start with the details of the data and model parameters.

We collect around 16K utterances in the media domains (movies, music, tv, and books) and around 10K utterances in places (businesses and

[3]Treenet: http://www.salford-systems.com/products/treenet is the implementation of the GBDT which is used in this paper.

	Movies		Tv		Music		Book		Overall Media		Places	
Feature Description	GBDT	SVM	GBDT	SVM	GBDT	SVM	GBDT	SVM	GBDT	SVM	GBDT	SVM
SLL	79.6	77.1	62.0	62.0	77.1	76.5	63.7	63.0	83.6	82.7	67.9	68.9
SIM	86.6	85.7	78.7	74.1	84.9	84.0	81.6	77.3	88.5	88.3	67.1	66.5
Knowledge Graph (KG)	81.0	82.0	64.8	65.6	86.3	85.4	77.8	77.9	84.4	84.1	76.5	76.5
SLU (Gold)	91.7	91.8	89.1	88.5	87.8	87.5	86.3	84.9	83.7	83.2	77.8	71.1
SLU (Pred.)	75.8	72.6	80.3	79.8	84.3	84.1	82.4	82.4	81.4	80.9	71.4	67.8
SIM+SLL	90.9	90.2	87.2	87.1	85.9	86.2	88.5	87.6	91.9	91.9	78.9	73.4
SIM+SLL+KG	91.7	91.3	89.9	89.1	89.1	87.7	91.4	90.3	93.0	92.7	85.9	82.3
SIM+SLL+KG+SLU(Gold)	**96.2**	95.01	**95.2**	95.09	**90.3**	89.9	**94.6**	94.0	**93.7**	93.2	**86.3**	84.3
SIM+SLL+KG+SLU(Pred.)	90.9	90.8	92.3	92.00	86.9	85.7	93.1	93.0	89.3	88.9	85.7	83.9

Table 5: Performance of the FIS models on test data using different features. Acc:Accuracy,. SIM: similarity features; SLU:Spoken Language Understanding features (intent and slot features); SLL:Semantic Locational Labeler features; Gold: using true intent and slot values, Pred.: using predicted intent and slot values from the SLU models.

Model:	Movies	TV	Music	Books	Places
Intent Acc.	84.5%	87.4%	87.6%	98.1%	89.5%
Slot F-score	92.1F	89.4F	88.5F	86.6F	88.4F

Table 4: The performance of the SLU Engine's intent detection models in accuracy (Acc.) and slot tagging models in F-Score on the test dataset.

locations) domain. We also construct additional negative instances from utterance-item pairs using first turn non-selection queries, which mainly indicate a new search or starting over. In total we compile around 250K utterance-item pairs for media domains and 150K utterance-item pairs for the places domain.[4] We randomly split each collection into 60%-20%-20% parts to construct the train/dev/test datasets. We use the dev set to tune the regularization parameter for the GBDT and SVM using LIBSVM (Chang and Lin, 2011) with linear kernel.

We use the training dataset to build the SLU intent and slot models for each domain. For the intent model, we use the GBDT classifier with n-gram and lexicon features. The lexicon entries are obtained from Freebase and are used as indicator variables, e.g., whether the utterance contains an instance which exists in the lexicon. Similarly, we train a semantic slot tagging model using CRF method. We use n-gram features with up to five-gram window, and lexicon features similar to the intent models. Table 4 shows the accuracy and F-score values of SLU models on the test data. The slot and intent performance is consistent accroess domains. The books domain has only two intents and hence we observe much better intent performance compared to other domains.

[4]In the final version of the paper, we will provide annotated data sets on a web page, which is reserved due to blind review.

6.1 Impact of Individual FIS Features

In our first experiment, we investigate the impact of individual feature sets on FIS model's performance. We train a set of FIS models on the entire media dataset to investigate the per-feature gain on the test dataset for each domain. We also train another set of FIS models with the same feature sets, this time on the places dataset and present the results on the places test set. Table 5 shows the results. We measure the performance starting with individual feature sets, and then incrementally add each feature set. Note that the SLU feature set includes the categorical intent, binary slot-value match and the longest slot value n-gram match with the item's title or meta information. The SLL feature set includes two features indicating the row and column (see §4.3).

As expected, larger gains in accuracy are observed when features that resolve different REs are used. Resolving locational cues in utterances with SLL features considerably impacts the performance when used together with similarity (SIM) features. We see a positive impact on performance as we add the knowledge graph features, which are used to resolve implicit REs. Using only the predicted SLU features in feature generation without golden values degrades the performance. Although the results are not statistically significant, the GBDT outperforms the SVM for almost all models, except for a few models, where the results are similar. However, the models which

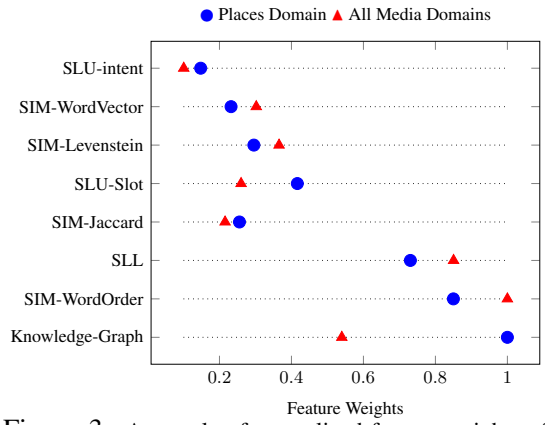


Figure 3: A sample of normalized feature weights of the GBDT FIS models across domains.

combine different features as apposed to individual feature set (the above the line models versusu below the horizantil line models) are statistically significant (based on the student t-test p¡0.01).

Next, we illustrate the significance of individual features across domains as well as devices. Fig. 3 compares the normalized feature weights of media and places domains. Across domains there are similar features with similar weight values such as SLU-intent, some similarity features (SIM-) and even spatial cue features (SLL). It is not surprising to observe that the places domain knowledge-graph meta feature weights are noticeably larger than all media model features. We think that this is due to the way the REs are used when the device changes (places app is on a phone with a smaller screen display). Especially, places application users refer items related to restaurants, libraries, etc., not so much by their names, but more so with implicit REs by using: the location (referring to the address: "*call the one on 31 street*") or cuisine ("*Chinese*"), or the star-rating ("*with the most stars*"), etc.

6.2 Resolution Across REs

We go on to analyze the performance of different RE types. A particularly interesting set of errors we found from the previous experiments are those that involve implicit referrals. Table 6 shows the distribution of different REs in the collected datasets.

Some noticeable instances with false positives for implicit locational REs include ambiguous cases or item referrals with one of its facets that require further resolution including comparison to other items, e.g., "*the nearest one*". Table 7 shows further examples. As might be expected, the locational cues are less common compared to other

	All Media		Places	
Utterance Type	%	Acc.	%	Acc.
All utterances	100%	93.7%	100%	86.3%
Direct/Indirect RE	81%	93.9%	73%	86.9%
Locational RE	19%	92.5%	28%	85.2%
Explicit RE	60%	94.3%	45%	88.4%
Implicit RE	21%	83.4%	28%	72.2%
Explicit Locational RE	15%	75.2%	24%	86.2%
Implicit Locational RE	3%	56.6%	2%	56.7%

Table 6: Distribution of referring expressions (**RE**) in the media (large screen like tv) and places (handheld device like phone) corpus and the FIS accuracies per RE type.

Utterance	Displayed on screen
"*the most rated restaurant*"	★★★'s next to each item
"*first thomas crown affair*"	original release (vs. remake)
"*second one over*"	(incomplete row/col. information)

Table 7: Display screen as user utters.

expressions. We also confirm that the handheld (places domain) users implicitly refer to the items more commonly compared to media app, and use the contextual information about the items such as their location, address, star-rating, etc. The models are considerably better at resolving explicit referrals (both non-spatial and spatial) compared to implicit ones. However, for locational referrals, the difference between the accuracy of implicit and explicit REs is significant (75.2% vs. 56.6% in media and 86.2% vs. 56.7% in places). Although not very common, we observe negative expressions, e.g., "*the one with no reviews*", which are harder for the FIS to resolve. They require quantifying over every other item on the screen, namely the context features, which we leave as a future work.

6.3 New Domains and Device Independence

In the series of experiments below, we empirically investigate the FIS model's robustness to when a new domain or device is introduced.

Robustness to New Domains: So far we trained media domain FIS models on utterances from all domains. To investigate how FIS models would behave when tested on a new domain, we train additional models by leaving out utterances from one domain and test on the left out domain. We used GBDT with all the feature sets. To set up an upper bound, we also train models on each individual domain and test on the same domain.

Table 8 shows the performance of the FIS mod-

	Models tested on:			
Model trained on:	**Movies**	**TV**	**Music**	**Books**
All domains	96.2%	95.2%	90.3%	94.6%
All other domains	94.6%	92.4%	89.7%	%
Only *this domain	96.4%	96.8%	93.4%	%

Table 8: Accuracy of FIS models tested on domains that are: seen at training time (all domains), unseen at training time (all other domains) and trained on individual domains.

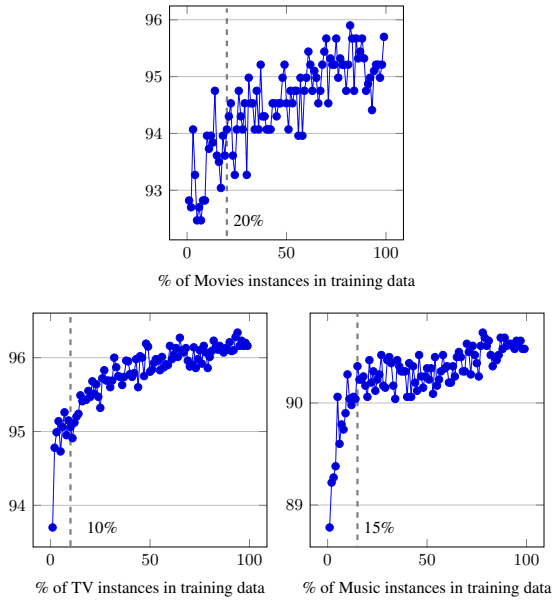


Figure 4: The accuracy (y-axis) versus the percentage (%) of in-domain utterances used in the training dataset. The dashed vertical line indicates an optimum threshold for the amount of in-domain data to be added to the training data.

els in accuracy on each media test domain. The first row shows the results when all domains are used at training time (same as in Table 5). The second row represents models where one domain is unseen at training time. We notice that the accuracy, although degraded for movies and tv domains, is in general not significantly effected by the domain variations. We setup another experiment, where we incrementally add utterances from the domain that we are testing the model on. For instance, we incrementally add random samples from movies training utterances on the dataset that does not contain movies utterances and test on all movies test data. The charts in Fig. 4 show the % improvement in accuracy as in-domain data is incrementally added to the training dataset. The results are interesting in that, using as low as 10-20% in-domain data is sufficient to build a flexible item selection model given enough utterances from other domains with varying REs.

Robustness to a New Device: The difference between the vocabulary and language usage observed in the data collected from the two devices

Media
"*only the new movies*" ; "*second one on the left*"
"*show me the thriller song*"; "*by Lewis Milestone*"
"*the first harry potter book*"
Places
"*directions to Les Schwab tire center*"
"*the closest one*" ;"*show me a map of ...*"
"*get hours of Peking restaurant*"; "*call Mike's burgers*"

Table 9: Sample of utterances collected from media and places applications illustrating the differences in language usage.

Trained on	Tested on Media	Tested on Places
Media	93.7 %	85.9%
Places	85.9%	86.3%
Media+Places	92.7%	85.8%

Table 10: Accuracy of FIS models tested on two separate devices (large screen media, and small screen places) that are unseen at test time.

is mainly due to changes in: (*i*) the screen design (places on phone has one column format wheres the media app has multi-column layout); (*ii*) the domain of the data. Table 9 shows some examples. Here, we add a little bit of complexity, and train one FIS model using the training data collected on one device and test the model on a different one, which is unseen at training time. Table 10 shows the comparisons for media and phone interfaces. The results are interesting. The performance of the places domain on phone does not get affected when the models are trained on the media data and tested on the phone device (86.3% down to 85.9% which is statistically insignificant). But when the data is trained on the places and tested on the media, we see a rather larger degradation on the performance (93.7% down to 85.9%). This is due to the fact that the media display screens are much complicated compared to phone resulting in a larger vocabulary with more variation in REs compared to places domain.

6.4 Conclusion

We presented a framework for identifying and recognizing referring expressions in user utterances of human-machine conversations in natural user interfaces. We use several on-screen cues to interpret whether the user is referring to on-screen items, and if so, which item is being referred to. We investigate the effect of different set of features on the FIS models performance. We also show that our model is domain and device independent which is very beneficial when new do-

mains are added to the application to cover more scenarios or when FIS is implemented on new devices. As a future work, we would like to adapt our model for different languages and include other features from multi modality including gesture or geo-location.

References

Dimitr Anastasiou and Cui Jian and Desislava Zhaekova. 2012. Speech and gesture interaction in an ambient assited living lab. *In. Proc. of the 1st Workshop on Speech and Multimodal Interaction in Assitive Environments at ACL'2012*.

Rajesh Balchandran, and Mark E. Epstein, and Gerasimos Potamianos, and Lsadislav Seredi. 2008. A multi-modal spoken dialog system for interactive tv. *In. Proc. of the 10th International Conference on Multimodal Interfaces*.

Christopher M. Bishop. 1995. *Neural networks for Pattern recognition*.

Kurt Bollacker and Colin Evans and Praveen Paritosh and Ttim Sturge and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. *In. Proc. of the 2008 International Conference on Management of Data (SIGMOD-08)*.

Richard A. Bolt. 1980. Put-that-there: Voice and gesture at the graphics interface. *Computer Graphics*.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with amazons mechanical turk. *In. Proc. of NAACL*.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Herbert H. Clark and Deanna Wilkes-Gibbs. Referring as colloborative processes.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20.

Robert Dale and Jette Viethen. 2009. Referring expression generation through attribute-based heuristics. *In Proc. of the 12th European Workshop on Natural Language Generation (ENLG)*.

Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics, 2001*.

Kotaro Funakoshi, Mikio Nakano, Takenobu Tokunaga, and Ryu Iida. 2012. A unified probabilistic approach to referring expressions. *In Proc. of the Special Interest Group on Discourse and Dialog (SIGDIAL)*.

Petra Gieselmann. 2004. Reference resolution mechanisms in dialogue management. *In Proc. of the 8th Workshop on the semantics and pragmatics of dialogues (CATALOG)*.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 410–419, Cambridge, MA, October. Association for Computational Linguistics.

Joakim Gustafson, Linda Bell, Jonas Beskow, Johan Boye, Rolf Carlson, Jens Edlund, Bjorn Granstrom, David House, and Mats Wiren. 2000. Adapt - a multimodal conversational dialogue system in an apartment domain. *In Proc. of the 6th International Conference on Spoken Language Processing (ICSLP)*, pages 134–137.

Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. 2009. *The Elements of Statistical Learning (2nd ed.) Chapter 10. Boosting and Additive Trees, 2009*.

Larry Heck, Dilek Hakkani-Tur, Madhu Chinthakunta, Gokhan Tur, Rukmini Iyer, Partha Parthasarathy, Lisa Stifelman, Elizabeth Shriberg, and Ashley Fidler. 2013. Multi-modal conversational search and browse. *In Proc. of the IEEE Workshop on Speech, Language and Audio in Multimedia*.

Dvaid Huggins-Daines and Alexander I. Rudnicky. 2008. Interactive asr error correction for touchscreen devices. *In Proc. of ACL, Demo session*.

Srinivasan Janarthanam and Oliver Lemon. 2010. Adaptive referring expression generation in spoken dialog systems: Evaluation with real users. *In Proc. of SIGDIAL 2010: the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Mark Johnston, Srinivas Bangalore, Gunaranjan Vasireddy, Amanda Stent, Patrick Ehlen, Marilyn Walker, and Steve Whittaker and Preetam Maloor. 2002. Match: an architecture for multimodal dialog systems. *In. Proc. of the ACL*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing.

David B. Koons, Carlton J. Sparrell, and Kristinn R. Thorisson. 1993. Integrating simultaneous input from speech, gaze and hand gestures. *In Proc. of the In Maybury, M. (Ed.), Intelligent Multimedia Interfaces*, pages 257–276.

John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. *In Proc. ICML*.

Vlademir Levenshtein. 1965. Binary codes capable of correcting deletions, insertions and reversals. *In Proc. of the Doklady Akademii Nauk SSSR*, 163:845–848.

Teruhisa Misu, Antoine Raux, Rakesh Gupta, and Ian Lane. 2014. Situated language understanding at 25 miles per hour. *In. Proc. of the SIGDIAL - Annual Meeting on Discourse and Dialogue*.

Renato De Mori, Frederic Bechet, Dilek Hakkani-Tur, Michael McTear, Giuseppe Riccardi, and Gokhan Tur. 2008. Spoken language understanding: A survey. *IEEE Signal Processing Magazine*, 25:50–58.

Joseph G. Neal. 1991. Intelligent multimedia interface technology. *In Proc. of the Intelligent User Interfaces: In Sullivan, J., and Tyler, S. (Eds.)*, pages 45–68.

Sharon Oviatt, Antonella DeAngeli, and Karen Khun. 1997. Integration and synchronization of input modes during multimodal human-computer interaction. *In Proc. of the Human Factors in Computing Systems: CHI*, pages 415–422.

Nobert Pfleger and Jan Alexandersson. 2006. Towards resolving referring expressions by implicitly activated referents in practical dialog systems. *In. Proc. of the 10th Workshop on the Semantics and Pragmatics of Dialog (SemDial-10)*.

Michael F. Schober and Herbert H. Clark. 1989. Understanding by addressees and overhearers. *In Proc. of the Cognitive Psychology*, pages 211–232.

Vlademrr Vapnik. 1995. *The nature of statistical learning theory*.

Building Chinese Discourse Corpus with Connective-driven Dependency Tree Structure

Yancui Li[1,2] **Wenhe Feng**[2] **Jing Sun**[1] **Fang Kong**[1] **Guodong Zhou**[1]
[1]Natural Language Processing Lab, School of Computer Science and Technology, Soochow University, Suzhou 215006, China
[2]Henan Institute of Science and Technology, Xinxiang 453003, China
{yancuili, wenhefeng}@gmail.com {20104027009, kongfang, gdzhou}@suda.edu.cn

Abstract

In this paper, we propose a Connective-driven Dependency Tree (CDT) scheme to represent the discourse rhetorical structure in Chinese language, with elementary discourse units as leaf nodes and connectives as non-leaf nodes, largely motivated by the Penn Discourse Treebank and the Rhetorical Structure Theory. In particular, connectives are employed to directly represent the hierarchy of the tree structure and the rhetorical relation of a discourse, while the nuclei of discourse units are globally determined with reference to the dependency theory. Guided by the CDT scheme, we manually annotate a Chinese Discourse Treebank (CDTB) of 500 documents. Preliminary evaluation justifies the appropriateness of the CDT scheme to Chinese discourse analysis and the usefulness of our manually annotated CDTB corpus.

1 Introduction

It is well-known that interpretation of a text requires understanding of its rhetorical relation hierarchy since discourse units rarely exist in isolation. Such discourse structure is fundamental to many text-based applications, such as summarization (Marcu, 2000) and question-answering (Verberne et al., 2007). Due to the wide and potential use of discourse structure, constructing discourse resources has been attracting more and more attention in recent years. In comparison with English, there are much fewer discourse resources for Chinese which largely restricts the researches in Chinese discourse analysis.

The general notion of discourse structure mainly consists of discourse unit, connective, structure, relation and nuclearity. However, previous studies on discourse failed to fully express these kinds of information. For example, the Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) represents a discourse as a tree with phrases or clauses as elementary discourse units (EDUs). However, RST ignores the importance of connectives to a great extent. Figure 1 gives an example tree structure with four EDUs (e1-e4). In comparison, Penn Discourse Treebank (PDTB) (Prasad et al., 2008) adopts the predicate-argument view of discourse relation, with discourse connective as predicate and two text spans as its arguments. Example (1) shows an explicit reason relation signaled by the discourse connective "particularly if" and an implicit result relation represented by the inserted discourse connective "so", with Arg1 in italics and Arg2 in bold. However, as a connective and its arguments are determined in a local contextual window, it is normally difficult to deduce a complete discourse structure from such a connective-argument scheme. In this sense, the PDTB at best only provides a partial solution to the discourse structure.

[Catching up with commercial competitors in retail banking and financial services,] e1 [they argue,] e2 [will be difficult,] e3 [particularly if market conditions turn sour.]e4

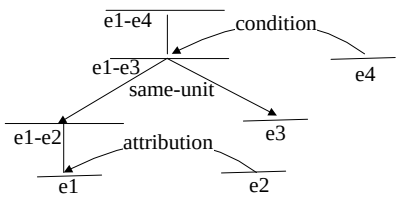


Figure 1: An example of discourse structure in RST

Example (1): An example of the connective-argument scheme in PDTB

A)[Catching up with commercial competitors in retail banking and financial services will be difficult]$_{Arg1}$, they argue, will be difficult, particularly **if** **[market**

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2105–2114, October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

conditions turn sour] $_{Arg2}$. (Contingency.Condition. Hypothetical) (0616)

B) So much of the stuff poured into its Austin, Texas, offices *[that its mail rooms there simply stopped delivering it.]*$_{Arg1}$ (Implicit = so)**[Now, thousands of mailers, catalogs and sales pitches go straight into the trash.]**$_{Arg2}$ (Contingency.Cause. Result) (0989)

Obviously, both RST and PDTB have their own advantages and disadvantages in representing different characteristics of the discourse structure. In this paper, we attempt to propose a new scheme to Chinese discourse structure, adopt advantages of the tree structure from RST and connective from PDTB. Meanwhile, the special characteristics of Chinese discourse structure are well addressed.

First, it is difficult to define EDU in Chinese due to the frequent occurrence of the ellipsis of subjects, objects and predicates, and the lack of functional marks for EDU. Second, the connectives in Chinese omit much more frequently than those in English with about 82.0% vs. 54.5% in Zhou and Xue (2012). In Example (2), there are even no explicit connectives. Third, previous studies have shown the difference in classifying Chinese discourse relations from English (Xing, 2001; Huang and Liao, 2011). This suggests that the discourse relations defined for English (both RST and PDTB) are not readily suitable for Chinese. Finally, the nucleus of a Chinese discourse relation is normally not directly related to a particular relation type but should be dynamically determined from the global meaning of a discourse.

Example (2): An example of discourse with 4 EDUs

[据悉， 东莞 海关 共 接受
According to reports,Dongguan Customs total accept
企业 合同 备案 八千四百多份，]e1 [比 试点
company contract record 8400 plus class, than pilot
前 略 有 上升，]e2 [企业 反应 良好，]e3
before a slight increase, company responses well,
[普遍 表示 接受。]e4
generally acknowledge acceptance.

"[According to reports, Dongguan District Customs accepted more than 8400 records of company contracts,] e1 [a slight increase from before the pilot.]e2 [Companies responded well,]e3 [generally acknowledging acceptance.]e4"

In this paper, we present a Connective-driven Dependency Tree (CDT) discourse representation scheme, which takes advantage of both RST and PDTB, with elementary discourse units (limited to clauses) as leaf nodes and connectives as non-leaf nodes. Especially, we define EDU from three aspects, and employ the connective' level and semantic to indicate the rhetorical structure and the discourse relation. Besides, the nuclearity of discourse units in a discourse relation is decided on the overall discourse meaning. On the basis, we adopt the CDT scheme to annotate a certain scale corpus, called Chinese Discourse Treebank (CDTB) thereafter in this paper. Evaluation shows the appropriateness of the CDT scheme to Chinese discourse analysis.

The rest of this paper is organized as follows. Section 2 overviews related work. In Section 3, we present the CDT discourse representation scheme. In Section 4, we describe the annotation of the CDTB corpus. Section 5 compares CDTB with other major discourse corpora. Section 6 gives the experimental results on EDU recognition, the crucial step for discourse parsing. Finally, conclusion is given in section 7.

2 Related Work

In the past decade, several discourse corpora for English have emerged, with the Rhetorical Structure Theory Discourse Treebank (RST-DT) (Carlson et al., 2003) and the Penn Discourse Treebank (PDTB) (Prasad et al., 2008) most prevalent.

In the RST framework, a text is represented as a discourse tree, with non-overlapping text spans (either phrases or clauses) as leaves, and adjacent nodes are related through particular rhetorical relations to form a discourse sub-tree, which is then related to other adjacent nodes in the tree structure. According to RST, there are two types of discourse relations, mononuclear and multinuclear. Figure 1 shows an example of discourse tree representation, following the notational convention of RST. Among the four EDUs (e1-e4), e1 and e2 are connected by a mononuclear relation "attribution", where e1 is the nucleus, the span (e1-e2) and the EDU e3 are further connected by a multi-nuclear relation "same-unit", where they are equally salient. Annotated according to the RST framework, the RST-DT consists of 385 documents from the Wall Street Journal (WSJ). Besides, the original 24 discourse relations defined by Mann and Thompson (1988) are further divided into a set of 18 relation classes with 78 finer grained rhetorical relations in RST-DT.

As the largest discourse corpus so far, the Penn Discourse Treebank (PDTB) contains over one million words from WSJ. With EDUs limited to clauses, the PDTB adopts the predicate-

argument view of discourse relations, with connective as predicate and two text spans as its arguments. Example (1) shows two annotation tokens for the connective "particularly if" and "so". The current version of PDTB 2.0 annotates 40600 tokens, including 18459 explicit relations of 100 distinct types (e.g. "particularly if" and "if" are the same type) and 16224 implicit discourse relations of 102 distinct token types. Besides, PDTB provides a three level hierarchy of relation tags with the first level consisting of four major relation classes (Temporal, Contingency, Comparison, and Expansion), which are further divided into 16 types and 23 subtypes.

In comparison, there are few researches on Chinese discourse annotation (Xue, 2005a; Chen, 2006; Yue, 2008; Huang and Chen, 2011; Zhou and Xue, 2012), with no exception employing existing RST or PDTB frameworks. For example, Zhou and Xue (2012) use the PDTB annotation guidelines to annotate Chinese discourse with 98 files from Chinese Treebank (Xue et al., 2005b) of Xinhua newswire. In particular, they adopt a lexically grounded approach and make some adaptation based on the linguistic and statistical characteristics of Chinese text, with Arg1 and Arg2 defined semantically and the senses of discourse relations annotated besides connectives and their lexical alternatives. The agreement on relation types reaches 95.1% and the agreement on implicit relations with exact span match reaches 76.9%.

Instead, Chen (2006) and Yue (2008) use RST to annotate Chinese discourse. Chen (2006) selects comma as the segmentation signal of EDUs (in Example (2), "据悉(According to reports)" will be segmented as an EDU), and finds that RST fails to deal with some special features of Chinese. Yue (2008) manually annotates a set of 97 texts according to RST and shows the cross-lingual transferability of RST to Chinese. However, it also shows that EDUs in Chinese are much different from those in English, and many relation types in Chinese have no correspondence to English, and vice versa.

3 Connective-driven Dependency Tree

An appropriate representation scheme is fundamental to linguistic resource construction. With reference to various theories and representation scheme on the tree structure and nuclearity of RST, the connective, relation and discourse structure of Chinese complex sentence (Xing, 2001), the sentence-group theory (Cao, 1984), the connective treatment of PDTB, the conjunction dependent analysis (Feng and Ji, 2011) and the center theory of dependency grammar (Hays, 1964), we propose a new discourse representation scheme for Chinese, called Connective-driven Dependency Tree (CDT), with EDUs as leaf nodes and connectives as non-leaf nodes, to accommodate the special characteristics of the Chinese language in discourse structure.

For instance, Example (3) consists of 2 sentences, which is part of a paragraph from "chtb_0001", and its corresponding CDT representation is shown in Figure 2. Here, the number of "|" in Example (3) stands for the level of EDUs in CDT and the numbers marked in Figure 2 (such as 1, 2 etc.) distinguish EDUs. While an arrow points to the main EDU or main discourse unit (called nucleus), the combination of different EDUs can be considered as EDUs in a higher level and the new discourse units can thus be combined into higher-level units from bottom to up. In this way, the discourse structure can be expressed as a tree structure via bottom-up combination of EDUs.

Obviously, such discourse structure is constructed by two kinds of basic units, EDUs (leaf nodes) and connectives (non-leaf nodes). On the one hand, connectives can represent the discourse structure by its hierarchical level in the tree. The discourse structure is independent on the connective level essentially, rather than the reverse. On the other hand, connectives themselves can represent the discourse relation. This is why we call the scheme "Connective-driven". As for the abstract discourse relation, we can construct a set of discourse relations, mapping a connective to discourse relation, according to the users' specific requirements.

Example (3): CDT example from CTB

1 浦东 开发 开放 是 一项 振兴 上海，建设
Pudong development open up is a promote Shanghai, construct
现代化 经济、贸易、 金融 中心 的 跨世纪
modern economy, trade, financial century De cross-century
工程，‖2 (因此) 大量 出现的是 以前 不曾
project, therefore a large number arisen De previously never
遇到过的 新 情 况、新问题。| 3 (对此)， 浦东 {不是}
encounter DE new situation, new problem.To this, Pudong not
简单 的 采取 "干 一段 时间，等 积累 了
simply DE adopting "does a period time, wait accumulate Le
经验 以后再制定 法规 条例" 的 做法，‖4{而是}
experience after re-enactment laws regulations De approach,but
借鉴 发达 国家 和 深圳 等 特区 的
learn developed countries and Shenzhen etc. special zone DE
经验 教训，|||| 5<并且>聘请 国内外 有关专家
experience lesson, Invite at home and abroad revlant expert
学者，|||| 6<并且>积极、及时地 制定 和 推出

scholars, actively, timely DI formulate and issuing
法规性 文件，||| 7 {使} 这些 经济 活动 一 出现就 被
statutory file, make these economic activity as soon as appear bei
纳入 法制 轨道。
bring into legality track.

"1 Pudong's development and opening up is a century-spanning undertaking for vigorously promoting Shanghai and constructing a modern economic, trade, and financial center. ‖ 2 **Because of this,** new situations and new questions that have not been encountered before are emerging in great numbers. | 3 In response to this, Pudong is not simply adopting an approach of "work for a short time and then draw up laws and regulations only after experience has been accumulated."‖ 4 **Instead,** Pudong is taking advantage of the lessons from experience of developed countries and special regions such as Shenzhen, ||||5 by hiring appropriate domestic and foreign specialists and scholars, ||||6 actively and promptly formulating and issuing regulatory documents. ||| 7 So these economic activities are incorporated into the sphere of influence of the legal system as soon as they appear."

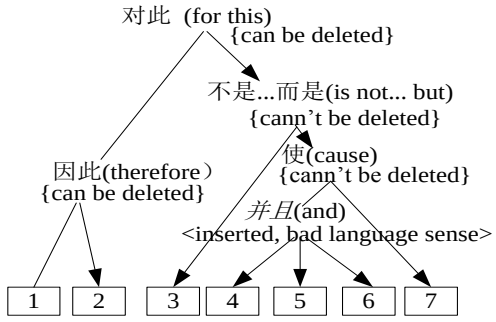


Figure 2: CDT representation of Example (3)

3.1 Elementary Discourse Unit

As the leaf nodes of CDT, EDUs are limited to clauses. In principle, EDUs play a crucial role to discourse analysis. Since from bottom-up discourse combination, EDUs are the start of discourse analysis, while from top-down discourse segmentation, they are the end of discourse analysis. Unfortunately, since there lacks obvious distinction between Chinese sentence structure and phrase structure, it is rather difficult to define Chinese EDU (clause). Till now, there is still no widely accepted definition in the Chinese linguistics community (Wang, 2010). Inspired by Li et al. (2013a), we give the definition of Chinese EDU from three perspectives. First, from the syntactic structure perspective, an EDU should contain at least one predicate and express at least one proposition. Second, from the functional perspective, an EDU should be related to other EDUs with some propositional function, i.e. not act as a grammatical element of other EDUs. Finally, from the morphological perspective, an EDU should be segmented by some punctuation, e.g. comma, semicolon and period. We use punctuation because there usually has a pause between clauses (EDUs), which can be shown in written commas, semicolons etc (Huang and Liao, 2011). Normally, it is easy to handle complex sentences and special sentence patterns (e.g. serial predicate sentences). For Example (4), A) is a single sentence with serial predicate; B) is complex sentence with two EDUs (clauses):

Example (4): EDU examples

A) He opened the door and went out. (single sentence, serial predicate, one EDU)
B) 1 He opened the door,| 2 **and** went out. (complex sentence, two EDUs)

Take as example, there exist 7 EDUs in Example (3), each marked with a number in front. According to our definition, the fragment "干一段时间，…法规条例"("work for a short time…has been accumulated") in EDU 3 is not segmented as a EDU since: 1) it acts as a grammatical element of other EDUs and has no direct relationship with other EDUs on propositional function; 2) it is marked by a pair of quotation marks and does not end with any punctuation. In contrast, the fragment "而是借鉴发达…法制轨道"("but learn developed…legality track.") is segment as 4 EDUs since it meets the three criteria in our EDU definition.

3.2 Connective

As non-leaf nodes in the CDT representation, connectives connect EDUs or discourse units. Thus, the main criterion of determining whether an expression is a connective is to check whether the two fragments it connects are EDUs (or discourse units). In our scheme, the list of explicit discourse connectives is judged by a data driven approach, i.e. with any discourse-like word or phrase marked as connective in the annotation practice, e.g. "因此(therefore)", "对此(to this)", "不是...而是...(is not...but....)", "使(so that), "正因为(just because)" in Example (3), "先...然后(first...then) ", "同时也(and at the same time)" in Example (5).

Example (5): Connective examples from CTB

A) 1<如果；只要>建筑公司进区，| 2 有关部门**先**送上这些法规性文件，‖ 3 **然后**有专门队伍进行监督检查。(chtb_0001)
1<If ; As long as>The construction company enters the region, |2 **first** the appropriate bureau delivers these regulatory documents,‖ 3 **Then** there is a specialized contingent that carries out a supervisory inspection.
B) 1 加工贸易…, 2 **同时 也** 是粤港澳台经贸合作的重要内容。(chtb_0031)

1 The processing trade …, | 2 **and at the same time** is important content in the economic and trade cooperation between Guangdong, Hong Kong, Macao and Taiwan.

It is worthy of mention that from the part-of-speech perspective, connectives are not necessarily conjunctions. For example, in Example (3) and (5), adverbs "先...然后(first…then)", verb phrases "不是…而是(is not…but)", and preposition phrases "对此(to this)" are determined as connectives. From the morphological perspective, a connective may contain more than one word, even discontinuous. As a common occurring phenomenon in Chinese discourse, there exist many paired Chinese connectives, e.g. "不是…而是 (is not…but)" in Figure 2. Even in some paired connectives, such as "因为…所以(because…so)", a word in a paired connective can appear independently as a connective. Please note that this may not be applied to other cases, e.g. "不是…而是 (is not…but)" as appeared in Example (3). Moreover, in many cases whether an expression is a connective or not depends on its meaning, e.g., "为 (in order to)" is a connective, while "为 (for)" is not. For the positional distribution, a connective may appear anywhere, i.e. in the beginning, middle, or the end of the first or second EDU. Example (3) and (5) show some of cases in different positions. The above characteristics pose special challenges on connective determination in Chinese language.

According to the appearance of a connective or not, a discourse relation can be either explicit or implicit. Previous studies have shown the difficulty of implicit relation recognition in English due to the omission of connectives (Pitler et al., 2009; Lin et al., 2009). This becomes even worse in Chinese since compared with the implicit ratio of 54.5% in English connectives, this ratio rises up to about 82% in Chinese (Zhou and Xue, 2012). It is worth noting that the majority of discourse relations in Chinese are implicit, so the insertion of a connective in an implicit position can significantly ease the understanding of the discourse. That is, a connective driven representation scheme is still applicable to a discourse with implicit connectives. To help determine implicit relations, two special strategies are proposed.

First, for each explicit connective, a decision is made whether or not it can be deleted without changing the rhetorical relation of a discourse. It should be emphasized that this constraint is largely semantic. The motivation behind the removal of explicit connectives is to enlarge implicit instances and help recognize implicit relations. As shown in Figure 2, we use the paired mark "()" to indicate that a connective can be deleted, e.g. connectives "(对此 to this)", "(因此 therefore)", "(正因此 just because)", and the paired mark "{}" to indicate that a connective cannot be deleted, e.g. connectives "{使 so that}", "{不是…而是 is not…but}".

Second, since a connective can be inserted to represent an implicit relation, our scheme tries to insert a connective which can be easily interpreted from the semantic perspective with little ambiguity into the most appropriate place. Most of the connective insertions for implicit relations occur between adjacent discourse spans. It is worth noting that not all implicit connectives are subjective to the language sense. To mark this difference, we cluster implicit connectives into two categories according to their language senses, either "good language intuition" or "bad language intuition". In our scheme, we use the paired mark "<>"to indicate inserted implicit connectives, e.g. connectives"<例如 e.g.>", "<却 but>" with "good language sense", connective "<并且 and>" with "bad language sense", as shown in Figure 2.

In some cases, it is possible that there exist several insertion options for an implicit connective due to the ambiguity in a discourse. For example, in Example (5A), connectives "如果 (if)" and "只要 (as long as)" are inserted into the first level to show the two discourse relation options. As far as this happens, connectives are inserted and ordered according to annotators' first intuition.

3.3 Discourse Structure

In Figure 2, the paragraph is organized as a tree structure, in which EDUs appear in the leaf nodes and the connectives appear in the non-leaf ones. The adoption of tree structure conforms to traditional Chinese discourse theories and practice. For example, a native Chinese speaker tends to determine the overall level boundary first and then the analysis goes on step by step to the individual clauses, when understanding a complex sentence. This process naturally forms a tree structure. Besides, tree structure is easier to formalize, compared with graph.

More specifically, the hierarchical structure of connectives indicates the hierarchical structure of discourse units. Apparently, discourse struc-

ture analysis can be viewed as hierarchical analysis of connectives, with hierarchical connective structure reflecting hierarchical combination of discourse units. Essentially, the discourse hierarchy indicates the correlation degrees of semantic relations in the discourse, the deeper tree level of two discourse units, the higher correlation degree of their semantic relation. Therefore, a discourse relation is the ultimate factor for the choice of hierarchical discourse structure. For a reference, please take Sentence 2 in Figure 2 as an example.

3.4 Discourse Relation

For discourse relation representation, a general approach is to assign an abstract relation type to a discourse relation directly, such as cause, conjunction, condition, purpose, etc, as done in RST-DT and PDTB. In our CDT scheme, we avoid to directly assign an abstract relation type to a discourse relation. Instead, we use the connective itself to express the discourse relation, as shown in Figure 2. In this way, the difficulty of pre-defining a set of acknowledged discourse relations and selecting an exact discourse relation can be avoided during the corpus annotation process. Since a Chinese discourse relation is largely controlled by connective (Xing, 2001), the key to determine a relation is to identify a suitable connective. Normally, most of relation annotations can easily map from connectives to abstract semantic classes of relations, if necessary, with the help of the discourse context. The majority of discourse relations in Chinese are implicit, but it makes sense to insist on a connective driven representation. With connective as a bridge, at least it makes discourse representation easier.

For the abstraction of discourse relations, we leave it in a later separate stage. Of course, there are cases where a connective may represent more than one discourse relation. For example, connective "而" can denotes the continuous relation "而 (especially)" and the transitional relation "而 (however)". Compared with annotating discourse relation directly, annotator's intuition is more accurate for specific connective. We don't object to label discourse relation, referring to the general work and Chinese analysis practice, give a set of relations (Figure 3), regarding it as connective's semantics, and then annotate the connective with it. In this way, we can obtain a general relation set and resolve the connective's polysemy problem. We believe that the connective itself is the foundation of discourse relation, and the relation set can be adjusted dynamically according to the application requirements.

Figure 3 shows a three-level set of discourse relations example. In the first level, this set contains four relations of causality, coordination, transition and explanation, which are further clustered into 17 sub-relations in the second level. For example, relation causality contains 6 sub-relations, i.e. cause-result, inference, hypothetical, purpose, condition and background. In the third level, the connectives are under each sub-relation. For example, cause-result relation can be represented by "because", 'therefore' etc. The numbers shown in the parentheses illustrate the distributions of different relations in our corpus. For example, there are 1335 causality relations in the first level, including 686 cause-result relations, 38 inference relations, 70 hypothetical relations, 335 purpose relations, 72 condition relations and 134 background relations.

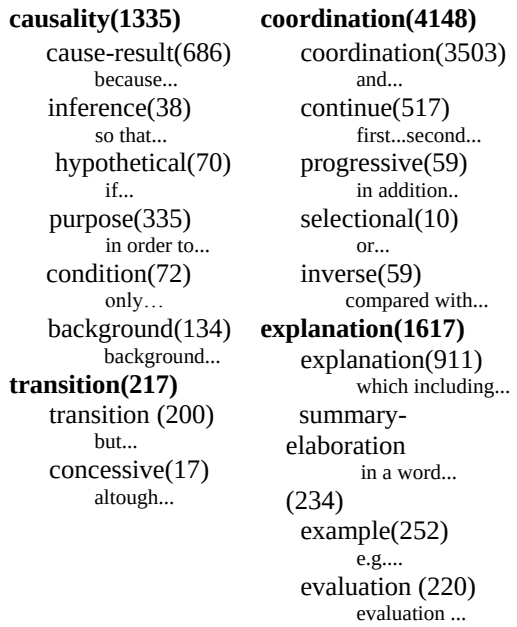


Figure 3: A three-level set of discourse relations

3.5 Nucleus and Satellite

Once discourse units are determined, adjacent spans are linked together via connectives to build a hierarchical structure. As stated above, discourse relations may be either mononuclear or multi-nuclear. A mononuclear relation holds between a nucleus and a satellite unit. Normally, the nucleus usually reflects the intention focus of the discourse and is thus more salient in the discourse structure, while the satellite usually represents supportive information for the nucleus. In comparison, a multi-nuclear relation usually

holds two or more discourse units of equal weight in the discourse structure.

For nucleus determination, we adopt the dependency grammar, and select the unit which can stand for the relationship with other discourse units in a discourse. As shown in Figure 2, on the first level, discourse relation "对此 (to this)" has the latter unit "浦东…法制轨道 (Pudong…as soon as they appear.)" as nucleus and the former unit "浦东…新问题 (Pudong …new problem)" as satellite, since the latter unit agrees with the main purpose of the discourse, which emphasizes some methods for the progress of Pudong. Moreover, since the combination of 4, 5 and 6 has the cause relation with 7, we choose 7 as nucleus because it can stand for the combination of 4, 5, 6 and 7, and has the selection relationship with 3.

4 Chinese Discourse Treebank

Given above the CDT scheme, we choose 500 Xinhua newswire documents from the Chinese Treebank (Xue et al., 2005b) in our Chinese Discourse Treebank (CDTB) annotation. In particular, we annotate one discourse tree for each paragraph.

In this section, we address the key issues with the CDTB annotation, such as annotator training, tagging strategies, corpus quality, along with the statistics of the CDTB corpus.

4.1 Annotator Training

The annotator team consists of a Ph.D. in Chinese linguistics as the supervisor (senior annotator) and four undergraduate students in Chinese linguistics as annotators (two pairs). The annotation is done in four phases. In the first phase, the annotators spend 3 months on learning the principles of CDT and the use of our developed discourse annotation tool. In the second phase, the annotators spend 2 months on independently annotating the same 50 documents (about 260 paraphrases), and another 2 months on cross-checking to resolve the difference and to revise the guidelines. In the third phase, the annotators spend 9 months on annotating the remaining 450 documents. In the final phase, the supervisor spends 3 months carefully proofread all 500 documents.

4.2 Tagging Strategies

In the CDTB annotation, we employ a top-down strategy. That is, we determine the overall level first and then the analysis goes on step by step to the individual EDUs. This strategy is adopted in our annotation tool. The advantages of the top-down strategy are three folds. First, such a strategy can easily grasp the whole discourse structure. This conforms to the global nature of discourse analysis. Second, due to the lack of clear difference between Chinese sentence and phrase structure, such a strategy can largely avoid the error propagation in Chinese EDU segmentation. Since in such a top-down strategy, EDU segmentation becomes an end question, and even if an EDU segmentation error happens, its impact is localized, i.e. with little impact on the whole discourse structure. Our annotation practice shows that such strategy is effective. Third, such a strategy accords with the cognitive of Chinese characteristics, and conforms to the mental process of Chinese discourse understanding (Huang and Liao, 2011). However, we do not exclude the bottom-up strategy. In some cases, on the cognitive psychological process, annotator is combine top-down and bottom-up strategies.

Take Example (3) as an example, an annotator first finds the first level, with the period at the end of sentence 1, and chooses discourse relation (either explicit or implicit), connective, and connective related information (e.g. whether can be added, deleted, and the language sense, etc.), nuclearity etc. Then, the annotator turns to sentence 1 and marks the second comma as level 2 with necessary information annotated, and goes on to sentence 2, recursively, until all EDUs are marked. In this way, a discourse tree with the CDT representation is constructed.

4.3 Quality Assurance

A number of steps are taken to ensure the quality of CDTB. These involve two tasks: checking the validity of the trees and tracking inter-annotator consistency.

4.3.1 Tree validation

We first manually check if a tree has a single root node and compare the tree with the document to check for missing sentence or fragments from the end of text. Then we check the attached information such as connectives, relations and nuclearity in the tree. We also check the tree with a tree traversal program to find the errors undetected by the manual validation process. Finally, all of the trees work successfully.

4.3.2 Consistency

To ensure the quality of CDTB, we adopt the inter-annotator consistency using Agreement and kappa on 60 documents (chtb0041-chtb

0100). Table 1 illustrates the inter-annotator consistency in details.

As shown in Table 1, we measure the agreement of EDU segmentation by determining whether punctuation (all period, comma etc. are considered) is treated as an EDU boundary. It shows that the agreement reaches 91.7% with Cohen's kappa value (Cohen, 1960) 0.91. This justifies the appropriateness of our EDU definition. Explicit or Implicit agreement 94.7% is calculate by the same EDU boundary (intersection) of two annotators. For the same explicit relation, the connective identification agreement is 82.3%, because this is strict measure when two annotators choose the same connective word. If we relax the measure to contain the same word, the agreement can reach 98%. For example, one annotate "也…并(also…and)", and the other annotate "并(and)" is wrong with our strict measure.

	Agreement	Kappa
EDU segmentation	91.7	0.91
Explicit or Implicit	94.7	0.81
Explicit connective identification	82.3	--
Implicit connective insertion	74.6	--
Mononuclear or Multinuclear	80.8	--
Nuclearity	82.4	--
Structure	77.4	--

Table 1: Inter-annotator consistency

It is not surprising that the agreement on implicit connective insertion with the same position and the same connective only reaches 74.6% since for some discourse relations, there may existing several connective alternatives. For example, both "so" and "therefore" can express the same causation relation. If we relax the constraint to the compatible connective, the agreement on implicit connective insertion can reach up to 84.5%.

Finally, it shows that the agreement on overall discourse structure (with the same connectives as non-leaf nodes, the same EDUs as leaf nodes) reaches 77.4%. This justifies the appropriateness of our CDT scheme, given the inherent ambiguity in Chinese discourse structure.

4.4 Corpus Statistics

Currently, the CDTB corpus consists of 500 newswire articles from Chinese Treebank, which are further divided into 2342 paragraphs with a CDT representation for one paragraph.

- For EDUs, CDTB contains 10650 EDUs with an average of 4.5 EDUs per tree. On average, there are 2 EDUs per sentence and 22 Chinese characters per EDU.
- For discourse relations, CDTB contains 7310 relations, of which 1812 are explicit relations (24.8%) and 5498 are implicit relations (75.2%). This indicates that implicit relations occur much more frequently in Chinese than in English, e.g. 75.2% in CDTB (Chinese) vs. ~50% in PDTB (English).
- With the deepest level of 9, most (98.5%) of discourse relations occur in level 1 (2342), level 2(2372), level 3(1532), level 4(712), and level 5(242). It also shows that 3557 (48.7%) relations are mononuclear relations with 2110 nucleus ahead, while the remaining 3754 relations are multi-nuclear. The numbers shown in the parentheses of Figure 3 illustrate the distributions of different relations. In comparison with the top 2 most frequently occurring relations in PDTB (English), i.e. the coordination and explanation relations, there exist 3503 (47.9%) and 911 instances respectively, with regard to the abstract relation set as shown in Figure 3.
- CDTB contains 282 connectives, among which 274 (140 can be deleted) appears as explicit connectives and 44 can be inserted in place of implicit connectives. Table 2 lists the top 10 frequent explicit connectives and implicit connectives.

Explicit connectives		Implicit connectives	
connectives	frequency	connectives	frequency
并(and)	208	因此(so)	368
其中(among them)	154	并(and)	354
也(also)	131	并且(and)	259
而(however)	70	例如(e.g)	140
但(but)	69	来(in order to)	68
还(also)	68	以(in order to)	61
使(so that)	56	然后(then)	55
以(in order to)	52	其中(among them)	48
为(in order to)	49	而(while)	47
同时(meanwhile)	46	因为(because)	32

Table 2: The most frequent connectives in CDTB

5 Comparison with other Discourse Banks

Table 3 compares the difference of CDTB with RST-DT and PDTB from various perspectives, such as EDU, connective, relation, structure and nuclearity.

	RST-DB	PDTB	CDTB
EDU	Clear defined; start of combination; one relation has two or more EDUs	Predicate-argument view; one relation has two arguments	Clear defined from three aspects; end of top-down segmentation; one relation has two or more EDUs
Connective	--	Mark explicit connectives and insert implicit connectives	Mark whether an connective can be deleted without changing the rhetorical relation; insert implicit connective with good intuition and bad intuition differentiated
Relation	Abstract set of relation types; annotate the relation types	Abstract set of relation types; annotate connective and relation type	Represent relation by connective; annotate connective and it's attribute; mapping of connective to the set of discourse relations in a later stage
Structure	Complete tree	Partial tree, deduced by connective and it's argument	Complete tree; top-down segmentation; structure can be represented by the connective hierarchy
Nuclearity	Determined by certain rhetorical relation	--	Determined by the global meaning of a discourse

Table 3: The comparison of RST-DT, PDTB and CDTB

6 Preliminary Experimentation

In order to evaluate the computability of CDTB, we give the experimental results on EDU recognition, which is crucial in discourse parsing. After excluding sentence end punctuations (such as period, question mark, and exclamatory mark), which are certainly EDU boundaries, there remains 7625 punctuations as EDU boundaries (positive instances) and 4876 punctuations as non-EDU boundaries (negative instances). With various features as adopted in Xue and Yang (2011) and Li et al. (2013b), Table 4 shows the performance of EDU recognition on the CDTB corpus with 10-fold cross validation.

Classifier	Gold standard parse			Automatic parse		
	Accuracy	F1(+)	F1(-)	Accuracy	F1(+)	F1(-)
MaxEnt	**90.6**	**91.1**	**90.5**	**89.0**	**90.3**	**87.2**
C45	90.2	90.5	90.1	88.7	90.0	87.7
NiveBayes	90.2	89.9	88.9	88.0	89.0	86.9

Table 4: Performance of EDUs recognition

As shown in Table 4, MaxEnt performs best, with accuracy up to 90.6% on gold standard parse tree, close to human agreement of 91.7%, and with accuracy up to 89% on automatic parse tree. This suggests the appropriateness of our definition of clause as EDU. Table 4 also gives the performance on both positive and negative instances. It shows better F1-measure on recognizing positive instances than negative instances.

7 Conclusions

In this paper, we propose a Connective-driven Dependency Tree (CDT) structure as a representation scheme for Chinese discourse structure. CDT takes advantage of both RST and PDTB, and well adapts to the special characteristics of Chinese discourse. In particular, we describe CDT in detail from various perspectives, such as EDU, connective, structure, relation and nuclearity. Given the CDT scheme, we annotate 500 documents in a top-down segmentation process to keep consistent with Chinese native's cognitive habit. Evaluation of the CDTB corpus on EDU recognition justifies the appropriateness of the CDT scheme to Chinese discourse structure and the usefulness of our CDTB corpus.

In the future work, we will focus on enlarging the scale of the corpus annotation and developing a complete Chinese discourse parser.

Acknowledgments

This research is supported by the Project 2012AA011102 under the National 863 High-Tech Program of China, by the National Natural Science Foundation of China, No.61331011, No.61273320.

The contact author of this paper, according to the meaning given to this role by Soochow University, is Guodong Zhou. The complete corpus is available for research purpose upon request.

Reference

Zheng Cao.1984. *Primary exploration on sentence group*. Zhejiang Education Press, Hangzhou,CN(in Chinese).

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. *Building a Discourse-tagged Corpus in the Framework of Rhetorical Structure Theory*. Springer Netherlands.

LiPing Chen. 2006. *English and Chinese discourse structure dimension theory and practice*. Ph.D. thesis, Shanghai international studies university doctoral dissertation.

Liping Chen. 2008. Chinese text structure annotation theory support，*Journal of Nanjing university of aeronautics and astronautics*, 10(3):69-71(in Chinese).

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20 (1): 37–46.

Wenhe Feng and Donghong Ji. 2011. Parallel structure analysis of the coordination structure and the controller status of connective. *Linguistic Sciences*, 2:168-181(in Chinese).

David G Hays. 1964. Dependency theory: formalism and some observations. *Language*, 40(4):511-525.

Hen-Hsen Huang and Hsin-Hsi Chen. 2011. Chinese Discourse Relation Recognition. In *Proceedings of 5th International Joint Conference on Natural Language Process*, pages 1442–1446, Chiang Mai, Thailand, November 2011.

Borong Huang and Xudong Liao. 2011. *Morden Chinese* (volume two, updated 5th edition). Higher Education Press. Beijing,CN(in Chinese).

Yancui Li, Wenhe Feng, and Guodong Zhou. 2013a. Elementary discourse unit in Chinese discourse structure analysis. In *Chinese Lexical Semantics*, pages 186-198, Wuhan, China, Springer Berlin Heidelberg.

Yancui Li, Wenhe Feng, and Guodong Zhou et al. 2013b. Research of Chinese Clause Identification Based on Comma. *Acta Scientiarum Naturalium Universitatis Pekinensis*, 49(1):7-14 (in Chinese with English abstract).

Ziheng Lin, Min-Yan Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 343–351, Singapore, 6-7 August 2009.

Daniel Marcu. 2000. *The theory and practice of discourse parsing and summarization*. MIT Press.

William Mann and Sandra Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243-281.

Emily Pitler, Annie Louis, and Ani Nenkova. Automatic sense prediction for implicit discourse relations in text. 2009. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 683–691, Suntec, Singapore, 2-7 August 2009.

Rashmi Prasad, Nikhil Dinesh, and Lee et al. 2008. The Penn Discourse TreeBank 2.0. In *Processdings of Sixth International Conference on Language Resources and Evaluation (LREC)*, pages 2961-2968, Marrakech, Morocco.

Susan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2007. Discourse-based answering of why-questions. *Traitement Automatique des Langues, special issue on Computational Approaches to Discourse and Document Processing*, 47(2):21-41.

Wenge Wang. 2010. The Current Research Situation of the Clause in Modern Chinese. *Chinese Language Learning*, (1): 67–76(in Chinese).

Fuyi Xing. 2003. *Research of Chinese complex sentence*. The Commercial Press, Beijing, CN (in Chinese).

Nianwen Xue. 2005a. Annotating the Discourse Connectives in the Chinese Treebank. In *Proceedings of the ACL Workshop on Frontiers in Corpus Annotation*, pages 84-91, Ann Arbor, Michigan.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005b. The Penn Chinese Treebank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering*, 11(2): 207-238.

Nianwen Xue and Yaqin Yang. 2011. Chinese sentence segmentation as comma classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics:Human Language Technologies*, pages 631-635, Portland, Oregon,USA, June 2011.

Ming Yue. 2008. Rhetorical Structure Annotation of Chinese News Commentaries. *Journal of Chinese Information Processing*, 22(4): 19-23(in Chinese with English abstract).

Yuping Zhou and Nianwen Xue. 2012. PDTB-style discourse annotation of Chinese text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 69-77, Jeju, Republic of Korea, 8-14 July 2012.

Prune-and-Score: Learning for Greedy Coreference Resolution

Chao Ma, Janardhan Rao Doppa†, J. Walker Orr, Prashanth Mannem
Xiaoli Fern, Tom Dietterich and **Prasad Tadepalli**
School of Electrical Engineering and Computer Science, Oregon State University
{machao,orr,mannemp,xfern,tgd,tadepall}@eecs.oregonstate.edu
† School of Electrical Engineering and Computer Science, Washington State University
jana@eecs.wsu.edu

Abstract

We propose a novel search-based approach for greedy coreference resolution, where the mentions are processed in order and added to previous coreference clusters. Our method is distinguished by the use of two functions to make each coreference decision: a *pruning* function that prunes bad coreference decisions from further consideration, and a *scoring* function that then selects the best among the remaining decisions. Our framework reduces learning of these functions to rank learning, which helps leverage powerful off-the-shelf rank-learners. We show that our *Prune-and-Score* approach is superior to using a single scoring function to make both decisions and outperforms several state-of-the-art approaches on multiple benchmark corpora including OntoNotes.

1 Introduction

Coreference resolution is the task of clustering a set of mentions in the text such that all mentions in the same cluster refer to the same entity. It is one of the first stages in deep language understanding and has a big potential impact on the rest of the stages. Several of the state-of-the-art approaches learn a scoring function defined over mention pair, cluster-mention or cluster-cluster pair to guide the coreference decision-making process (Daumé III, 2006; Bengtson and Roth, 2008; Rahman and Ng, 2011b; Stoyanov and Eisner, 2012; Chang et al., 2013; Durrett et al., 2013; Durrett and Klein, 2013). One common and persistent problem with these approaches is that the scoring function has to make all the coreference decisions, which leads to a highly non-realizable learning problem.

Inspired by the recent success of the $\mathcal{HC}$-Search Framework (Doppa et al., 2014a) for studying a variety of structured prediction problems (Lam et al., 2013; Doppa et al., 2014c), we study a novel approach for search-based coreference resolution called *Prune-and-Score*. $\mathcal{HC}$-Search is a divide-and-conquer solution that learns multiple components with pre-defined roles, and each of them contribute towards the overall goal by making the role of the other components easier. The $\mathcal{HC}$-Search framework operates in the space of complete outputs, and relies on the loss function which is only defined on the complete outputs to drive its learning. Unfortunately, this method does not work for incremental coreference resolution since the search space for coreference resolution consists of partial outputs, i.e., a set of mentions only some of which have been clustered so far.

We develop an alternative framework to $\mathcal{HC}$-Search that allows us to effectively learn from partial output spaces and apply it to greedy coreference resolution. The key idea of our work is to address the problem of non-realizability of the scoring function by learning two different functions: 1) a *pruning function* to prune most of the bad decisions, and 2) a *scoring function* to pick the best decision among those that are remaining. Our Prune-and-Score approach is a particular instantiation of the general idea of learning nearly-sound constraints for pruning, and leveraging the learned constraints to learn improved heuristic functions for guiding the search. The pruning constraints can take different forms (e.g., classifiers, decision-list, or ranking functions) depending on the search architecture. Therefore, other coreference resolution systems (Chang et al., 2013; Durrett and Klein, 2013; Björkelund and Kuhn, 2014) can also benefit from this idea. While our basic idea of two-level selection might appear similar to the coarse-to-fine inference architectures (Felzenszwalb and McAllester, 2007; Weiss and Taskar, 2010), the details differ significantly. Importantly, our pruning and scoring functions operate sequentially at

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2115–2126,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

each greedy search step, whereas in the cascades approach, the second level function makes its prediction only when the first level decision-making is done.

Summary of Contributions. The main contributions of our work are as follows. First, we motivate and introduce the *Prune-and-Score* approach to search-based coreference resolution. Second, we identify a decomposition of the overall loss of the Prune-and-Score approach into the pruning loss and the scoring loss, and reduce the problem of learning these two functions to rank learning, which allows us to leverage powerful and efficient off-the-shelf rank learners. Third, we evaluate our approach on OntoNotes, ACE, and MUC data, and show that it compares favorably to several state-of-the-art approaches as well as a greedy search-based approach that uses a single scoring function.

The remainder of the paper proceeds as follows. In Section 2, we dicuss the related work. We introduce our problem setup in Section 3 and then describe our *Prune-and-Score* approach in Section 4. We explain our approaches for learning the pruning and scoring functions in Section 5. Section 6 presents our experimental results followed by the conclusions in Section 7.

2 Related Work

The work on learning-based coreference resolution can be broadly classified into three types. First, the *pair-wise classifier* approaches learn a classifier on mention pairs (edges) (Soon et al., 2001; Ng and Cardie, 2002; Bengtson and Roth, 2008), and perform some form of approximate decoding or post-processing using the pair-wise scores to make predictions. However, the pair-wise classifier approach suffers from several drawbacks including class imbalance (fewer positive edges compared to negative edges) and not being able to leverage the global structure (instead making independent local decisions).

Second, the *global* approaches such as Structured SVMs and Conditional Random Fields (CRFs) learn a cost function to score a potential clustering output for a given input set of mentions (Mccallum and Wellner, 2003; Finley and Joachims, 2005; Culotta et al., 2007; Yu and Joachims, 2009; Haghighi and Klein, 2010; Wick et al., 2011; Wick et al., 2012; Fernandes et al., 2012). These methods address some of the problems with pair-wise classifiers, however, they suffer from the intractability of "Argmin" inference (finding the least cost clustering output among exponential possibilities) that is encountered during both training and testing. As a result, they resort to approximate inference algorithms (e.g., MCMC, loopy belief propagation), which can suffer from local optima.

Third, the *incremental* approaches construct the clustering output incrementally by processing the mentions in some order (Daumé III, 2006; Denis and Baldridge, 2008; Rahman and Ng, 2011b; Stoyanov and Eisner, 2012; Chang et al., 2013; Durrett et al., 2013; Durrett and Klein, 2013). These methods learn a scoring function to guide the decision-making process and differ in the form of the scoring function (e.g., mention pair, cluster-mention or cluster-cluster pair) and how it is being learned. They have shown great success and are very efficient. Indeed, several of the approaches that have achieved state-of-the-art results on OntoNotes fall under this category (Chang et al., 2013; Durrett et al., 2013; Durrett and Klein, 2013; Björkelund and Kuhn, 2014). However, their efficiency requirement leads to a highly non-realizable learning problem. Our Prune-and-Score approach is complementary to these methods, as we show that having a pruning function (or a set of learned pruning rules) makes the learning problem easier and can improve over the performance of scoring-only approaches. Also, the models in (Chang et al., 2013; Durrett et al., 2013) try to leverage cluster-level information implicitly (via a latent antecedents) from mention-pair features, whereas our model explicitly leverages the cluster level information.

Coreference resolution systems can benefit by incorporating the world knowledge including rules, constraints, and additional information from external knowledge bases (Lee et al., 2013; Rahman and Ng, 2011a; Ratinov and Roth, 2012; Chang et al., 2013; Zheng et al., 2013; Hajishirzi et al., 2013). Our work is orthogonal to this line of work, but domain constraints and rules can be incorporated into our model as done in (Chang et al., 2013).

3 Problem Setup

Coreference resolution is a structured prediction problem where the set of mentions $m_1, m_2, \cdots, m_D$ extracted from a document cor-

reponds to a structured input x and the structured output y corresponds to a partition of the mentions into a set of clusters $C_1, C_2, \cdots, C_k$. Each mention m_i belongs to exactly one of the clusters C_j. We are provided with a training set of input-output pairs drawn from an unknown distribution $\mathcal{D}$, and the goal is to return a function/predictor from inputs to outputs. The learned predictor is evaluated against a non-negative *loss function* $L : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \mapsto \Re^+$, $L(x, y', y)$ is the loss associated with predicting incorrect output y' for input x when the true output is y (e.g., B-Cubed Score).

In this work, we formulate the coreference resolution problem in a search-based framework. There are three key elements in this framework: 1) the *Search space* $\mathcal{S}_p$ whose states correspond to partial clustering outputs; 2) the *Action pruning function* $\mathcal{F}_{prune}$ that is used to prune irrelevant actions at each state; and 3) the *Action scoring function* $\mathcal{F}_{score}$ that is used to construct a complete clustering output by selecting actions from those that are left after pruning. $\mathcal{S}_p$ is a 3-tuple $\langle I, A, T \rangle$, where I is the initial state function, A gives the set of possible actions in a given state, and T is a predicate which is true for terminal states. In our case, $s_0 = I(x)$ corresponds to a state where every mention is unresolved, and $A(s_i)$ consists of actions to place the next mention m_{i+1} in each cluster in s_i or a NEW action which creates a new cluster for it. Terminal nodes correspond to states with all mentions resolved.

We focus on greedy search. The decision process for constructing an output corresponds to selecting a sequence of actions leading from the initial state to a terminal state using both $\mathcal{F}_{prune}$ and $\mathcal{F}_{score}$, which are parameterized functions over state-action pairs ($F_{prune}(\phi_1(s, a)) \in \Re$ and $F_{score}(\phi_2(s, a)) \in \Re$), where ϕ_1 and ϕ_2 stand for feature functions. We want to learn the parameters of both $\mathcal{F}_{prune}$ and $\mathcal{F}_{score}$ such that the predicted outputs on unseen inputs have low expected loss.

4 Greedy Prune-and-Score Approach

Our greedy *Prune-and-Score* approach for coreference resolution is parameterized by a pruning function $\mathcal{F}_{prune} : \mathcal{S} \times \mathcal{A} \mapsto \Re$, a scoring function $\mathcal{F}_{score} : \mathcal{S} \times \mathcal{A} \mapsto \Re$, and a pruning parameter $b \in [1, A_{max}]$, where A_{max} is the maximum number of actions at any state $s \in \mathcal{S}$. Given a set of input mentions $m_1, m_2, \cdots, m_D$ extracted from a document (input x), and a pruning parameter b, our Prune-and-Score approach makes predictions as follows. The search starts at the initial state $s_0 = I(x)$ (see Algorithm 1). At each non-terminal state s, the pruning function $\mathcal{F}_{prune}$ retains only the top b actions (A') from $A(s)$ (Step 3), and the scoring function $\mathcal{F}_{score}$ picks the best scoring action $a_p \in A'$ (Step 4) to reach the next state. When a terminal state is reached its contents are returned as the prediction. Figure 1 illustrates the decision-making process of our Prune-and-Score approach for an example state.

Algorithm 1 Greedy Prune-and-Score Resolver

Input: x = set of mentions $m_1, m_2, \cdots, m_D$ from a document D, $\langle I, A, T \rangle$ = Search space definition, $\mathcal{F}_{prune}$ = learned pruning function, b = pruning parameter, $\mathcal{F}_{score}$ = learned scoring function

1: $s \leftarrow I(x)$ *// initial state*
2: **while not** $T(s)$ **do**
3: $A' \leftarrow$ Top b actions from $A(s)$ according to $\mathcal{F}_{prune}$ *// prune*
4: $a_p \leftarrow \arg\max_{a \in A'} \mathcal{F}_{score}(s, a)$ *// score*
5: $s \leftarrow$ Apply a_p on s
6: **end while**
7: **return** coreference output corresponding to s

We now formalize the learning objective of our Prune-and-Score approach. Let $\hat{y}$ be the predicted coreference output for a coreference input-output pair (x, y^*). The expected loss of the greedy Prune-and-Score approach $\mathcal{E}(\mathcal{F}_{prune}, \mathcal{F}_{score})$ for a given pruning function $\mathcal{F}_{prune}$ and scoring function $\mathcal{F}_{score}$ can be defined as follows.

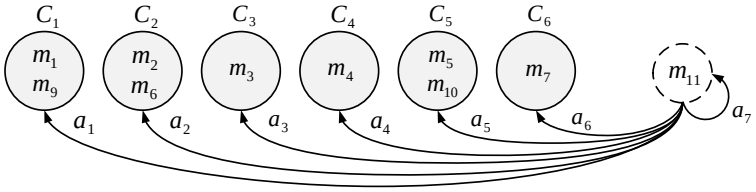
$$\mathcal{E}(\mathcal{F}_{prune}, \mathcal{F}_{score}) = \mathbb{E}_{(x,y^*)\sim\mathcal{D}}\ L\left(x, \hat{y}, y^*\right)$$

Our goal is to learn an optimal pair of pruning and scoring functions $\left(\mathcal{F}^o_{prune}, \mathcal{F}^o_{score}\right)$ that minimizes the expected loss of the Prune-and-Score approach. The behavior of our Prune-and-Score approach depends on the pruning parameter b, which dictates the workload of pruning and scoring functions. For small values of b (aggressive pruning), pruning function learning may be harder, but scoring function learning will be easier. Similarly, for large values of b (conservative pruning), scoring function learning becomes hard, but pruning function learning is easy. Therefore, we would expect beneficial behavior if pruning function can aggressively prune (small values of b) with little loss in accuracy. It is interesting to note that our Prune-and-Score approach degenerates to existing incremental approaches that use only the scoring function for search (Daumé III, 2006; Rahman and

(a) Text with input set of mentions

Ramallah (West Bank $_2$)$_1$ 10-15 (AFP$_3$) - Eyewitnesses$_4$ reported that Palestinians$_5$ demonstrated today Sunday in the West Bank$_6$ against the Sharm el-Sheikh$_7$ summit to be held in Egypt$_8$ tomorrow Monday. In Ramallah$_9$, around 500 people$_{10}$ took to **the town**$_{11}$'s streets chanting slogans denouncing the summit ...

(b) Illustration of Prune-and-Score approach



State: $s = \{C_1, C_2, C_3, C_4, C_5, C_6\}$ Actions: $A(s) = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$

Pruning step: a_2 (2.5) a_1 (2.2) a_7 (1.9) a_5 (1.5) a_6 (1.4) a_3 (0.7) a_4 (0.4) ← $\mathcal{F}_{prune}$ values

$b = 3$

$A'(s) = \{a_2, a_1, a_7\}$

Scoring step: a_1 (4.5) a_2 (3.1) a_7 (2.6) ← $\mathcal{F}_{score}$ values

Decision: a_1 is the best action for state s

Figure 1: Illustration of Prune-and-Score approach. (a) Text with input set of mentions. Mentions are highlighted and numbered. (b) Illustration of decision-making process for mention m_{11}. The partial clustering output corresponding to the current state s consists of six clusters denoted by $C_1, C_2, \cdots, C_6$. Highlighted circles correspond to the clusters. Edges from mention m_{11} to each of the six clusters and to itself stand for the set of possible actions $A(s)$ in state s, and are denoted by $a_1, a_2, \cdots, a_7$. The pruning function $\mathcal{F}_{prune}$ scores all the actions in $A(s)$ and only keeps the top 3 actions $A' = \{a_2, a_1, a_7\}$ as specified by the pruning parameter b. The scoring function picks the best scoring action $a_1 \in A'$ as the final decision, and mention m_{11} is merged with cluster C_1.

Ng, 2011b) when $b = \infty$. Additionally, for $b = 1$, our pruning function coincides with the scoring function.

Analysis of Representational Power. The following proposition formalizes the intuition that two functions are strictly better than one in expressive power. See Appendix for the proof.

Proposition 1. *Let $\mathcal{F}_{prune}$ and $\mathcal{F}_{score}$ be functions from the same function space. Then for all learning problems,* $\min_{\mathcal{F}_{score}} \mathcal{E}(\mathcal{F}_{score}, \mathcal{F}_{score}) \geq \min_{(\mathcal{F}_{prune}, \mathcal{F}_{score})} \mathcal{E}(\mathcal{F}_{prune}, \mathcal{F}_{score})$. *Moreover there exist learning problems for which* $\min_{\mathcal{F}_{score}} \mathcal{E}(\mathcal{F}_{score}, \mathcal{F}_{score})$ *can be arbitrarily worse than* $\min_{(\mathcal{F}_{prune}, \mathcal{F}_{score})} \mathcal{E}(\mathcal{F}_{prune}, \mathcal{F}_{score})$.

5 Learning Algorithms

In general, learning the optimal $\left(\mathcal{F}^o_{prune}, \mathcal{F}^o_{score}\right)$ pair can be intractable due to their potential interdependence. Specifically, when learning $\mathcal{F}_{prune}$ in the worst case there can be ambiguity about which of the non-optimal actions to retain, and for only some of those an effective $\mathcal{F}_{score}$ can be found. However, we observe a loss decomposition in terms of the individual losses due to $\mathcal{F}_{prune}$ and $\mathcal{F}_{score}$, and develop a stage-wise learning approach that first learns $\mathcal{F}_{prune}$ and then learns a corresponding $\mathcal{F}_{score}$.

5.1 Loss Decomposition

The overall loss of the *Prune-and-Score* approach $\mathcal{E}\left(\mathcal{F}_{prune}, \mathcal{F}_{score}\right)$ can be decomposed into *pruning loss* ϵ_{prune}, the loss due to $\mathcal{F}_{prune}$ not being able to retain the *optimal* terminal state in the search space; and *scoring loss* $\epsilon_{score|\mathcal{F}_{prune}}$, the additional loss due to $\mathcal{F}_{score}$ not guiding the greedy search to the best terminal state after pruning using $\mathcal{F}_{prune}$. Below, we will define these losses more formally.

Pruning Loss is defined as the expected loss of the *Prune-and-Score* approach when we perform greedy search with $\mathcal{F}_{prune}$ and $\mathcal{F}^*_{score}$, the optimal scoring function. A scoring function is said to be *optimal* if at every state s in the search space

$\mathcal{S}_p$, and for any set of remaining actions $A(s)$, it can score each action $a \in A(s)$ such that greedy search can reach the best terminal state (as evaluated by task loss function L) that is reachable from s through $A(s)$. Unfortunately, computing the optimal scoring function is highly intractable for the non-decomposable loss functions that are employed in coreference resolution (e.g., B-Cubed F1). The main difficulty is that the decision at any one state has interdependencies with future decisions (see Section 5.5 in (Daumé III, 2006) for more details). So we need to resort to some form of *approximate* optimal scoring function that exhibits the intended behavior. This is very similar to the dynamic oracle concept developed for dependency parsing (Goldberg and Nivre, 2013).

Let y^*_{prune} be the coreference output corresponding to the terminal state reached from input x by *Prune-and-Score* approach when performing search using $\mathcal{F}_{prune}$ and $\mathcal{F}^*_{score}$. Then the pruning loss can be expressed as follows.

$$\epsilon_{prune} = \mathbb{E}_{(x,y^*)\sim\mathcal{D}}\, L\left(x, y^*_{prune}, y^*\right)$$

Scoring Loss is defined as the additional loss due to $\mathcal{F}_{score}$ not guiding the greedy search to the best terminal state reachable via the pruning function $\mathcal{F}_{score}$ (i.e., y^*_{prune}). Let $\hat{y}$ be the coreference output corresponding to the terminal state reached by *Prune-and-Score* approach by performing search with $\mathcal{F}_{prune}$ and $\mathcal{F}_{score}$ for an input x. Then the scoring loss can be expressed as follows:

$$\epsilon_{score|\mathcal{F}_{prune}} = \mathbb{E}_{(x,y^*)\sim\mathcal{D}}\, L\left(x, \hat{y}, y^*\right) - L\left(x, y^*_{prune}, y^*\right)$$

The overall loss decomposition of our *Prune-and-Score* approach can be expressed as follows.

$$\mathcal{E}\left(\mathcal{F}_{prune}, \mathcal{F}_{score}\right) = \underbrace{\mathbb{E}_{(x,y^*)\sim\mathcal{D}}\, L\left(x, y^*_{prune}, y^*\right)}_{\epsilon_{\mathbf{prune}}} + \underbrace{\mathbb{E}_{(x,y^*)\sim\mathcal{D}}\, L\left(x, \hat{y}, y^*\right) - L\left(x, y^*_{prune}, y^*\right)}_{\epsilon_{\mathbf{score}|\mathcal{F}_{\mathbf{prune}}}}$$

5.2 Stage-wise Learning

The loss decomposition motivates a learning approach that targets minimizing the errors of pruning and scoring functions independently. In particular, we optimize the overall loss of the *Prune-and-Score* approach in a stage-wise manner. We first train a pruning function $\hat{\mathcal{F}}_{prune}$ to optimize the pruning loss component ϵ_{prune} and then train a scoring function $\hat{\mathcal{F}}_{score}$ to optimize the scoring loss $\epsilon_{score|\hat{\mathcal{F}}_{prune}}$ conditioned on $\hat{\mathcal{F}}_{prune}$.

$$\hat{\mathcal{F}}_{prune} \approx \arg\min_{\mathcal{F}_{prune}\in\mathbf{F_p}} \quad \epsilon_{prune}$$
$$\hat{\mathcal{F}}_{score} \approx \arg\min_{\mathcal{F}_{score}\in\mathbf{F_s}} \quad \epsilon_{score|\hat{\mathcal{F}}_{prune}}$$

Note that this approach is myopic in the sense that $\hat{\mathcal{F}}_{prune}$ is learned without considering the implications for learning $\hat{\mathcal{F}}_{score}$. Below, we first describe our approach for pruning function learning, and then explain our scoring function learning algorithm.

5.3 Pruning Function Learning

In our greedy *Prune-and-Score* approach, the role of the pruning function $\mathcal{F}_{prune}$ is to prune away irrelevant actions (as specified by the pruning parameter b) at each search step. More specifically, we want $\mathcal{F}_{prune}$ to score actions $A(s)$ at each state s such that the optimal action $a^* \in A(s)$ is ranked within the top b actions to minimize ϵ_{prune}. For this, we assume that for any training input-output pair (x, y^*) there exists a unique action sequence, or *solution path* (initial state to terminal state), for producing y^* from x. More formally, let $(s^*_0, a^*_0), (s^*_1, a^*_1), \cdots, (s^*_D, \varnothing)$ correspond to the sequence of state-action pairs along this solution path, where s^*_0 is the initial state and s^*_D is the terminal state. The goal is to learn the parameters of $\mathcal{F}_{prune}$ such that at each state s^*_i, $a^*_i \in A(s^*_i)$ is ranked among the top b actions.

While we can employ an *online-LaSO* style approach (III and Marcu, 2005; Xu et al., 2009) to learn the parameters of the pruning function, it is quite inefficient, as it must regenerate the same search trajectory again and again until it learns to make the right decision. Additionally, this approach limits applicability of the off-the-shelf learners to learn the parameters of $\mathcal{F}_{prune}$. To overcome these drawbacks, we apply offline training.

Reduction to Rank Learning. We reduce the pruning function learning to a rank learning problem. This allows us to leverage powerful and efficient off-the-shelf rank-learners (Liu, 2009). The reduction is as follows. At each state s^*_i on the solution path of a training example (x, y^*), we create an example by labeling optimal action $a^*_i \in A(s^*_i)$ as the only relevant action, and then try to learn

a ranking function that can rank actions such that the relevant action a_i^* is in the top b actions, where b is the input pruning paramter. In other words, we have a rank learning problem, where the learner's goal is to optimize the *Precision at Top-b*. The training approach creates such an example for each state s in the solution path. The set of aggregate imitation examples collected over all the training data is then given to a rank learner (e.g., LambdaMART (Burges, 2010)) to learn the parameters of $\mathcal{F}_{prune}$ by optimizing the *Precision at Top-b* loss. See appendix for the pseudocode.

If we can learn a function $\mathcal{F}_{prune}$ that is consistent with these imitation examples, then the learned pruning function is guaranteed to keep the solution path within the pruned space for all the training examples. We can also employ more advanced imitation learning algorithms including DAgger (Ross et al., 2011) and SEARN (Hal Daumé III et al., 2009) if we are provided with an (approximate) optimal scoring function $\mathcal{F}^*_{score}$ that can pick optimal actions at states that are not in the solution path (i.e., *off-trajectory* states).

5.4 Scoring Function Learning

Given a learned pruning function $\mathcal{F}_{prune}$, we want to learn a scoring function that can pick the best action from the b actions that remain after pruning at each state. We formulate this problem in the framework of *imitation learning* (Khardon, 1999). More formally, let $(\hat{s}_0, a_0^*), (\hat{s}_1, a_1^*), \cdots, (\hat{s}_D^*, \varnothing)$ correspond to the sequence of state-action pairs along the greedy trajectory obtained by running the Prune-and-Score approach with $\mathcal{F}_{prune}$ and $\mathcal{F}^*_{score}$, the optimal scoring function, on a training example (x, y^*), where $\hat{s}_D^*$ is the best terminal state in the pruned space. The goal of our imitation training approach is to learn the parameters of $\mathcal{F}_{score}$ such that at each state $\hat{s}_i$, $a_i^* \in A'$ is ranked higher than all other actions in A', where $A' \subseteq A(\hat{s}_i)$ is the set of b actions that remain after pruning.

It is important to note that the distribution of states in the pruned space due to $\mathcal{F}_{prune}$ on the testing data may be somewhat different from those on training data. Therefore, we train our scoring function via cross-validation by training the scoring function on heldout data that was not used to train the pruning function. This methodology is commonly employed in Re-Ranking and Stacking approaches (Collins, 2000; Cohen and de Carvalho, 2005).

Our scoring function learning procedure uses cross validation and consists of the following four steps. First, we divide the training data $\mathcal{D}$ into k folds. Second, we learn k different pruners, where each pruning function $\mathcal{F}^i_{prune}$ is learned using the data from all the folds excluding the i^{th} fold. Third, we generate ranking examples for scoring function learning as described above using each pruning function $\mathcal{F}^i_{prune}$ on the data it was not trained on. Finally, we give the aggregate set of ranking examples $\mathcal{R}$ to a rank learner (e.g., SVM-Rank or LambdaMART) to learn the scoring function $\mathcal{F}_{score}$. See appendix for the pseudocode.

Approximate Optimal Scoring Function. If the learned pruning function is not consistent with the training data, we will encounter states $\hat{s}_i$ that are not on the target path, and we will need some supervision for learning in those cases. As discussed before in Section 5.1, computing an optimal scoring function $\mathcal{F}^*_{score}$ is intractable for combinatorial loss functions that are used for coreference resolution. So we employ an approximate function from existing work that is amenable to evaluate partial outputs (Daumé III, 2006). It is a variant of the ACE scoring function that removes the bipartite matching step from the ACE metric. Moreover this score is computed only on the partial coreference output corresponding to the "after state" s' resulting from taking action a in state s, i.e., $\mathcal{F}^*_{score}(s, a) = \mathcal{F}^*_{score}(s')$. To further simplify the computation, we give uniform weight to the three types of costs: 1) Credit for correct linking, 2) Penalty for incorrect linking, and 3) Penalty for missing links. Intuitively, this is similar to the correct-link count computed only on a subgraph. We direct the reader to (Daumé III, 2006) for more details (see Section 5.5).

6 Experiments and Results

In this section, we evaluate our greedy Prune-and-Score approach on three benchmark corpora – OntoNotes 5.0 (Pradhan et al., 2012), ACE 2004 (NIST, 2004), and MUC6 (MUC6, 1995) – and compare it against the state-of-the-art approaches for coreference resolution. For OntoNotes data, we report the results on both gold mentions and predicted mentions. We also report the results on gold mentions for ACE 2004 and MUC6 data.

6.1 Experimental Setup

Datasets. For OntoNotes corpus, we employ the official split for training, validation, and testing. There are 2802 documents in the training set; 343 documents in the validation set; and 345 documents in the testing set. The ACE 2004 corpus contains 443 documents. We follow the (Culotta et al., 2007; Bengtson and Roth, 2008) split in our experiments by employing 268 documents for training, 68 documents for validation, and 107 documents (ACE2004-CULOTTA-TEST) for testing. We also evaluate our system on the 128 newswire documents in ACE 2004 corpus for a fair comparison with the state-of-the-art. The MUC6 corpus containts 255 documents. We employ the official test set of 30 documents (MUC6-TEST) for testing purposes. From the remaining 225 documents, which includes 195 official training documents and 30 dry-run test documents, we randomly pick 30 documents for validation, and use the remaining ones for training.

Evaluation Metrics. We compute three most popular performance metrics for coreference resolution: MUC (Vilain et al., 1995), B-Cubed (Bagga and Baldwin, 1998), and Entity-based CEAF (CEAF$_{\phi 4}$) (Luo, 2005). As it is commonly done in CoNLL shared tasks (Pradhan et al., 2012), we employ the average F1 score (CoNLL F1) of these three metrics for comparison purposes. We evaluate all the results using the updated version[1] (7.0) of the coreference scorer.

Features. We built[2] our coreference resolver based on the Easy-first coreference system (Stoyanov and Eisner, 2012), which is derived from the Reconcile system (Stoyanov et al., 2010). We essentially employ the same features as in the Easy-first system. However, we provide some high-level details that are necessary for subsequent discussion. Recall that our features $\phi(s, a)$ for both pruning and scoring functions are defined over state-action pairs, where each state s consists of a set of clusters and an action a corresponds to merging an unprocessed mention m with a cluster C in state s or create one for itself. Therefore, $\phi(s, a)$ defines features over cluster-mention pairs (C, m). Our feature vector consists of three parts: a) mention pair features; b) entity pair features; and c) a single indicator feature to represent NEW action (i.e., mention m starts its own cluster). For mention pair features, we average the pair-wise features over all links between m and every mention m_c in cluster C (often referred to as *average-link*). Note that, we cannot employ the *best-link* feature representation because we perform offline training and do not have weights for scoring the links. For entity pair features, we treat mention m as a singleton entity and compute features by pairing it with the entity represented by cluster C (exactly as in the Easy-first system). The indicator feature will be 1 for the NEW action and 0 for all other actions.We have a total of 140 features: 90 mention pair features; 49 entity pair features; and one NEW indicator feature. We believe that our approach can benefit from employing features of the mention for the NEW action (Rahman and Ng, 2011b; Durrett and Klein, 2013). However, we were constrained by the Reconcile system and could not leverage these features for the NEW action.

Base Rank-Learner. Our pruning and scoring function learning algorithms need a base rank-learner. We employ LambdaMART (Burges, 2010), a state-of-the art rank learner from the RankLib[3] library. LambdaMART is a variant of boosted regression trees. We use a learning rate of 0.1, specify the maximum number of boosting iterations (or trees) as 1000 noting that its actual value is automatically decided based on the validation set, and tune the number of leaves per tree based on the validation data. Once we fix the hyper-parameters of LambdaMART, we train the final model on all of the training data. LambdaMART uses an internal train/validation split of the input ranking examples to decide when to stop the boosting iterations. We fixed this ratio to 0.8 noting that the performance is not sensitive to this parameter. For scoring function learning, we used 5 folds for the cross-validation training.

Pruning Parameter b. The hyper-parameter b controls the amount of pruning in our Prune-and-Score approach. We perform experiments with different values of b and pick the best value based on the performance on the validation set.

Singleton Mention Filter for OntoNotes Corpus. We employ the Illinois-Coref system (Chang et al., 2012) to extract system mentions for our OntoNotes experiments, and observe that the num-

[1] http://code.google.com/p/reference-coreference-scorers/

[2] See http://research.engr.oregonstate.edu/dral/ for our software.

[3] http://sourceforge.net/p/lemur/wiki/RankLib/

ber of predicted mentions is thrice the number of gold mentions. Since the training data provides the clustering supervision for only gold mentions, it is not clear how to train with the system mentions that are not part of gold mentions. A common way of dealing with this problem is to treat all the extra system mentions as singleton clusters (Durrett and Klein, 2013; Chang et al., 2013). However, this solution most likely will not work with our current feature representation (i.e., NEW action is represented as a single indicator feature). Recall that to predict these extra system mentions as singleton clusters with our incremental clustering approach, the learned model should first predict a NEW action while processing these mentions to form a temporary singleton cluster, and then refrain from merging any of the subsequent mentions with that cluster so that it becomes a singleton cluster in the final clustering output. However, in OntoNotes corpus, the training data does not include singleton clusters for the gold mentions. Therefore, only the large number (57%) of system mentions that are not part of gold mentions will constitute the set of singleton clusters. This leads to a highly imbalanced learning problem because our model needs to learn (the weight of the single indicator feature) to predict NEW as the best action for a large set of mentions, which will bias our model to predict large number of NEW actions during testing. As a result, we will generate many singleton clusters, which will hurt the recall of the mention detection after post-processing. Therefore, we aim to learn a singleton mention filter that will be used as a pre-processor before training and testing to overcome this problem. We would like to point out that our filter is complementary to other solutions (e.g., employing features that can discriminate a given mention to be anaphoric or not in place of our single indicator feature, or using a customized loss to weight our ranking examples for cost-sensitive training)(Durrett and Klein, 2013).

Filter Learning. The singleton mention filter is a classifier that will label a given mention as "singleton" or not. We represent each mention m in a document by averaging the mention-pair features $\phi(m, m')$ of the k-most similar mentions (obtained by ranking all other mentions m' in the document with a learned ranking function R given m) and then learn a decision-tree classifier by optimizing the F1 loss. We learn the mention-ranking function R by optimizing the recall of positive pairs for a given k, and employ LambdaMART as our base ranker. The hyper-parameters are tuned based on the performance on the validation set.

6.2 Results

We first describe the results of the learned singleton mention filter, and then the performance of our Prune-and-Score approach with and without the filter. Next, we compare the results of our approach with several state-of-the-art approaches for coreference resolution.

Singleton Mention Filter Results. Table 1 shows the performance of the learned singleton mention filter with $k = 2$ noting that the results are robust for all values of $k \geq 2$. As we can see, the learned filter improves the precision of the mention detection with only small loss in the recall of gold mentions.

	Mention Detection Accuracy		
	P	R	F1
Before-filtering	43.18% (16664/38596)	86.99% (16664/19156)	57.71%
After-filtering	79.02% (15516/19640)	80.98% (15516/19156)	79.97%

Table 1: Performance of the singleton mention filter on the OntoNotes 5.0 development set. The numerators of the fractions in the brackets show the exact numbers of mentions that are matched with the gold mentions.

Prune-and-Score Results. Table 2 shows the performance of Prune-and-Score approach with and without the singleton mention filter. We can see that the results with filter are much better than the corresponding results without the filter. These results show that our approach can benefit from having a good singleton mention filter.

Filter settings	MUC	B^3	$CEAF_{\phi 4}$	CoNLL
OntoNotes 5.0 Dev Set w. Predict Ment.				
O.S. (w.o. Filter)	66.73	53.40	44.23	54.79
P&S (w.o. Filter)	65.93	52.96	50.24	56.38
P&S (w. Filter)	**71.18**	**58.87**	**57.88**	**62.64**

Table 2: Performance of Prune-and-Score approach with and without the singleton mention filter, and Only-Score approach without the filter.

Table 3 shows the performance of different configurations of our Prune-and-Score approach. As we can see, Prune-and-Score gives better results than the configuration where we employ only the scoring function ($b = \infty$) for small values of b.

	MUC			B^3			$CEAF_{\phi4}$			CoNLL
	P	R	F1	P	R	F1	P	R	F1	Avg-F1
a. Results on OntoNotes 5.0 Test Set with Predicted Mentions										
Prune-and-Score	81.03	66.16	**72.84**	66.90	51.10	57.94	68.75	44.34	53.91	61.56
Only-Scoring	75.95	61.53	67.98	63.94	47.37	54.42	58.54	49.76	53.79	58.73
HOTCoref	67.46	74.3	70.72	54.96	62.71	**58.58**	52.27	59.4	**55.61**	**61.63**
CPL^3M	-	-	69.48	-	-	57.44	-	-	53.07	60.00
Berkeley	74.89	67.17	70.82	64.26	53.09	58.14	58.12	52.67	55.27	61.41
Fernandes et al., 2012	75.91	65.83	70.51	65.19	51.55	57.58	57.28	50.82	53.86	60.65
Stanford	65.31	64.11	64.71	56.54	48.58	52.26	46.67	52.29	49.32	55.43
b. Results on OntoNotes 5.0 Test Set with Gold Mentions										
Prune-and-Score	88.10	85.85	86.96	76.82	76.16	76.49	80.90	74.06	**77.33**	**80.26**
Only-Scoring	86.96	84.52	85.73	74.51	74.25	74.38	79.04	70.67	74.62	78.24
CPL^3M	-	-	84.80	-	-	**78.74**	-	-	68.75	77.43
Berkeley	85.73	89.26	**87.46**	78.23	75.11	76.63	82.89	70.86	76.40	80.16
Stanford	89.94	78.17	83.64	81.75	68.95	74.81	73.97	61.20	66.98	75.14
c. Results on ACE2004 Culotta Test Set with Gold Mentions										
Prune-and-Score	85.57	72.68	**78.60**	90.09	77.02	**83.04**	74.64	86.02	**79.42**	**80.35**
Only-Scoring	82.75	69.25	75.40	88.54	74.22	80.75	73.69	85.22	78.58	78.24
CPL^3M	-	-	78.29	-	-	82.20	-	-	79.26	79.91
Stanford	82.91	69.90	75.85	89.14	74.05	80.90	75.67	77.45	76.55	77.77
d. Results on ACE2004 Newswire with Gold Mentions										
Prune-and-Score	89.72	75.72	**82.13**	90.89	76.15	**82.87**	72.43	86.83	**78.69**	**81.23**
Only-Scoring	86.92	76.49	81.37	88.10	75.83	81.51	73.15	84.31	78.05	80.31
Easy-first	-	-	80.1	-	-	81.8	-	-	-	-
Stanford	84.75	75.34	79.77	87.50	74.59	80.53	73.32	81.49	77.19	79.16
e. Results on MUC6 Test Set with Gold Mentions										
Prune-and-Score	89.53	82.75	86.01	86.48	76.18	**81.00**	60.74	80.33	**68.68**	**78.56**
Only-Scoring	86.77	80.96	83.76	81.72	72.99	77.11	57.56	75.38	64.91	75.26
Easy-first	-	-	**88.2**	-	-	77.5	-	-	-	-
Stanford	91.19	69.54	78.91	91.07	63.39	74.75	62.43	69.62	65.83	73.16

Table 4: Comparison of Prune-and-Score with state-of-the-art approaches. Metric values reflect version 7 of CoNLL scorer.

The performance is clearly better than the degenerate case ($b = \infty$) over a wide range of b values, suggesting that it is not necessary to carefully tune the parameter b.

Pruning param. b	MUC	B^3	$CEAF_{\phi4}$	CoNLL
OntoNotes 5.0 Dev Set w. Predict Ment.				
2	69.12	56.80	56.30	60.74
3	70.50	57.89	57.24	61.88
4	71.00	58.65	57.41	62.35
5	**71.18**	**58.87**	**57.88**	**62.64**
6	70.93	58.66	57.85	62.48
8	70.12	58.13	57.37	61.87
10	70.24	58.34	56.27	61.61
20	67.97	57.73	56.63	60.78
∞	67.03	56.31	55.56	59.63

Table 3: Performance of Prune-and-Score approach with different values of the pruning parameter b. For $b = \infty$, Prune-and-Score becomes an Only-Scoring algorithm.

Comparison to State-of-the-Art. Table 4 shows the results of our **Prune-and-Score** approach compared with the following state-of-the-art coreference resolution approaches: **HOTCoref** system (Björkelund and Kuhn, 2014); **Berkeley** system with the FINAL feature set (Durrett and Klein, 2013); **CPL^3M** system (Chang et al., 2013); **Stanford** system (Lee et al., 2013); **Easy-first** system (Stoyanov and Eisner, 2012); and **Fernandes et al., 2012** (Fernandes et al., 2012). **Only Scoring** is the special case of our Prune-and-Score approach where we employ only the scoring function. This corresponds to existing incremental approaches (Daumé III, 2006; Rahman and Ng, 2011b). We report the best published results for CPL^3M system, Easy-first, and Fernandes et al., 2012. We ran the publicly available software to generate the results for Berkeley and Stanford systems with the updated CoNLL scorer. We include the results of Prune-and-Score for best b on the development set with singleton mention filter for the comparison. In Table 4, '-' indicates that we could not find published results for those cases. We see

that results of the Prune-and-Score approach are comparable to or better than the state-of-the-art including Only-Scoring.

7 Conclusions and Future Work

We introduced the Prune-and-Score approach for greedy coreference resolution whose main idea is to learn a pruning function along with a scoring function to effectively guide the search. We showed that our approach improves over the methods that only learn a scoring function, and gives comparable or better results than several state-of-the-art coreference resolution systems.

Our Prune-and-Score approach is a particular instantiation of the general idea of learning nearly-sound constraints for pruning, and leveraging the learned constraints to learn improved heuristic functions for guiding the search (See (Chen et al., 2014) for another instantiation of this idea for multi-object tracking in videos). Therefore, other coreference resolution systems (Chang et al., 2013; Durrett and Klein, 2013; Björkelund and Kuhn, 2014) can also benefit from this idea. One way to further improve the peformance of our approach is to perform a search in the Limited Discrepancy Search (LDS) space (Doppa et al., 2014b) using the learned functions.

Future work should apply this general idea to other natural language processing tasks including dependency parsing (Nivre et al., 2007) and information extraction (Li et al., 2013). We would expect more beneficial behavior with the pruning constraints for problems with large action sets (e.g., labeled dependency parsing). It would be interesting and useful to generalize this approach to search spaces where there are multiple target paths from the initial state to the terminal state, e.g., as in the Easy-first framework.

Acknowledgments

Authors would like to thank Veselin Stoyanov (JHU) for answering several questions related to the Easy-first and Reconcile systems; Van Dang (UMass, Amherst) for technical discussions related to the RankLib library; Kai-Wei Chang (UIUC) for the help related to the Illinois-Coref mention extractor; and Greg Durrett (UC Berkeley) for his help with the Berkeley system. This work was supported in part by NSF grants IIS 1219258, IIS 1018490 and in part by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under Contract No. FA8750-13-2-0033. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF, the DARPA, the Air Force Research Laboratory (AFRL), or the US government.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pages 294–303.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 47–57, Baltimore, Maryland, June. Association for Computational Linguistics.

Christopher Burges. 2010. From RankNet to LambdaRank to LambdaMART: An overview. *Microsoft Technical Report*, (MSR-TR-2010).

Kai-Wei Chang, Rajhans Samdani, Alla Rozovskaya, Mark Sammons, and Dan Roth. 2012. Illinois-Coref: The UI system in the CoNLL-2012 shared task. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 113–117, Jeju Island, Korea, July. Association for Computational Linguistics.

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A constrained latent variable model for coreference resolution. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pages 601–612.

Sheng Chen, Alan Fern, and Sinisa Todorovic. 2014. Multi-object tracking via constrained sequential labeling. In *To appear in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

William W. Cohen and Vitor Rocha de Carvalho. 2005. Stacked sequential learning. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 671–676.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 175–182.

Aron Culotta, Michael L. Wick, and Andrew McCallum. 2007. First-order probabilistic models for coreference resolution. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, pages 81–88.

Hal Daumé III. 2006. *Practical Structured Learning Techniques for Natural Language Processing*. Ph.D. thesis, University of Southern California, Los Angeles, CA.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pages 660–669.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. 2014a. HC-Search: A learning framework for search-based structured prediction. *Journal of Artificial Intelligence Research (JAIR)*, 50:369–407.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. 2014b. Structured prediction via output space search. *Journal of Machine Learning Research (JMLR)*, 15:1317–1350.

Janardhan Rao Doppa, Jun Yu, Chao Ma, Alan Fern, and Prasad Tadepalli. 2014c. HC-Search for multi-label prediction: An empirical study. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1971–1982.

Greg Durrett, David Leo Wright Hall, and Dan Klein. 2013. Decentralized entity-level modeling for coreference resolution. In *Proceedings of Association of Computational Linguistics (ACL) Conference*, pages 114–124.

Pedro F. Felzenszwalb and David A. McAllester. 2007. The generalized A* architecture. *Journal of Artificial Intelligence Research (JAIR)*, 29:153–190.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. International Conference on Computational Natural Language Learning (CoNLL), pages 41–48.

Thomas Finley and Thorsten Joachims. 2005. Supervised clustering with support vector machines. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 217–224.

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

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*.

Hannaneh Hajishirzi, Leila Zilles, Daniel S. Weld, and Luke S. Zettlemoyer. 2013. Joint coreference resolution and named-entity linking with multi-pass sieves. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 289–299.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning Journal (MLJ)*, 75(3):297–325.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML*.

Roni Khardon. 1999. Learning to take actions. *Machine Learning Journal (MLJ)*, 35(1):57–90.

Michael Lam, Janardhan Rao Doppa, Xu Hu, Sinisa Todorovic, Thomas Dietterich, Abigail Reft, and Marymegan Daly. 2013. Learning to detect basal tubules of nematocysts in sem images. In *ICCV Workshop on Computer Vision for Accelerated Biosciences (CVAB)*. IEEE.

Heeyoung Lee, Angel X. Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 73–82.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 25–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

Andrew Mccallum and Ben Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 905–912. MIT Press.

MUC6. 1995. Coreference task definition. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 335–344.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of Association of Computational Linguistics (ACL) Conference*, pages 104–111.

NIST. 2004. The ACE evaluation plan.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Proceedings of the Joint Conference on EMNLP and CoNLL: Shared Task*, pages 1–40.

Altaf Rahman and Vincent Ng. 2011a. Coreference resolution with world knowledge. In *Proceedings of Association of Computational Linguistics (ACL) Conference*, pages 814–824.

Altaf Rahman and Vincent Ng. 2011b. Narrowing the modeling gap: A cluster-ranking approach to coreference resolution. *Journal of Artificial Intelligence Research (JAIR)*, 40:469–521.

Lev-Arie Ratinov and Dan Roth. 2012. Learning-based multi-sieve co-reference resolution with knowledge. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP) Conference*, pages 1234–1244.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. *Journal of Machine Learning Research - Proceedings Track*, 15:627–635.

Wee Meng Soon, Daniel Chung, Daniel Chung Yong Lim, Yong Lim, and Hwee Tou Ng. 2001. A machine learning approach to coreference resolution of noun phrases.

Veselin Stoyanov and Jason Eisner. 2012. Easy-first coreference resolution. In *Proceedings of International Conference on Computational Linguistics (COLING)*, pages 2519–2534.

Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. 2010. Coreference resolution with reconcile. In *Proceedings of Association of Computational Linguistics (ACL) Conference*, pages 156–161.

Marc B. Vilain, John D. Burger, John S. Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *MUC*, pages 45–52.

David Weiss and Benjamin Taskar. 2010. Structured prediction cascades. *Journal of Machine Learning Research - Proceedings Track*, 9:916–923.

Michael L. Wick, Khashayar Rohanimanesh, Kedar Bellare, Aron Culotta, and Andrew McCallum. 2011. SampleRank: Training factor graphs with atomic gradients. In *Proceedings of International Conference on Machine Learning (ICML)*.

Michael L. Wick, Sameer Singh, and Andrew McCallum. 2012. A discriminative hierarchical model for fast coreference at large scale. In *Proceedings of Association of Computational Linguistics (ACL) Conference*, pages 379–388.

Yuehua Xu, Alan Fern, and Sung Wook Yoon. 2009. Learning linear ranking functions for beam search with application to planning. *Journal of Machine Learning Research (JMLR)*, 10:1571–1610.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *Proceedings of International Conference on Machine Learning (ICML)*.

Jiaping Zheng, Luke Vilnis, Sameer Singh, Jinho D. Choi, and Andrew McCallum. 2013. Dynamic knowledge-base alignment for coreference resolution. In *Conference on Computational Natural Language Learning (CoNLL)*.

Summarizing Online Forum Discussions – Can Dialog Acts of Individual Messages Help?

Sumit Bhatia[1], **Prakhar Biyani**[2] **and Prasenjit Mitra**[2]
[1]IBM Almaden Research Centre, 650 Harry Road, San Jose, CA 95123, USA
[2]Information Science and Technology, Pennsylvania State University, University Park, PA 16802
sumit.bhatia@us.ibm.com, {pxb5080, pmitra}@ist.psu.edu

Abstract

A typical discussion thread in an online forum spans multiple pages involving participation from multiple users and thus, may contain multiple view-points and solutions. A user interested in the topic of discussion or having a problem similar to being discussed in the thread may not want to read all the previous posts but only a few selected posts that provide her a concise summary of the ongoing discussion. This paper describes an extractive summarization technique that uses textual features and dialog act information of individual messages to select a subset of posts. Proposed approach is evaluated using two real life forum datasets.

1 Introduction

In recent times, online discussion boards (or *forums*) have become quite popular as they provide an easily accessible platform to users in different parts of the world to come together, share information and discuss issues of common interest. The archives of web forums contain millions of discussion threads and act as a valuable repository of human generated information that can be utilized for various applications. Oftentimes, the discussions in a thread span multiple pages involving participation from multiple users and thus, may contain multiple view-points and solutions. In such a case, the end-user may prefer a concise summary of the ongoing discussion to save time. Further, such a summary helps the user to understand the background of the whole discussion as well as provides an overview of different view-points in a time efficient manner. In addition to generic forums on the web, automatic forum summarization methods can prove to be useful for various domain specific applications, such as helping students and supporting tutors in virtual learning environments (Carbonaro, 2010).

A typical discussion thread in a web forum consists of a number of individual *posts* or messages posted by different participating users. Often, the thread initiator posts a question to which other users reply, leading to an active discussion. As an example, consider the discussion thread shown in Figure 1 where the thread starter describes his problem about the missing headphone switch in his Linux installation. In the third post in the thread, some other user asks about some clarifying details and in the next post the topic starter provides the requested details that makes the problem clearer. On receiving additional details about the problem, some other user provides a possible solution to the problem (fifth post). The topic starter tries the suggested solution and reports his experience in the next post (sixth post). Thus, we see that each individual post in a discussion thread serves a different purpose in the discussion and we posit that *identifying the purpose of each such post is essential for creating effective summaries of the discussions.* Intuitively, the most important messages in a discussion are the ones that describe the problem being discussed and the solutions being proposed to solve the problem.

The *role* of an individual message in a discussion is typically specified in terms of *dialog acts*. There have been efforts to automatically assign dialog acts to messages in online forum discussions (Jeong et al., 2009; Joty et al., 2011; Bhatia et al., 2012) and also using dialog acts for linguistic analysis of forum data, such as in subjectivity analysis of forum threads (Biyani et al., 2012; Biyani et al., 2014). In this paper, we describe our initial efforts towards addressing the problem of *automatically creating summaries of such online discussion threads*. We frame forum summarization as a classification problem and identify messages that should be included in a summary of the

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2127–2131,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

discussion. In addition to textual features, we employ dialog act labels of individual messages for summarization and show that incorporating dialog acts leads to substantial improvements in summarization performance.

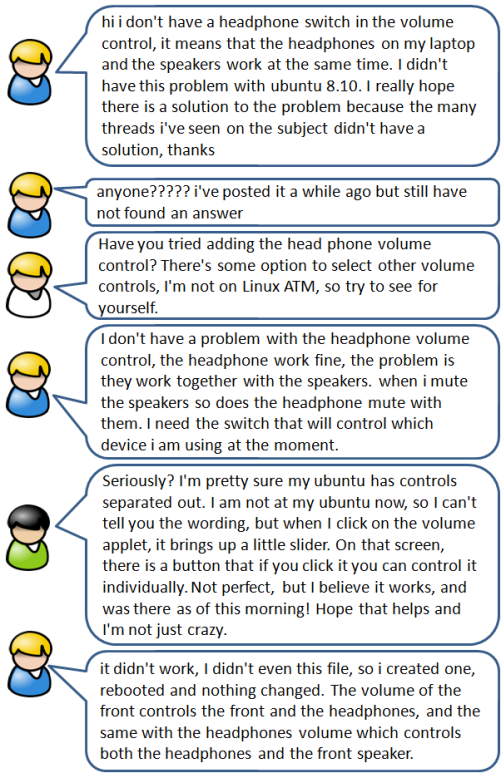


Figure 1: An example thread illustrating different role played by each post in the discussion. Different users are indicated by different colors.

2 Definition of Dialog Acts Used

We use the same set of dialog acts as defined by Bhatia et al. (2012). Note that based on the application context and requirements new dialog acts can be defined and added.

1. Question: The poster asks a question which initiates discussion in the thread. This is usually the first post in the thread but not always. Often, the topic initiator or some other user may ask other related questions in the thread.

2. Repeat Question: Some user repeats a previously asked question (e.g. *Me too having the same problem.*).

3. Clarification: The poster asks clarifying questions in order to gather more details about the problem or question being asked. For example, *Could you provide more details about the issue you are facing*.

4. Further Details: The poster provides more details about the problem as asked by other fellow posters.

5. Solution: The poster suggests a solution to the problem being discussed in the thread.

6. Positive Feedback: Somebody tries the suggested solution and provides a positive feedback if the solution worked.

7. Negative Feedback: Somebody tries the suggested solution and provides a negative feedback if the solution did not work.

8. Junk: There is no useful information in the post. For example, someone justs posts a smiley or some comments that is not useful to topic being discussed. For example, *"bump", "sigh"*, etc., or messages posted by forum moderators such as *this thread is being closed for discussion.*

3 Proposed Approach for Thread Summarization

In general, text summarization techniques can be classified into two categories, namely extractive Summarization, and Abstractive Summarization (Hahn and Mani, 2000). Extractive summarization involves extracting salient units of text (e.g., sentences) from the document and then concatenating them to form a shorter version of the document. Abstractive summarization, on the other hand, involves generating new sentences by utilizing the information extracted from the document corpus (Carenini and Cheung, 2008), and often involves advanced natural language processing tools such as parsers, lexicons and grammars, and domain-specific knowledge bases (Hahn and Mani, 2000). Owing to their simplicity and good performance, extractive summarization techniques are often the preferred tools of choice for various summarization tasks (Liu and Liu, 2009) and we also adopt an extractive approach for discussion summarization in this work.

3.1 Summarization Unit – Individual Sentence vs Individual Message

Before we can perform extractive summarization on discussion threads, we need to define an appropriate text unit that will be used to construct the desired summaries. For typical summarization tasks, a sentence is usually treated as a unit of text and summaries are constructed by extracting most relevant sentences from a document. However, a typical discussion thread is different from

a generic document in that the text of a discussion thread is created by multiple authors (users participating in the thread). Further, the text of a discussion can be divided into individual user messages, each message serving a specific role in the whole discussion. In that sense, summarizing a discussion thread is similar to the task of multi-document summarization where content of multiple documents that are topically related is summarized simultaneously to construct an inclusive, coherent summary. However, we also note that an individual user message in a discussion is much smaller than a stand-alone document (compare $3 \sim 4$ sentences in a message to a few dozen sentences in a document). Thus, the sentences in a message are much more coherent and contextually related to each other than in a stand-alone document. Hence, selecting just a few sentences from a message may lead to loss of context and make the resulting summaries hard to comprehend. Therefore, in this work, we choose each individual message as a text unit and thus, the thread summaries are created by extracting most relevant posts from a discussion.

3.2 Framing Thread Summarization as Post Classification

We consider the problem of extracting relevant posts from a discussion thread as a binary classification problem where the task is to classify a given post as either belonging to the summary or not. We perform classification in a supervised fashion by employing following features.

1. Similarity with Title (TitleSim): This feature is computed as the cosine similarity score between the post and the title of the thread.

2. Length of Post (Length): The number of unique words in the post.

3. Post Position (Position): The normalized position of the post in the discussion thread. It is defined as follows:

$$\frac{\text{Position of the post in the thread}}{\text{Total \# of posts in the thread}} \quad (1)$$

4. Centroid Similarity (Centroid): This feature is obtained by computing the cosine similarity score between the post document vector and the vector obtained as the centroid of all the post vectors of the thread. Similarity with centroid measures the *relatedness* of each post with the underlying discussion topic. A post with a higher similarity score with the thread centroid vector indicates that the post better represents the basic ideas of the thread.

5. Inter Post Similarity: This feature is computed by taking the mean of the post's cosine similarity scores with all the other posts in the thread.

6. Dialog Act Label (Class): This is a set of binary features indicating the dialog act class label of the post. We have one binary feature corresponding to each dialog act.

4 Experimental Evaluation

4.1 Data Description

We used the dataset used by Bhatia et al. (2012) that consists of randomly sampled 100 threads from two different online discussion forums – `ubuntuforums.org` and `tripadvisor.com`. There are a total of 556 posts in the 100 threads from Ubuntu dataset and 916 posts in 100 threads from NYC dataset. The associated dialog act labels of individual messages in each of the threads are also available.

Next, for creating data for the summarization task, two independent human evaluators (H1 and H2) were recruited to create summaries of the discussion threads in the two datasets. For each thread, the evaluators were asked to read the whole discussion and write a summary of the discussion in their own words. The annotators were requested to keep the length of summaries roughly between 10% and 25% of the original text length. Thus for each thread, we obtain two human written summaries.

These hand-written summaries were then used to identify most relevant posts in a discussion thread in a manner similar to one used by Rambow et al. (2004). We compute cosine similarity scores for each post in the thread with the corresponding thread summary and the top k ranked posts are then selected to be part of the summary of the thread. The number k is determined by the compression factor used for creating summaries. We choose a compression factor of 20%. The top k ranked posts, thus constitute the gold summary of each thread. Note that we obtain two gold summaries for each thread – one corresponding to each evaluator. This summarization data can be downloaded for research purposes from `http://sumitbhatia.net/source/datasets.html`.

Evaluator	Method	Ubuntu		NYC	
		Precision	F-1	Precision	F-1
H1	Baseline	0.39	0.53	0.32	0.46
	Without Dialog Acts	0.578	0.536	0.739	0.607
	With Dialog Acts	0.620	0.608	0.760	0.655
	Gain	+7.27%	+13.43%	+2.84%	+7.91%
H2	Baseline	0.38	0.52	0.31	0.45
	Without Dialog Acts	0.739	0.607	0.588	0.561
	With Dialog Acts	0.760	0.655	0.652	0.588
	Gain	+14.94%	+20.53%	+10.88%	+4.81%

Table 1: Results of post classification for summarization task. H1 and H2 correspond to the two human evaluators. Percentage improvements obtained by addition of post class label information is also reported.

4.2 Baseline

As a baseline method, we use a rule based classifier that classifies all the *Question* and *Solution* posts in a thread as belonging to the summary and discards the remaining posts.

4.3 Results and Discussions

We used Naive Bayes classifier as implemented in the Weka machine learning toolkit (Hall et al., 2009) for classification experiments. We trained the classifier on 75% of the data and used the remaining 25% for testing. Table 1 reports the classification results using *(i)* the baseline method,*(ii)* features 1–5 only, and *(iii)* using all the features (dialog act labels, in addition to the five features). For both the datasets, we observe that incorporating dialog act information along with textual features results in performance gain across all reported metrics. The strong performance improvements achieved for the two datasets corroborate the proposed hypothesis that knowing the role of each individual message in an online discussion can help create better summaries of discussion threads. Further, we observe that the precision values are very low for the baseline algorithm (from 0.31 to 0.39) with moderate F-1 values (0.45 to 0.53), indicating a higher recall. This means that even though many of the posts in the gold summaries belong to question and solution categories, not all the posts belonging to these two categories are useful for summarization. Using textual features and dialog act labels in a supervised machine learning framework captures the distinguishing characteristics of *in-summary* and *out of summary* posts and thus, yields a much better classification performance.

5 Related Work

Among various applications of text summarization, work on E-Mail thread summarization (Rambow et al., 2004; Cohen et al., 2004) can be considered as closely related to the problem discussed in this paper. An E-Mail thread is similar to a forum discussion thread in that it involves back and forth communication with the participants, however, the problem of discussion thread summarization is very different (and difficult) due to a relatively larger number of participants, highly informal and noisy language, and frequent topic drifts in discussions. Zhou and Hovy (2005) identify clusters in internet relay chats (irc) and then employ lexical and structural features to summarize each cluster. Ren et al. (2011) have proposed a forum summarization algorithm that models the reply structures in a discussion thread.

6 Conclusions and Future Work

We proposed that dialog act labels of individual messages in an online forums can be helpful in summarizing discussion threads. We framed discussion thread summarization as a binary classification problem and tested our hypothesis on two different datasets. We found that for both the datasets, incorporating dialog act information as features improves classification performance as measured in terms of precision and F-1 measure. As future work, we plan to explore various other forum specific features such as user reputation and quality of content to improve summarization performance.

References

Sumit Bhatia, Prakhar Biyani, and Prasenjit Mitra. 2012. Classifying user messages for managing web forum data. In *Proceedings of the 15th International Workshop on the Web and Databases 2012, WebDB 2012, Scottsdale, AZ, USA, May 20, 2012*, pages 13–18.

Prakhar Biyani, Sumit Bhatia, Cornelia Caragea, and Prasenjit Mitra. 2012. Thread specific features are helpful for identifying subjectivity orientation of online forum threads. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pages 295–310.

Prakhar Biyani, Sumit Bhatia, Cornelia Caragea, and Prasenjit Mitra. 2014. Using non-lexical features for identifying factual and opinionative threads in online forums. *Knowledge-Based Systems, In Press, doi = http://dx.doi.org/10.1016/j.knosys.2014.04.048.*

Antonella Carbonaro. 2010. Towards an automatic forum summarization to support tutoring. In MiltiadisD. Lytras, Patricia Ordonez De Pablos, David Avison, Janice Sipior, Qun Jin, Walter Leal, Lorna Uden, Michael Thomas, Sara Cervai, and David Horner, editors, *Technology Enhanced Learning. Quality of Teaching and Educational Reform*, volume 73 of *Communications in Computer and Information Science*, pages 141–147. Springer Berlin Heidelberg.

Giuseppe Carenini and Jackie Chi Kit Cheung. 2008. Extractive vs. nlg-based abstractive summarization of evaluative text: The effect of corpus controversiality. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 33–41. Association for Computational Linguistics.

William W. Cohen, Vitor R. Carvalho, and Tom M. Mitchell. 2004. Learning to Classify Email into "Speech Acts". In *EMNLP*, pages 309–316. ACL.

Udo Hahn and Inderjeet Mani. 2000. The challenges of automatic summarization. *Computer*, 33(11):29–36.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1).

Minwoo Jeong, Chin-Yew Lin, and Gary Geunbae Lee. 2009. Semi-supervised speech act recognition in emails and forums. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1250–1259.

Shafiq R. Joty, Giuseppe Carenini, and Chin-Yew Lin. 2011. Unsupervised modeling of dialog acts in asynchronous conversations. In *IJCAI*, pages 1807–1813. IJCAI/AAAI.

Fei Liu and Yang Liu. 2009. From extractive to abstractive meeting summaries: can it be done by sentence compression? In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 261–264. Association for Computational Linguistics.

O. Rambow, L. Shrestha, J. Chen, and C. Laurdisen. 2004. Summarizing email threads. *Proceedings of HLT-NAACL 2004: Short Papers*.

Zhaochun Ren, Jun Ma, Shuaiqiang Wang, and Yang Liu. 2011. Summarizing web forum threads based on a latent topic propagation process. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 879–884, New York, NY, USA. ACM.

Liang Zhou and Eduard Hovy. 2005. Digesting virtual "geek" culture: The summarization of technical internet relay chats. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 298–305, Stroudsburg, PA, USA. Association for Computational Linguistics.

Author Index